A Hierarchical Distributed Protocol for MPLS Path Creation

Mohamed El-Darieby[1], Dorina Petriu[1], and Jerry Rolia[2]

[1]Systems and Computer Engineering Dept.
Carleton University
Ottawa, ON, K1S 5B6, Canada
{mdarieby, petriu}@sce.carleton.ca

[2]Internet Systems and Storage Lab.
Hewlett-Packard Labs
Palo Alto, CA, 94040, USA
jar@hpl.hp.com

Abstract

Network service provisioning involves the control of network resources through signaling, routing and management protocols that achieve Quality of Service and Traffic Engineering objectives. Network service providers are typically faced with scalability problems due to the explosion in network size and demand. In this paper we propose a new hierarchical distributed protocol (HDP) for the provisioning of MPLS-based tunnels. The protocol encapsulates the required signaling and routing functionality to share network resources among services. The protocol alleviates the scalability problem through extending the current two-tier architecture of the Internet to a multi-level hierarchical one. It enables mechanisms for Quality of Service, and intra- and inter-domain Traffic Engineering. Analytical analysis and simulation results show that the protocol exploits parallelism in routing computation to reduce the setup time of an MPLS path. This comes at the expense of an increased message complexity relative to other hierarchical protocols.

Keywords: Service provisioning, Virtual networks, MPLS, Traffic engineering, Hierarchical networks, PNNI.

1. Introduction

Computer networks have become ubiquitous with end users increasingly dependent on network services. Service Providers (SP) must now offer reliable and flexible network services with specific Quality of Service (QoS) levels to their users. As a result timely and efficient network service provisioning is a competitive differentiator among service providers.

Each SP controls the allocation/ deallocation of its network resources to different network services. This involves selecting the appropriate network elements to participate in the provisioning of the service and as a result is subject to QoS routing and signaling. Moreover, each SP also aims to optimize the utilization of its resources via Traffic Engineering (TE). The process of *Service Control* (or creation) is invoked in response to a user request for a service. Service control calls for QoS and TE signaling, routing and management protocols. A SP should also monitor the execution of the provisioned network services and should respond to network failures and dynamics. This is known as *Service management* and lasts for the duration of the service. It involves Fault, Configuration, Accounting, Performance, and Security (FCAPS) management.

MPLS promises to be a prominent technology for network service provisioning. MPLS tunnels, also referred to as Label Switched Paths (LSP), are used to traverse packet-switched networks, SONET networks, as well as optical networks. Tunnels help to enable QoS and TE by controlling routing and enabling resource allocation mechanisms [5].

QoS and TE introduce scalability problems. The growth of network size (i.e. number of network elements) results in an explosion in the number of routing and signaling messages required. TE requires the route of an MPLS path to be centrally computed either at the ingress point to the network or at a central management node (e.g. a Bandwidth Broker). This implies that the *computing node* has complete routing information about the entire network. For this to happen each network node advertises its state frequently to the computing nodes in the network; which is a scalability issue for large networks. In [12, 13], we evaluated the performance of a

Multi-Protocol Label Switching (MPLS)-based service provisioning architecture as the underlying physical network grows in size. We answered the following questions: a) how do increases in the size of the physical network affect the service creation performance under different loads, and b) how sensitive is the service creation performance to various architectural parameters. These parameters include the number of Autonomous Systems (AS) and the average number of routers within an AS. Detailed performance results showed that the CPU of the edge router was the system bottleneck because it calculates centrally the explicit route for the path.

Goals of this paper

In this paper, we propose a new hierarchical protocol for the provisioning of MPLS-based paths. The protocol encapsulates the required signaling and routing protocols. The protocol scales up to very large networks extending the current two-tier architecture of the Internet [1] to a multi-level one that resembles PNNI-based networks [10, 11]. The protocol exploits parallelism in routing computations. Analytical analysis and simulation results show that the proposed protocol reduces the setup time of an MPLS path. This comes at the expense of an increased number of messages, as discussed in section 4.

The protocol is designed to suit the current requirements of the Internet architecture. It controls mechanisms for QoS, intra- and inter-domain TE [5] which leads to better utilization the of network resources. The protocol responds to changes in network topology dynamics (e.g. failures) and to changes and fluctuations in workload. This results in provisioning MPLS paths that better meet the requirements of service consumers and providers.

