This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Managed Dynamic VPN Service: Core Capacity Sharing Schemes For Improved VPN Performance

Ravi S.Ravindran[1], Changcheng Huang[1], K.Thulasiraman[2]
(1.Carleton University, Ottawa, 2.University of Oklahoma, Norman)

Abstract—Managed service framework enables a service provider to offer more demanding and revenue generating services. IETF proposed Provider Provisioned IP-VPN service is a well known managed service. In [6] we first proposed a new framework to enable managed dynamic VPN service using the notion of topology abstraction. In [7] we proposed several distributed heuristics that can be applied in the context of [6]. The focus of this paper is to study the problem of enabling dynamic managed service using topology abstraction in a centralized manner whose objective is to maximize the network utilization and VPN call performance. The centralized scheme proposed in this paper applies the maximum concurrent flow theory and proposes two extensions to it. The two extension aims at improving the conservative nature of using maximum concurrent flow theory by improving the statistical multiplexing of available core network resources among the VPNs. We study the proposed approaches using a simulation environment of an IP/MPLS network providing managed IP-VPN service with appropriate extensions required to realize the components of the Managed Dynamic VPN Service as proposed in [6].

***Keywords*: IP-VPN Service, Topology Abstraction, Maximum Concurrent Flow.**

I. INTRODUCTION

A Virtual Private Network (VPN) refers to a distributed network of geographically dispersed network entities belonging to the same authority virtualized as a private network by overlaying it over a service provider's core network. A common form of VPN service that is well known is to connect branch offices belonging to an enterprise. In this form of connectivity there are two basic models of VPN depending on whether the VPN Service Provider (VSP) takes part in routing the packets from a VPN site. Based on this distinction we have Customer Premise Equipment (CPE) based VPN, in which the VSP is unaware of any VPN existence. In a CPE based VPN the provider only provides fixed bit rate pipes or virtual circuits over ATM or FrameRelay transport technologies between the sites. The other case is the peer based VPN service, where the VSP is VPN aware, and hence in addition to providing VPN connectivity, also participates in VPN routing. The latter case is of interest here. For insight into various areas of VPN research related to this paper, refer to works in [1]-[4] and references therein, which we do not summarize because of space constraint. In recent years IETF has evolved solutions to enable provider provisioned IP-VPN service over MPLS based transport network. A relevant standard proposed to realize managed IP-VPN service is the BGP/MPLS based solution proposed in [5].

The motivation for the requirement of dynamic VPN service proposed in [6] is based on the premise that, future networking services will have to address the need of bandwidth intensive applications such as high definition Video broadcast/multicast from an enterprises and access service providers, or mass online Interactive gaming that requires significant bandwidths for short windows of time during a day. The dynamic VPN service using topology abstraction proposed in [6] tries to achieve two key objectives. First is to enable dynamic bandwidth service that would enable the VSP to share information about the resource availability in the core with the VPNs using the notion of topology abstraction formerly used in routing protocols like PNNI for ATM networks for scalability reasons. Second is to have the proposed framework to be realized over current managed IP-VPN solutions such as in [5]. We henceforth refer to the service definition proposed in [6] as Managed Dynamic VPN Service (MDVS). One of the challenges of MDVS is to solve the problem of capacity exposed to the set of VPNs subscribing to MDVS. We called this problem as the VPN Topology Abstraction (VPN-TA) problem, and proposed distributed solutions to solve it in [7]. The key contribution of this paper is the use of Maximum-Concurrent flow (MConF) theory to solve a similar problem applied in a centralized context. In addition to showing a way to adapt MConF theory to solve VPN-CS problem, two improvements to it have also been proposed that aims to improve the observed conservative nature of applying MConF.

The rest of the paper is organized as follows. Section 2 summarizes briefly MDVS framework and process proposed in [6]. Section 3 and 4 defines and formulates the VPN Core Capacity Sharing problem and discusses the heuristic and its extensions based on maximum concurrent flow theory. Section 5 presents simulation results that discuss the performance of heuristics suggested in this paper over an IP/MPLS environment implementing MDVS.

1-4244-0353-7/07/$25.00 ©2007 IEEE

Fig.1.1 shows the graph theory notations used in the paper.