We focus on service (MPLS path) creation but do discuss relevant interactions between creation and management systems. Section 2 describes the protocol, and the deployment of inter-domain TE. In the following section, we relate our protocol to other research and standards. Then, in section 4, we analyze the

overhead of our protocol and compare it to flat and hierarchical protocols. Conclusions are offered in the last section.

2. Hierarchical Networks

In this section, we describe the new Hierarchical Distributed Protocol (HDP). An MPLS path creation scenario employing the protocol is then described. Path creation involves the exchange of signaling messages, making routing decisions and allocating network resources. We also discuss how the protocol enables TE.

A hierarchical network is a traditional solution to the scaling problem [1, 10, 11]. Nodes are organized into different interconnected sub-networks called domains. A domain consists of a number of interconnected network nodes (i.e. IP routers or ATM switches). A managing node controls a domain. A managing node maintains topological and state information about the domain and its nodes and links. In the Internet literature, a domain is called an Autonomous System (AS) and the managing node is called a Bandwidth Broker (BB). In the ATM (PNNI) literature, a domain is known as a Peer Group and the managing node is known as a Logical Group Node. In this paper we use the Internet terminology. We will also assume that a BB is a server node that is separate from the physical nodes of the AS. Server nodes are not routers or switches. They are cluster based server farms that can grow in capacity based on the intensity of path creation requests.

The physical nodes of an AS belong to level-0 while the BB's represent level-1 nodes of the network hierarchy. At level-1, the BB's are interconnected in a manner that corresponds to that of the level-0 Inter-AS connectivity. Level-1 BB's are grouped into virtual level-1 AS's. Each virtual level-1 AS is abstracted, represented and managed in turn by a single level-2 BB. The level-2 BB's maintain topology and state information about the virtual level-1 AS's. Level-2 BB's are interconnected in a manner that corresponds to the interconnectivity of level-1 AS's. They are assumed to be separate servers as well.

The process of grouping BB's into logical AS's and abstracting such AS's into BB's (at the next higher level) continues till a rooted hierarchy of BB's is constructed. This hierarchy is assumed to be static. Fault-tolerant

signaling channels between different (same-level and child-parent) BB's are also assumed. Internet standards (e.g. OSPF) propose a two-level hierarchy [1]. The ATM PNNI standard supports a hierarchy of up to 105 levels [10].

An example of a networking hierarchy is depicted in Figure 1. Physical (level-1) nodes and links are grouped into 10 different AS's. For example, nodes 2.1.1, 2.1.2, and 2.1.3 are grouped together into AS (2.1). The 10 level-1 AS's are managed by 10 level-2 BB'S. These 10 BB's are grouped into 3 virtual level-2 AS's (AS's 2, 3, and 4). Each of the level-2 AS's is managed by a level-3 BB. These are BB's 2, 3, and 4 that are grouped into a single level-3 AS. The root of the BB hierarchy is at level-4 that manages the level-3 AS.

The Hierarchical Distributed Protocol (HDP)

HDP is executed in response to a request received by a physical node to create an MPLS path. The request specifies the required connectivity among a source and destination pair to be connected. The bandwidth (or in general the QoS) requirements of the path are specified in the request.

The node records the request information and sends a *Find_root* message to its parent BB in search for a BB that has a view of the two endpoints of the path. The parent BB will check the nodes of the AS it is managing against the endpoints of the path. If the BB finds the destination endpoint to be outside of its view, it consults its parent BB. The process is repeated up the hierarchy, till we reach a BB that has a view of the source and destination points. This BB is the root of that path management hierarchy and we will assume that it is at level (L+1).

For example, in Figure 1, node 4.3.4 receives the request for an LSP that has 2.2.1 as its destination endpoint. It sends a *Find-Root* message to its BB (node 4.3). The BB finds the other endpoint to be outside its view. It sends *Find_Root* message to its BB (node 4) that finds a similar result. So, it consults its parent BB (node BB*) that finds itself able to view both path endpoints.

The next step is to calculate the explicit routes connecting the endpoints. The routing calculations are done at each level of the network hierarchy. The root BB at level L+1 initiates it. It calculates an abstract route for the path at level L. The root node, then, informs the BB's along the calculated route by sending *Notify* messages to them. Each of the level-L BB's starts calculating in parallel a route within the managed logical AS at level L-1.

To achieve inter-AS TE, the routing process can be controlled to avoid highly utilized AS's. Standard topology aggregation schemes [15] may be used to express the utilization of a network AS. The output of these mechanisms and the state maintained about inter-AS links can be used by the routing process to route the path through less-utilized AS's and to avoid highly utilized ones. For example, in Figure 1, in spite of the longer hop count along the preferred route, the highly utilized AS 3.2 was avoided and AS 3.1 was preferred. Moreover, the central calculation of the explicit route [5] enables intra-AS TE.

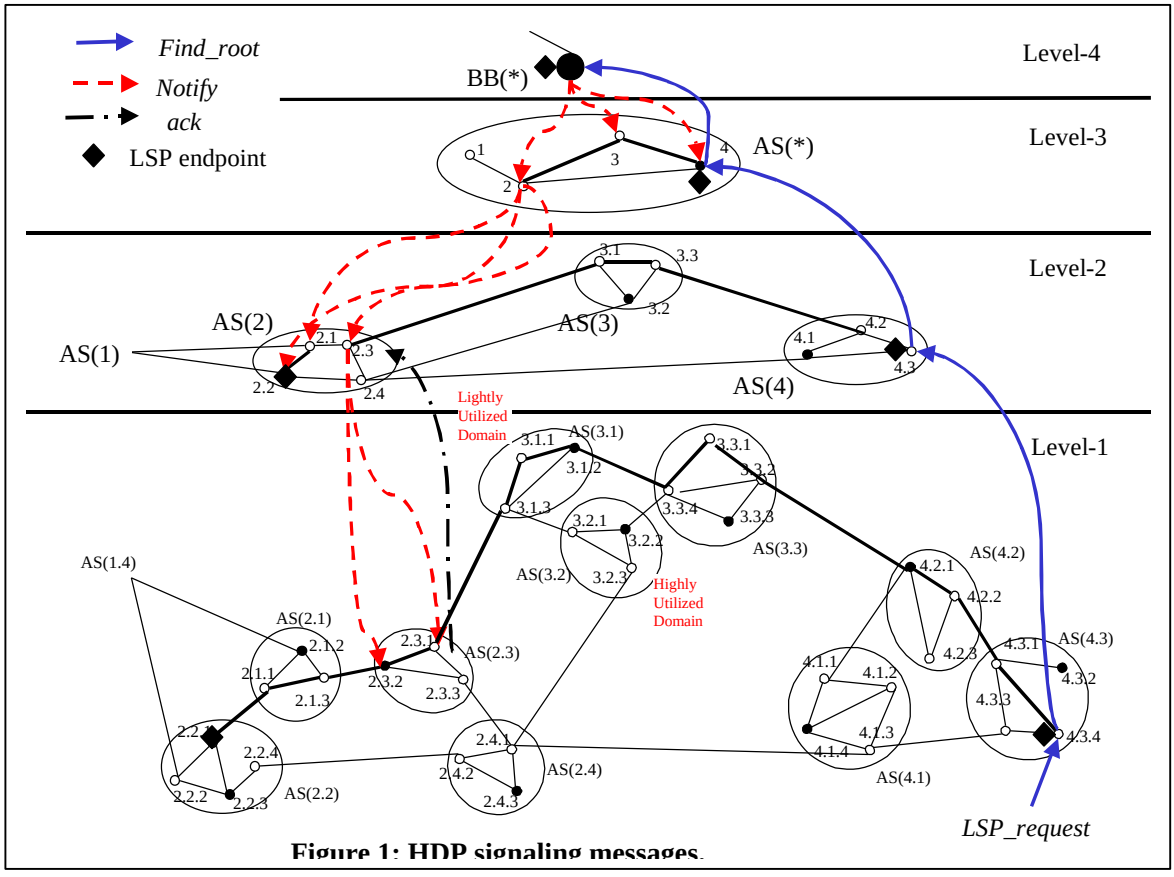

Figure 1: HDP signaling messages.

Each of the level-L BB's maintains state information about its corresponding level L-1 AS. A level-L BB calculates a route within its level L-1 AS, then, it sends *Notify* messages to the level L-1 nodes along the *calculated* route. Notification messages bear information regarding path requirements. To calculate an intra-AS route, a BB has to select the ingress and egress nodes for that route. The selection can be based on policies such as using the widest inter-AS link.

To decrease the routing overhead and setup time, intra-AS routes are calculated in parallel. Each of the BB's (say at level N) initiates the calculation of a route within the boundaries of its managed level (N-1) AS. The protocol relies on standard mechanisms for intra-AS QoS routing [7] to calculate intra-AS routes.