Notation	Description
$G(V,E)$	VSP's core network
VPN_{abs}	Set of MDVS VPN Subscribers
PE	Set of all PE border nodes
CE	Set of all CE border nodes
PE_k	Set of border nodes hosting VPN k
CE_k	Set of CE nodes of VPN k
$CE_{k,b}$	Set of CE nodes of VPN k neighboring PE node b
$G_{abs(k)(x)}(V_k, E_k)$	Abstract topology of type x for VPN k
$SLA(k)$	SLA parameters for VPN k
$TT_{abs(k)}$	Abstract topology type subscribed by VPN k
RI_k	Abstract topology refresh interval for VPN k
$VPN_{abs}(s_k, d_k)$.	Set of VPN's that include src-dest pair (s_k, d_k)
FM	Fully Meshed type abstraction
SR	Source Rooted type abstraction
ST	Star topology type abstraction
SN	Simple node type abstraction

Fig 1.1: Graph Theory Notations

II. MANAGED DYNAMIC VPN SERVICE ARCHITECTURE

In this section we summarize the framework proposed in [6] to set the context for the remaining discussion. One of the main building block over which the solution in [5] builds on is the use of VPN specific Virtual Routing and Forwarding instance (VRF) tables within PE routers to logically separate the VPN contexts. To realize MDVS we augment the architecture in [5] with three additional components: *VPN SLA Database*, *Abstract Topology Generation Module,* and *Central Server* (CS). The *VPN SLA Database* stores the SLA definitions for all the VPN's subscribed to the MDVS service. The two important SLA parameters of a MDVS SLA definition [6] are the *Abstract Topology Type* (e.g. Fully Meshed, Source Rooted Star, Star, Simple Node abstract topology) and the *Abstract Topology Refresh Interval* whose use is explained below. *Abstract Topology Generation Module* is responsible for generating abstract graphs applying the *Abstract Topology Type* SLA parameter for the VPNs. The abstract graphs are generated from sub-graphs that are either computed independently by border nodes as explored in [7], or by a *Central Server* as discussed in this paper. The *Abstract Topology Generation Module* updates the abstract topologies to the VPNs periodically applying the *Abstract Topology Refresh Interval* SLA parameter. The abstract graphs flooded to the VPNs are used by the CE nodes to compute an end to end path traversing the VSP's core network, and check on the availability of the desired QoS.

Fig 2.1 shows the steps involved in the topology abstraction process executed by a PE node b. $VPN_{abs(b)}$ here represents the set of VPNs hosted on the border node b. The procedure iterates for each of the VPNs for which abstract topology needs to be generated. The pseudo code shown in the figure executes in three stages. As part of the first iteration, *Step 1* identifies the subsets of border nodes PE_k and CE_k for each VPN k. *Step 2* is the critical step where an intermediate sub-graph $SG(V_k, E_k)$ is determined using the core network from which the abstract topology is derived. The partition graph in this case is obtained from the CS. The interaction between the CS and the border nodes can be implemented such that the computed partition graph for a given VPN is either periodically or on-demand given to the border nodes. The computation of the sub-graph $SG(V_k, E_k)$ is formulated as a problem in the next section which is solved using MConF flow theory by the CS. *Step 3* uses this computed partitioned graph $SG(V_k,E_k)$ to derive a fully meshed abstract topology for the given VPN k, which we represent as $G_{abs(k)(FM)}(V_k,E_k)$. The second phase of iteration includes *Step 4* that applies the fairness criteria. We enforce the fairness criterion by requiring the VSP to share the available resource fairly among all the VPN's, such that for any two VPN $x, y \in VPN_{abs}$ with similar type of abstract topology type subscription i.e. $TT_{abs(x)} = TT_{abs(y)}$, sharing a common border node $(b_1,b_2) \in B$, $bw_{abs(x)}[b_1][b_2] = bw_{abs(y)}[b_1][b_2]$. *Step 5* applies the topology type SLA parameter $x=TT_{abs(k)}$ from possible types (i.e *FM, SR, ST, SN*) to the fully meshed abstract graphs $G_{abs(k)(FM)}(V_k,E_k)$ resulting in the desired topology abstraction $G_{abs(k)(x)}(V_k,E_k)$. In *Step 6* the PE updates the subset of locally attached CE nodes $CE_{k,b}$ with this abstract topology. Assuming that the partition graph $SG(V_k, E_k)$ is available, the complexity of the topology abstraction process is dominated by the *Step 2* in the first iteration, where any algorithm proposed in [7] can be applied, which are all polynomial time algorithms. If the algorithm applied from [7] is the Maximum Capacity Heuristic, the complexity of the abstraction process is $O(|VPN_{abs}|*|PE_K|*|V|^2)$.

***Abstract_Topology*(G, b, $VPN_{abs(b)}$)**
G: VSP Core Graph
b: Concerned PE node
$VPN_{abs(b)}$: Set of subscribed VPN's on border node *b*
***Begin**:*
//Generate Partition Graph and fully meshed abstraction
For each VPN $k \in VPN_{abs(b)}$
  ***Step 1** : Find set $PE_k \subset B$ and $CE_k \subset C$* for VPN *k*
  Step 2** : Obtain graph Partition $SG(V_k, E_k)$ **from CM
  ***Step 3** : Use $SG(V_k, E_k)$* to *generate Abstract Graph $G_{abs(k)(FM)}(V_k,E_k)$*
End For

//Apply fairness criteria to each VPN
For each VPN $k \in VPN_{abs(b)}$
  Step 4 : Apply fairness policy to $G_{abs(k)(FM)}(V_k,E_k)$
End For

//Apply SLA parameters to generate required abstraction
For each VPN $k \in VPN_{abs(b)}$
  ***Step 5** :* Apply $x = TT_{abs(k)}$ to $G_{abs(k)(FM)}(V_k,E_k)$ to generate $G_{abs(k)(x)}(V_k,E_k)$
  Step 6 : Update VPN k node's $CE_{k,b}$ with $G_{abs(k)(x)}(V_k,E_k)$
End For
End

Fig 2.1 Topology Abstraction Process in MDVS

III. VPN CAPACITY SHARING PROBLEM (VPN-CS)

The key challenge in MDVS is to decide the core capacity partition for a VPN from which an abstract topology can be derived. There are two objectives associated with the core partition problem. The first objective is to maximize core network utilization. This also correlates with the goal of maximizing the revenue generated out of MDVS service. The second objective is to maximize a VPN's call performance.

The parameter of interest in the context of MDVS is *VPN Call Success Ratio*. The *Call Success Ratio* of a VPN is the measure of making a right bandwidth request decision by a VPN using the abstraction provided by the VSP. This includes the calls either being computed correctly or those rejected by the VPN locally using the abstract representation of the core network. A CE node rejects a path locally when the local path computation using the abstract topology fails to find a feasible path. The partition sub-graph $SG(V_k,E_k)$ are generated by the CS which has visibility of the global topology and link state and access to MDVS service information. We call the problem solved by the CS to generate these partition graphs for each of the VPNs as the *VPN Capacity Sharing Problem*, which can be stated as follows:

VPN Capacity Sharing Problem (VPN-CS): *Given graph* $G(V,E)$ *providing MDVS to the set of VPNs* (VPN_{abs})*, the objective is to compute virtual partitions* $SG(V_k, E_k)$ $\forall$ $k \in VPN_{abs}$ *of the network so as to maximize core network utilization* U_c *and VPN call success ratio* P_v, $\forall v \in VPN_{abs}$.

The objective P_v, refers to the average VPN Call Success Ratio for a given VPN v. The two objectives as stated as part of VPN-CS problem could be positively or negatively correlated depending on the solution of the VPN-CS problem**.** The positive correlation between the two objective functions can be explained as follows: Improved network utilization can be achieved by maximizing bandwidth included in the partitioning process, resulting in efficient bandwidth exposure to a VPN. This decreases the probability of a VPN terminating calls wrongly because of negative correlation between the exposed resources and availability of core network resources, thereby improving the Call Success Ratio of a VPN. The maximum network utilization and VPN Call Success Ratio could be negatively correlated when the heuristic solution is too pessimistic or too aggressive during the partitioning process. This could lead to either exposing less capacity than that is available or lead to over-subscribing the available resources. Hence solution to the VPN-CS problem has to strike the balance between the two objectives, which is to improve network utilization without impacting the VPN's Call Success Ratio. We begin our solution by focusing on objective of maximizing U_c. Considering the relationship between the maximization objectives of the VPN-CS problem and the total abstracted capacity, which is the aggregate of the link capacities of all the partition graphs $SG(V_k, E_k)$, we formulate the VPN-CS problem with the goal of maximizing the total abstracted capacity. In the formulation, let K_v represent the set of all source destination pairs of VPN v. The decision variables of the formulation are as follows. The variable $x_{(i,j)}^{k,v}$ denotes the partition resource assigned to source destination pair (s_k,d_k) of $v \in VPN_{abs}$ on an edge $(i,j) \in E$. Let the net flow achieved for each source-destination commodity k of VPN v be $f_{k,v}$. Using these definitions the VPN-CS problem can be formulated as in (1)-(5) with the objective of maximizing the aggregate resources considered as part of the partitioning process. In the formulation (1)–(3) enforces the supply demand condition for each commodity k for each VPN v. (4) enforces the capacity bound for each link of the graph. (5) Enforces the fairness constraint by forcing the net flow for every pair of commodities belonging to two different VPNs *v1* and *v2* having the same source destination pair to be equal. The worst case scenario of the formulation can be assessed assuming that all the VPNs are hosted on all the border nodes. In this case the number of variables is $O(|E|*|VPN_{abs}|*|B|^2)$, and the number of possible constraints would be in the order $O(|VPN_{abs}|^2*|B|^2)$**.**