For example, in Figure 1, the root BB calculates a route at level-3 nodes that traverses nodes 4, 3, and 2. The root BB will notify the three nodes with its decision. Node 3 maintains information about AS (3) and the links connecting it to its neighboring AS's (i.e. AS (4) and AS (2)). We show only the notification message sent to node 2 in the figure for the sake of clarity. Once node 2 receives the notification from the root, it starts choosing the ingress and egress points to the AS it is managing. This is done using the information maintained by node 2 maintains and assumes the information is up to date. If the information is not adequate a future crankback may result. Node 2 may have to negotiate with node 3 on the choice of the ingress point to its AS. Then Node 2 calculates a route within its AS and notifies the corresponding BB's. Assume this route traverses nodes 2.3, 2.1, and 2.2. Each of these three BB's applies, in turn, the same algorithm. The figure shows the notification messages sent from BB 2.3 to physical nodes 2.3.1 and 2.3.2.

The above procedure is repeated at all levels of the hierarchy until we reach the physical nodes residing at level-1 of the hierarchy. Notification messages are sent by level-2 BB's to all physical nodes along the calculated routes. The physical nodes try *in parallel* to allocate their required network resources. If enough resources exist, the nodes admit the request and positively acknowledge its managing BB. Each physical node will send an *ack* message. The messages travel up the hierarchy following the reverse of the route the notification messages followed. As the root node receives an *ack* message for that path, it reports the success

of path creation to the node that generated the MPLS path creation request. This completes the process of creating the path. Figure 2 shows the algorithm for this protocol.

In case of a lack of network resources, the protocol uses a *cranckback* message to report these conditions. The message is reported to BB's at higher levels in the hierarchy. It causes these BB's to re-apply their routing algorithms for calculating tentative alternative routes.

The state information maintained at different nodes of the hierarchy must be updated. The physical nodes start sending *Update* messages up the hierarchy updating the state of each BB. In other words, each BB updates the information about the AS it's managing and about the incoming/outgoing links of that AS. The update messages travel all up the hierarchy, in successively aggregated form, until they reach the root BB.

2.3 Path Management

Service management functionality is deployed after the service has been created. Standard service management functions are: Fault, Configuration, Accounting, Performance, and Security (FCAPS). To implement these functions, information about the topology of the path created must be maintained at the managing nodes. The HDP enables the implementation of these functions. The BB's that participate in the MPLS path creation maintain the layout and topology of that path.

For example, monitoring the performance of the path can be done through monitoring the collective and individual performance of the different segments of the path in each AS. This can be done be probing the state of each path segment. This probing information can be accumulated, abstracted, and maintained at higher levels of the management hierarchy. This results in reducing the amount of management information maintained at the physical nodes that carry data traffic.

```
Processing @ level-N node Ai(n): {
—While(true){
—Wait for a message;
—If (Ai(n) is not dst_node) error—ignore;
Switch (received message){
        Case LSP_request_message (dst_node, src_node, LSP_request){
                —If (Ai(n) has processed a LSP_request for the same LSP before) error-ignore;
                —Else{ —Record LSP_request information;
                        —Send (Find_root, parent_BB, LSP_request, roue_to_root);
                    }
         }
        Case Find_root (dst_node, LSP_request, roue_to_root) {
                —Record LSP information;
                —If (at least one LSP_endpoint is not under jurisdiciton of Ai(n))
                        —Send (Find_root, parent_BB, LSP request, route_to_root);
                —Else DoRoute                   // we are @ the root BB
        }
        Case Notify {dst_node, src_node, LSP_request, route_to_root, calculated_route){
                —If (Ai(n) is the root of managing hierarchy) error-ignore;
                —If @ a physical node{
                        —Try allocating resources;    // Vertical Signaling
                        —If (failed) Send (Crankback, parent BB, Ai(n) , failure_code);
                        —Else {—Update local resource tables;
                                —Send (ack, parentBB, other info);
                             }
                    }
                —If (Ai(n) is-an intermediate BB node)-- DoRoute
        }
        Case ack (dst_node, amount_of_resources_allocated){
                —If (Ai(n) is a physical node) error-ignore;
                —Update local state information for that domain;
                —If (Ai(n) has not received all acks) wait;
                —Else {
                    —If (Ai(n) is an intermediate managing BB node)
                        —Send (ack, parent_BB, amount_of_resources_allocated_for that_LSP);
                        —ElseIf (Ai(n) is the root of the LSP managing hierarchy){
                        —Send (ack, parent_BB, amount_of_resources_allocated_for that_LSP);
                        —Notify source node of the creation of the LSP so as to notify the requesting node.
                    }
        }
        Case Crankback(dst_node, source_node, failure_code)){
                —If (Ai(n) is a physical node) error-ignore;
                -- DoRoute
        }
} /* end switch*/ } /* end while*/ }/* end method*/
method: DoRoute {
        —Calculate an explicit route within that domain connecting ingress and egress;
        —If (no route exist) —Send (Crankback, parent_BB, Ai(n),code);
        —Else {     —Record information about the nodes along the calculated_route
                    —For (all nodes, Aj(n-1), along the calculated_route)
                        —Send (Notify, Aj(n-1), Ai(n), LSP_request, route_to_root, calculated_route);
        }
}
```