$$MAX \quad \sum_{v \in VPN_{abs}} \sum_{k \in K_v} f_{k,v}$$

Subject To:

$$\sum_{(i,j)\in E} x_{(i,j)}^{k,v} - \sum_{(j,i)\in E} x_{(j,i)}^{k,v} = f_{k,v} \quad \forall v \in VPN_{abs},\ k \in K_v \ \ i = s_k \qquad (1)$$

$$\sum_{(i,j)\in E} x_{(i,j)}^{k,v} - \sum_{(j,i)\in E} x_{(j,i)}^{k,v} = -f_{k,v} \quad \forall v \in VPN_{abs},\ k \in K_v, i = d_k \qquad (2)$$

$$\sum_{(i,j)\in E} x_{(i,j)}^{k,v} - \sum_{(j,i)\in E} x_{(j,i)}^{k,v} = 0 \ \forall v \in VPN_{abs},\ k \in K_v, i \neq s_k i \neq d_k \qquad (3)$$

$$\sum_{v \in VPN_{abs}} \sum_{k} x_{(i,j)}^{k} \leq C(i,j) \quad (i,j) \in E \qquad (4)$$

$$f_{(k,v1)} - f_{(k,v2)} = 0 \quad \forall (v1,v2) \in VPN_{abs}, k \in (K_{v1} \cap K_{v2}) \qquad (5)$$

For an MDVS framework, where the goal is to enable a dynamic bandwidth request based on topology abstraction in smaller timescales, the linear programming formulation would be too complex to solve. Instead we propose algorithms based on a variant of multi-commodity flow problem which also enforces fairness, namely the Maximum Concurrent Flow problem which we modify to achieve the objective stated above, which also takes advantage of available approximation heuristics from the literature. The maximum concurrent flow based approach is discussed next.

3.1 Maximum Concurrent Flow (MConF) Approach

Given a network $G(V,E)$ and set K of source-destination pairs (s_i, d_i), i=1,2, …$|K|$. Also given the demands $D(s_i, d_i)$, i=1,2, …k, the objective of the MConF Problem is to maximize the value of 'Z' such that there exists a flow that satisfies the demands $Z^* D(s_i, d_i)$, $\forall$ i= 1, 2,…k . The node link formulation of the MConF problem is as follows. In the formulation (6)-(9) for a commodity $k \in K$, $x_{(i,j)}^{k}$ would represent the flow on link (i,j) due to flow between source-destination pair (s_k , d_k).

Maximum Concurrent Flow problem (**MConF**)

Objective: *Max Z*

Subject To:

$$\sum_{(i,j)\in E} x_{(i,j)}^{k} - \sum_{(j,i)\in E} x_{(j,i)}^{k} = Z*(-D(s_k,d_k)) \ \forall k \ i = s_k \qquad (6)$$

$$\sum_{(i,j)\in E} x_{(i,j)}^{k} - \sum_{(j,i)\in E} x_{(j,i)}^{k} = 0 \ \forall k \ i \neq (s_k,d_k) \qquad (7)$$

$$\sum_{(i,j)\in E} x_{(i,j)}^{k} - \sum_{(j,i)\in E} x_{(j,i)}^{k} = Z*(D(s_k,d_k)) \ \forall k \ i = d_k \qquad (8)$$