Figure 2: Algorithm for the HDP protocol

3. Related Work

The Internet has a two-level routing architecture where individual administrative AS's independently make their own routing decisions based on providers' policies [1]. This decomposition allows each AS to deploy its own strategies. The architecture incorporates different protocols for inter- and intra- AS resource management and QoS support. A BB acts as a resource manager for an AS. BB's communicate with each other to establish inter-AS routes.

The Clearing House architecture [4] proposes organizing the Internet into a multi-level hierarchy of clearing houses (another name for a BB). Each clearing house manages a single physical or logical domain of the hierarchy. The architecture performs in-advance inter-AS resource reservations of network resources. This pro-actively sets up paths (i.e. prior to the creation request arrival). The performance of such an approach is based, mainly, on the characteristics of the algorithm for predicting future traffic. Instead, we consider an *online* (i.e. reactive to request arrivals) approach to creating tunnels.

The ATM PNNI (Private Network-to-Network Interface) [10, 11] is a hierarchical routing protocol that scales to very large networks. Logical nodes connected with logical links exist at each level of the PNNI hierarchical model of the network. Nodes at a specific level are grouped into a number of Peer Groups (PG). A single node, called Peer Group Leader (PGL), represents each PG. A PGL represents its PG in a parent PG. A PGL is elected among the nodes of the PG by executing a leader election algorithm.

PNNI follows what may be called leaf-controlled routing where a physical node would do the routing calculations. It depends on the topological information it maintains about its PG and about the hierarchy to calculate a route. This information enables source nodes to have enough knowledge about the whole network to calculate a route to any destination node. It uses this information to calculate a route within its PG and to calculate a PG-level route to the destination.

Having done this route calculation, this node will forward a signaling message along the route to the ingress point of the next PG along the calculated route. This ingress point will have to do another route calculation for the route within its PG. The process continues till we reach the destination node.

This is different from the proposed HDP. We do not assume that physical nodes will maintain information about the hierarchy. They just need to know about their BB, and the BB's maintain information about the hierarchy. The route calculations we propose are done in parallel (as opposed to the in-series route calculation of PNNI). This reduces the setup time of a MPLS path, but comes at the expense of an increased number of signaling messages required for our protocol.

The *viewserver framework* [3] is a network hierarchy that provides a physical node with enough information to calculate the route all the way to the destination (even if the destination is outside of the node's AS). The protocol to achieve this involves going up and down the hierarchy. The going up part (similar to ours) has the objective of finding a parent "view server" (the root BB in our case) that controls both the source and destination nodes of the requested connection. The information about this AS and other relevant AS's are accumulated and forwarded down the hierarchy toward the source node. The accumulated view follows the same exact path followed by the signaling messages to find the root of the hierarchy. Along each stop of the path, more detailed information about network AS's is accumulated. The source node, now having appropriate information, *centrally* calculates the route to the destination node.

This is different from what we are proposing. In HDP, a BB calculates a route only within its AS. Different BB's for different potential AS's calculate the route in parallel. These different calculations are guided by information provided from nodes at a higher level in the hierarchy. The information provides the BB with only the ingress and egress routers of its AS. BB's coordinate the route calculation among themselves. This is opposed to providing a single node with the complete set of information required to centrally calculate a route. HRDP distributes the computational load between many BB's. This comes at the expense of an increased number of the signaling messages in HDP. However by off-loading the processing of message handling to

server farms we decouple path creation workload intensity from processing power requirements on routers and switches.

4. Evaluation

In this section we evaluate the performance of HDP. We study the number of messages and the computational complexity of the algorithm. We first analyze the complexity of HDP analytically and compare it to that of other hierarchical and flat protocols. Then we show the results of simulation experiments that we conducted to verify our analytical analysis.

We assume a hierarchy of (L+1) uniform levels (including the root BB). For a uniform hierarchy we assume the average number of nodes in a physical AS and of BB's in a logical AS to be *m*. We call *m* the network fan-out factor. Hence, the average number of nodes at level *N* would be m^N. We also assume the MPLS path would traverse *d* nodes (BB's) in an AS. We call *d* the path fan-out factor. So, the length of the LSP, *l*, at level *N* would be d^N. According to [14], the number of edges, *E*, in a N-node domain can be estimated by $E = \frac{m^L}{2(R+1)}\left(1-\left(\frac{1}{m^{L.R}+1}\right)\right)$. We take R = –0.80, which is the average for the values of R given in [14].

With *m* equal to the number of nodes in a domain, the number of edges is given by $E = 2.5*m^{0.8} * (m^{0.2}-1)$.

To establish a path, the root BB would have to notify *d* BB's at level-L. Hence, *d* messages will be sent from the root BB to level-L nodes. Each of the level-L BB's would send *d* notification messages to the nodes that will carry the data of the path in the (L-1) level AS. Hence, d^2 notification messages will be sent from level-L to level L-1. To reach the physical nodes (at level-1), the total number of notification messages would be $O\left(\sum_{i=1}^{i=L} d^i\right)$ messages. This is equal to the number of BB's/ nodes that will process these messages (in parallel).

Deploying a two-pass signaling protocol, the number of acknowledgment messages will be the same as the

number of notification messages. Hence the total number of messages of a two pass HDP would be $O\left(2.\sum_{i=1}^{i=L} d^i\right)$.

The setup time for the complete path would be equal to that of calculating intra-AS routes within BB's and for signaling messages exchange. Each BB applies an intra-AS routing algorithm to select the nodes that the path traverses. We assume each BB deploys a shortest path algorithm (e.g. Dijkstra's) that has a complexity of O(E.log m) [7]. Since this calculation is done at each level in parallel in each BB, it follows that the routing time at each level would be O(E.log m) as well. For L levels of BB's in the hierarchy, the route calculation time would be O(L.E.log m) and is performed within server farms.

To compare our algorithm to PNNI, we assume the same hierarchy dimensions. In PNNI, to setup a virtual circuit of the same length as our MPLS path of length d^L, all the nodes along the path will be involved in exchanging signaling messages [10, 11]. For the two-pass signaling protocol case, the total number of messages is of order O ($2.d^L$).

In PNNI, each edge router of an AS (Peer Group) would do the intra-domain routing calculations. For a virtual circuit of that length, the virtual circuit would traverse d^{L-1} AS's. Hence, the total routing time would be O (d^{L-1}.E.log m). It is obvious that our algorithm has a smaller routing computations time than PNNI. This is mainly due to the parallelism in processing of signaling messages and of route calculations. As mentioned before, this comes at the expense of an increased number of messages for our protocol.

To compare HDP with flat routing algorithms, we consider the algorithm presented in [6, 9]. This algorithm requires no routing computation done at any node. The message complexity is of O ($E+2.d^L$) where E, d, and L are defined as above. The setup time complexity of this algorithm is O ($2.d^L$). This algorithm has a lighter computational load than HDP and PNNI, but comes at a higher message complexity than both hierarchical algorithms.

We study the effect of the number of levels in the hierarchy on message and computational complexities for both HDP and PNNI. For the sake of this comparison, we consider a network of fixed size of 4^8 nodes and a path of a fixed length of 256 nodes. We consider different hierarchies H_1, H_2, H_3 and H_4, having (L+1) equal to 2, 3 5 and 9, respectively. The H_1 hierarchy (included just for the sake of comparison) has all its nodes arranged in a single physical system. The H_2 hierarchy resembles the current architecture of the Internet. It has its physical nodes arranged in 64 AS's each having 64 nodes. While H_3 is a hierarchy with one more level than that of the current Internet, H_4 has two more levels. As the number of levels increases the number of nodes m per AS decreases. The four hierarchies have the parameters and performance shown in Table 1.