$$\sum_{k} x_{(i,j)}^{k} \leq C(i,j) \quad (i,j) \in E \qquad (9)$$

The value of Z is called the throughput of the concurrent flow. A good analysis of the MConF problem and optimality conditions has been discussed in references provided in [8]. In [8] authors propose a *Fully Polynomial Time Approximation Scheme* (FPTAS) for the problem. The VPN-CS problem differs from the MConF problem in a significant way. In the case of VPN-CS problem the commodity demands $D(s_k, d_k)$ are not known. To solve the VPN-CS problem we initialize $D(s_k,d_k)$ to $MaxF(k)$, where $MaxF(k)$ is the maximum flow for commodity k. The rationale for choosing the maximum flow capacity as the source-destination commodity demand is because of the objective of maximizing the resources considered during the generation of the partition graphs. In the context of VPN-CS problem, a source-destination pair (s_k, d_k) could appear in more than one VPN. But as noted in formulation (1)-(5), modeling the problem for each of the VPN source-destination demand pair would give rise to a commodity complexity of $O(|VPN_{abs}|^2*|B|^2)$**.** To reduce this complexity we propose here an aggregated approach. Here we define a commodity k as source destination pair (s_k,d_k), where the commodity flow represents the aggregate flow for all VPNs which includes source-destination pair (s_k,d_k), we represent this set of VPNs as $VPN_{abs}(s_k, d_k)$. This reduces the number of considered commodities to $O(|B|^2)$. With respect to the formulation (6)-(9) $x^k_{(i,j)}$, for edge $(i,j)\in E$, represents the flow between border nodes s_k and d_k.. With these considerations we next propose a centralized heuristic to solve VPN-CS problem and compute abstractions for the VPN's.

Multi-Concurrent flow heuristic for VPN-CS Problem
Input: $G(V,E)$, K, VPN_{abs}
Output: $SG(V_x, E_x), x \in VPN_{abs}$

Step 1: For each border node pair (s_k,d_k), $k \in K$, the supply for s_k is assigned to $UB_{max}(k)$ and demand at d_k is assigned to $-UB_{max}(k)$

Step 2: Solve the Maximum Single Concurrent Flow Problem.

Step 3: For each pair of border nodes (s_k,d_k) and identify set $VPN_{abs}(s_k,d_k)$, divide the path flow's among the VPN's common to the demand pair, i.e $\forall$ $v\in$ VPN(s_k,d_k), $x^{k,v}_{(i,j)} = x^k_{(i,j)} / |VPN_{abs}(s_k,d_k)|$. Here $x^{k,v}_{(i,j)}$ is the capacity assigned to VPN v on link (i,j) for VPN source-destination pair (s_k,d_k) .

Step 4: For all VPN $v \in$ VPN$_{abs}$, create subgraph SG(V_v, E_v), where V_v is the set of border nodes of VPN v and E_v is the set of all links that have nonzero $x^{k,v}_{(i,j)}$.

Step 5: For each VPN v, distribute SG(V_v, E_v) to PE$_v$.

Fig 3.1, Maximum Concurrent flow Based heuristic for VPN-CS

Fig. 3.1 shows the steps involved in MConF based heuristic for VPN-CS problem. In *Step 1*, for each border node pair (s_k ,d_k), the supply for s_k is set to $UB_{max}(k)$ and demand at d_k is set to $-UB_{max}(k)$ where $UB_{max}(k)$ is the value of a maximum flow from s_k to destination d_k With these initializations, *Step 2* solves the MConF problem and finds the throughput as well as the flow assignment for each commodity k on each link $(i,j) \in E$. Once the allocation for each source-destination pair is obtained, the task is to share the computed capacity for each source-destination commodity for each of the related VPNs. In *Step 3*, we solve this problem as follows. First we identify the set of VPNs, $VPN_{abs}(s_k ,d_k)$, which could have a potential path request from s_k to d_k.. We denote the capacity assigned for VPN $v\in VPN_{abs}(s_k,d_k)$ on link (i,j) as $x^{k,v}_{(i,j)}$ .To ensure fairness to all VPN commodities we divide computed partitioned resource $x^k_{(i,j)}$ equally among all VPN $v\in VPN_{abs}(s_k,d_k)$. We repeat this for each pair (s_k,d_k). *Step 4* uses the capacities logically partitioned for each of the VPNs to realize the VPN subgraph $SG(V_v, E_v)$ which is distributed to nodes PE_v. The nodes in PE_v would be used in *Step 3* as shown in Fig 2.1 to generate fully meshed abstracts graph $G_{abs(v)(FM)}(V_v,E_v)$. To generate $G_{abs(v)(FM)}(V_v,E_v)$ we can apply one of the heuristics presented in [7], the simplest of which is to apply the *Maximum Capacity Path* heuristic with complexity of $O(|V^2|)$. [8] Proposes the cheapest ε-approximation algorithm for MConF problem with a complexity of $O(\varepsilon^{-2}|E|*(|E|+|K|))$ which can be applied in *Step 2* of MConF heuristic. With respect to the complexity of the MConF heuristic which is obviously dominated by *Step 2* making the net complexity $O(\varepsilon^{-2}|E|*(|E|+|K|))$.