	H_1	H_2	H_3	H_4
$L+1$	2	3	5	9
m	4^8	64	16	4
$E = 2.5*m^{0.8}*(m^{0.2}-1)$	146011	428	17	3
d	256	16	4	2
HDP Message complexity $O\left(2.\sum_{i=1}^{i=L} d^i\right)$	513	547	685	1020
PNNI Message complexity $O(2.d^L) = O(2.l)$	~2*256=512	~2*256=512	~2*256=512	~2*256=512
HDP Computational complexity O(L. E. log m)	~146011*4.8= 700853	~2*428*1.8= 1541	~4*17*1.2= 82	~8*3*0.6= 14
PNNI Computational complexity $O(d^{L-1}.E.\log m)$	~256*146011*4.8= 179418317	~256*428*1.8= 197222	~256*17*1.2= 5222	~256*3*0.6= 461

Table 1: Message and computational complexity for the hierarchical protocols

Table 1 gives the approximate number of messages exchanged to establish a path (message complexity) and a measure of computation required to select a shortest path (computational complexity). HDP causes more messages than PNNI, but has significantly lower time to establish the path.

We used the OMNet++ discreet event simulator [16] to develop simulation models for HDP. We used the models to determine the number of exchanged signaling messages and the setup time for different topologies of the network and for different hierarchy structures. We assumed OC3 links (i.e. full duplex 155 Mbps). The background traffic in each link is uniformly distributed between 50 Kbps and 120 Kbps. The path creation requests arrive at the physical nodes of the hierarchy as per a Poisson distribution. 500 requests are generated during each simulation run. We varied the size of the physical network and the length of the path to study the effect of different parameters on the performance of HDP.

Figure 3 shows the number of simulated HDP signaling messages exchanged for hierarchies with different depths and for different path lengths. For the same path fan-out, the depth of the hierarchy has a dramatic effect on the number of messages exchanged. For example, for *d=3,* the number of messages grows from 27 to 245 to 732 as the depth of the hierarchy increases from 3 to 5 to 6. Moreover, for a hierarchy of L+1 = 3, and a path fan-out factor of 2, 3, 4, 5, and 6, the number of messages is 15, 27, 43, 63, and 87, respectively. This corresponds to the creation of paths of length 4, 9, 16, 25, and 36 nodes, respectively. For a hierarchy of L+1 =5, and a path fan-out factor of 2, 3, and 4, the number of messages is 65, 245 and 685, respectively. This corresponds to the creation of paths of length 16, 81 and 256 nodes, respectively. The simulation model gives us the exact number of messages exchanged, which is identical to the analytical analysis presented above.

We simulated path setup time as the depth of the hierarchy, *L+1,* and the physical network size change for a path of 256 nodes of length. The results are shown in Figure 4. We changed the depth of the hierarchy from 3, to 5, to 9 levels. For a hierarchy of L+1=3, the setup time increases from 260 to 1020 to 4020 to 10020 (ms) as the network size changes from 2^8, to 3^8, to 4^8 and 5^8 nodes. Figure 4 shows the setup time for the hierarchies H_2, H_3 and H_4 described above. It is clear that, for a network of a given size and a path of a fixed