IV. Improving Network Utilization and Success Ratio for MConf Based Heuristic

The MConF based heuristic proposed for the VPN-CS problem solves the problem of aggregated source-destination commodity pairs by virtually partitioning the graph for each commodity. This approach could be conservative and result in a situation analogous to dedicated partitioning of resources for VPNs under known traffic conditions. One of the advantages of applying MConF heuristic is to minimize the conflict for resources among VPNs which is expected to be result in minimal crankback of VPN calls. But conservative nature of partitioning the available resources could also lead to poor performance in terms of VPNs' Call Success Ratio P_v metric due to high number of calls being rejected locally. This is because of exact partitioning of resources among VPNs as a result of applying MConF heuristic. Considering this, we propose two schemes by which the network utilization and VPN success ratio can both be improved while at the same aim to preserve the low call crankback property.

4.1 Over-subscription of Network links

We first pre-determine an over subscription factor f_{os} by which the residual capacity of all the network links of the core network would to be oversubscribed. The CS oversubscribes the link with the pre-determined f_{os} before applying the MConF heuristic. The factor which affects the performance with this proposal is the choice of f_{os}. Oversubscribing technique results in higher residual core capacity available for sharing among commodities. This increases the exposed capacity to the VPNs. An observation to be made here is that a very aggressive oversubscription of link residual capacities may improve the VPN success ratio, but could also lead to high number of crankback calls. We suggest choosing the

subscription factor through simulations such that the VPN crankback performance is within acceptable threshold.

4.2 MultiPath Flow Abstraction

The enhancement adopted here takes advantage of the fact that the partitions generated by MConF heuristic are non-overlapping and hence the VSP could apply an aggressive abstraction algorithm when abstracting capacity using the partition graph $SG(V_k,,E_k)$ in *Step 3* of the topology abstraction process shown in Fig 2.1. In the prior discussions we suggested using Maximum Capacity heuristic, proposed in [7] to compute capacity to the virtual links. Maximum Capacity heuristic uses the shortest widest path algorithm which associates the bottle neck capacity of a single path to the virtual link. This choice of algorithm may turn out to be very conservative as capacity more than the maximum single flow capacity may be available in the core network because of statistical multiplexing of requests among the VPNs. Considering this we propose an enhancement to the MConF based heuristic. In *Step 3* of Fig 2.1 where the abstract topology is generated using $SG(V_k,,E_k)$ we suggest using the maximum flow (max-flow) capacity algorithm [9] to compute virtual link capacity between a pair of border nodes for a given VPN. For a given set of demand requests of a VPN, abstraction derived from max-flow algorithm is expected to result in better VPN call performance than the virtual link capacity derived from the single path abstraction.

V. SIMULATION STUDY

A discrete event simulation was implemented using OpNet, on a random network topology of 25 nodes with an average node degree of 4. Five nodes of the core topology were chosen randomly as the PE nodes. Each of the PE nodes was configured to handle three different VPN instances. In addition the centralized server (CS) is also implemented which generates the VPN partitions for the border PE nodes. The core network routing and signaling were implemented to simulate an MPLS control plane. The signaling plane was implemented to simulate RSVP-TE behavior. The nodes periodically update their link state routing database by reading the current state of links during the simulation. Hence, from the simulations perspective the internal core LSDB associated with the core nodes (PE and P) is always synchronous with the link state of the core topology. In order to separate the context for each VPN, the border nodes are configured to have a VRF [5] for each VPN its serves. The abstract topology that is updated by the CS is stored in the VRFs that are periodically updated to the VPNs. The CS provides the border node a fully meshed abstraction for each of the VPNs it serves. All the VPNs for the sake analysis are assumed to subscribe to the Source Star (SR) abstraction [6]. The dynamic bandwidth call requests from the VPN client nodes are modeled as Poisson arrivals, with the capacity requested itself being uniformly distributed between [1-U], here U is the upper bound of the uniformly distributed demand. Similarly the call holding times are exponentially distributed. For the ε-approximation algorithm proposed in [8], we choose ε of 0.9 for all the cases considered in our simulations.

We define four performance metrics to measure the VPN's call performance and VSP's MDVS efficiency: *VPN Success Ratio*, *VPN CrankBack Ratio, VPN MissCall Ratio* and *Average Network Utilization*. We define the two new performance metrics *VPN CrankBack Ratio and VPN MissCall Ratio*. The *VPN CrankBack Ratio* is defined as the fraction of calls that have been cranked back because of a successful path computation at the VPN's source end, but rejected from the neighboring border or intermediate core node because of insufficient link capacity. The *VPN MissCall Ratio* is defined as ratio of calls that have been terminated locally by the CE node even if there was enough resource to accommodate the calls, to the total calls originated by the VPN. A good abstraction scheme would ideally have a miss call ratio of zero. In all the scenarios considered below the VPN's are assumed to be uniformly loading the network, and the results presented are the average of the performance behavior for all the three VPNs.

5.1 Performance of MConF Based Heuristic

Fig 5.1 compares the behavior of success and miscall ratio with varying network load. The load here refers to the ratio of mean holding times of calls to their mean inter-arrival times. With increasing load, we notice a decrease in the success ratio metric performance, while the miscall ratio metric increases. The VPN crankback ratio was observed to be almost zero in all the cases, showing the usefulness of applying a MConF heuristic for the VPN-CS problem. VPN success and misscall ratio are observed to be inversely related, as decreasing success ratio with no crankbacks indicates more number of calls being rejected locally by the VPN even though there is sufficient resource in the network core to satisfy the requests. The consequence of decreasing VPN success ratio is detrimental to the VSP since a conservative capacity exposure implies calls being rejected locally by the VPNs resulting in the loss of valuable revenue. This also contributes towards poor network utilization which is noticed in the graph for lower load conditions which is caused because of high miss call ratio. We next study the improvement of the performance metrics by applying the enhancements proposed in Section 4.1and 4.2**.**

5.2 Performance of MConF Based Heuristic with Enhancement

Fig 5.2 compares the VPN success ratio with increasing over-subscription factor. In this scenario we fix U and VPN load for all the different cases. From the graph we see that increase in over-subscription factor improves the VPN performance statistics. This can be attributed to increasing virtual capacity being exposed to the VPNs, thereby increasing the number of calls being processed successfully by the VPN and being admitted by the VSP. But we also expect crankback ratio to increase with large oversubscription factors, but this was not obvious with the simulation set up used to derive results in Fig 5.2. In another set up whose results are shown in Fig 5.3 the network load was increased by factoring U by 10. Here we

observed that the crankback ratio became significant for higher subscription factor. In these conditions we recommend the VSP to choose an appropriate oversubscription factor so as to bound the crankback ratio to less than a certain threshold which can be determined by appropriate simulation setup.

Fig 5.4 shows the VPN performance of applying the discussion presented in Section 4.2 where we use a multi-path flow abstraction of virtual capacity solving Maximum Flow problem during the abstraction process instead of a shortest widest path algorithm. We observed very good improvement in the performance applying the max-flow abstraction strategy. This can be attributed to two reasons: first because applying max-flow algorithm to abstract available resources from the partition graph increases the exposed virtual capacity. Secondly the mechanism also takes advantage of the statistical multiplexing of demands among the source-destination pairs of a VPN whose virtual capacities are derived from non-overlapping partition graphs determined by solving the VPN-CS problem. Comparing this with oversubscription method, we observe a consistent performance in terms of VPN success ratio for varying load conditions. But this may not be true all the time. With increasing user demand (i.e *U*) there may be a point beyond which the max-flow strategy may not improve the VPN success ratio. Also unlike the over-subscription enhancement max-flow abstraction method lacks a tunable parameter which can be used to control the VPN performance, but this flexibility also lends itself as problem for the VSP to choose the right oversubscription factor.

VI. CONCLUSIONS

This paper is part of an ongoing investigation of a novel dynamic managed VPN service first proposed in [6][7]. In this paper we studied the problem of capacity sharing among VPNs subscribing to Managed Dynamic VPN Service (MDVS) in a centralized context. The problem to be solved by the central server has been defined as the VPN Capacity Sharing problem. The paper proposes to solve the problem using Maximum Concurrent Flow (MConF) theory and propose two extensions to it. The scenario simulating MDVS and heuristic solutions demonstrates the usefulness of these solutions.

VII. REFERENCES

[1] Chun Tung Chou, "Traffic Engineering for MPLS-based Virtual Private Networks", IEEE, 2002

[2] Debasis Mitra, John A. Morrison, K.G.Ramakrishnan, "Virtual Private Neworks: Joint Resource Allocation and Routing Design", IEEE, 1999

[3] N.G.Duffield, P.Goyal, A.Greenberg, P.Mishra, "A flexible model for resource management in VPN", in Proc. ACM SIGGCOMM, 1998, pp 95-108

[4] Rebecca Issacs, " Light Weight Dynamic Programmable VPN", IEEE-OPENARCH, 2000.

[5] Eric Rosen et al,"BGP/MPLS IP VPN's", IETF, RFC 4364

[6] Ravi Ravindran, ChangCheng Huang, K.Thulasiraman, "Topology Aggregation as a VPN Service",IEEE, ICC 2005

[7] Ravi Ravindran, ChangCheng Huang, K.Thulasiraman, "A Dynamic Managed VPN Service: Architecture and Algorithms", IEEE, ICC 2006

[8] Lisa K. Fleischer, "Approximating Fractional Multicommodity Flow Independent of the Number of Commodities", IEEE, Foundations of Computer Science,1999

[9] Ravindra K.Ahuja, Thomas L. Magnanti, James B. Orlin, "Network Fows", Prentice-Hall, 1993

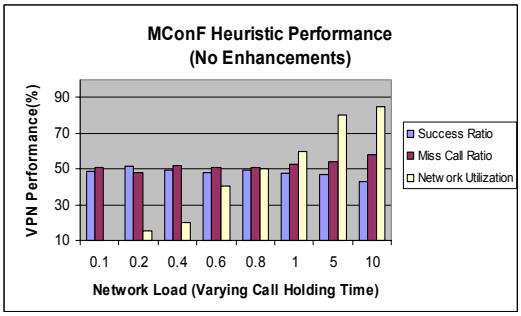

Fig. 5.1: Performance with varying Call Holding time

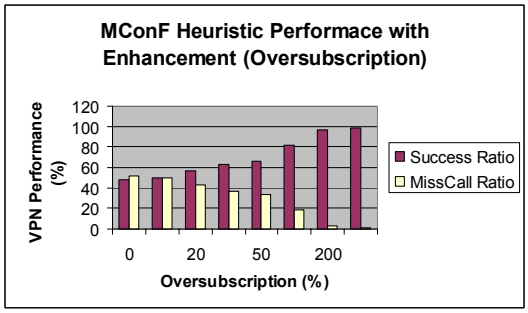

Fig. 5.2: Performance with varying Oversubscription factor

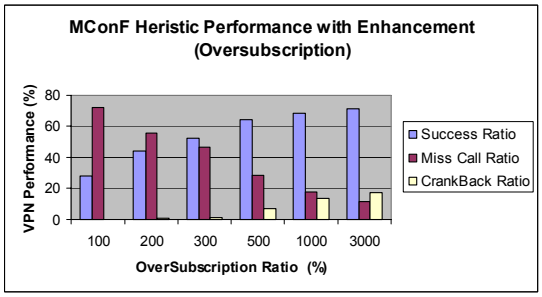

Fig. 5.3: Performance with varying Oversubscription factor

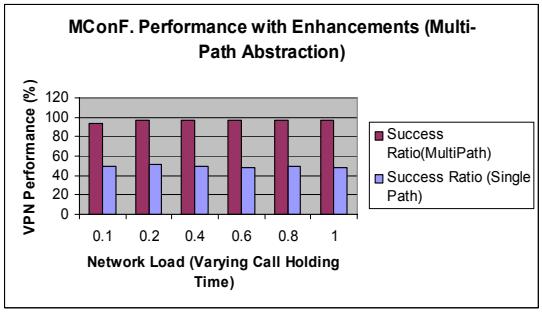

Fig.5.4: Performance with varying network load