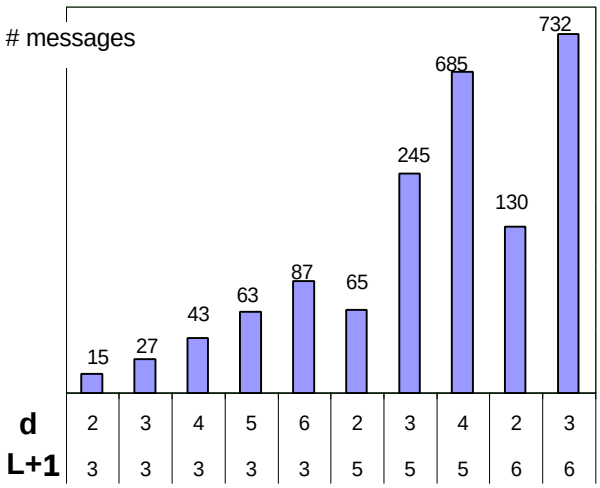

Figure 3:Number of setup messages for different paths

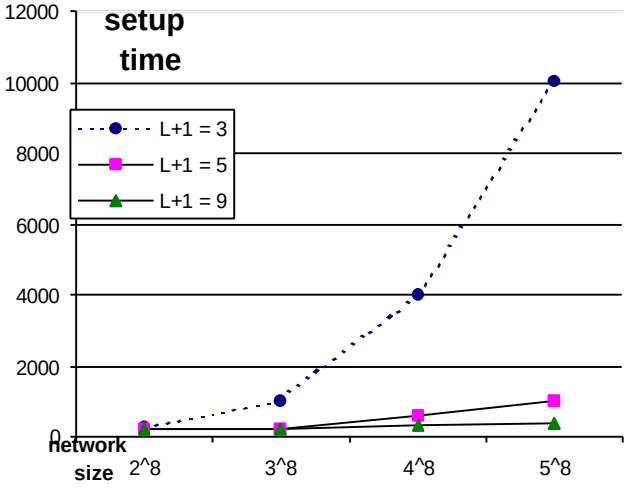

Figure 4: Setup times for different hierarchical structures

length, a hierarchy with more levels ensures a considerably smaller setup time than a hierarchy with fewer levels, in spite of the fact that the number of signaling messages grows with the number of levels. This is due to two factors: a) at each level, the routing computations for various AS's are done in parallel, and b) the number of nodes *m* per AS goes down for a hierarchy with more levels, for the same size of the physical network. The results are consistent with the analytical expressions presented above.

5. Conclusion

We presented a novel Hierarchical Distributed Protocol (HDP) for the creation of MPLS paths. HDP supports routing and resource reservation functions required to achieve QoS and Traffic Engineering objectives. It addresses the disadvantages of the current flat architectures to meet such objectives. It scales up to very large networks extending the current two-tier architecture of the Internet to a multi-level one. It also enhances existing hierarchical protocols (i.e. PNNI) by exploiting parallelism in performing routing computations within different network domains.

The disadvantages of HDP are that it requires each BB to calculate centrally the route within its managed domain. It also assumes that the results of (offline) negotiation among neighbor BBs on inter-domain paths are

available at the time of the computation. This requires the BB's to have a fairly updated view of the state of the domains they are managing. The BB computation executes on server nodes that are expected to be clusters of commodity servers in server farms, not on routers or switches.

We showed with analytical analysis and simulation results that HDP reduces the setup time of an MPLS path creation. This comes at the expense of an increased number of signaling messages compared to PNNI.

We focused on service (MPLS path) creation aspects and we studied its performance. We are planning on doing more in depth assessments of the HDP performance and the impact of update messages and the impact of using some pre-computed paths to decrease the numbers of messages exchanged. We also intend to focus on management aspects of service provisioning. We will concentrate on scenarios for the dynamics (e.g. failures) and reconfiguration of network topologies.

References:

1 A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A Two-Tier Resource Management Model for the Internet," in proceedings of Global Internet 99, Dec 1999. Available at http://www.cs.ucla.edu/~terzis/

2 B. Awerbuch, Y. Du, and Y. Shavitt, " The Effect of Network Hierarchy Structure on Performance of ATM PNNI Hierarchical Routing," Computer Communications Journal, vol 23, pp 980-986, 2000.

3 C. Alaettinoglu, and A. U. Shankar, "The Viewserver Hierarchy for Interdomain Routing: Protocols and Evaluation," IEEE Journal on Selected Areas in Communications, Vol. 13 (8), pp. 1396-1410, October 1995.

4 C. Chuah, L. Subramanian, R. H. Katz and A. D. Joseph, "QoS Provisioning Using A Clearing House Architecture", International Workshop on Quality of Service (IWQoS), Pittsburgh, PA, June 5-7, 2000.

5 D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, X. Xiao, "Overview and Principles of Internet Traffic Engineering," IETF INTERNET DRAFT, August 2001. Available at http://www.awduche.com/

6 V. Sarangan, D. Ghosh and R. Acharya, " Distributed QoS Routing for Multimedia Traffic," In proceedings of IEEE GlobeComm 2000, San Francisco, CA, USA, Nov 2000.

7. G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, and S. Tripathi, ``Intra-Domain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis," Special Issue of IEEE Networks on Integrated and Differentiated Services for the Internet, September 1999.

8 D. Reeves, and H. Salama, “A Distributed Algorithm for Delay-Constrained Unicast Routing,” IEEE/ACM Transactions on Networking, vol (8), No (2), pp 239-250, 2000.

9 S. Chen, and K. Nahrstedt, “Distributed QoS Routing”, Technical Report UIUCDCS-R-97-2017, Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA, 1997.

10. R. Cohen, R. Emek and E. Felstine, "Framework for Multicast in Hierarchical Networks," In proceedingso of IEEE InfoCom'2000, Tel-Aviv, March 2000.

11. E. Felstine and R. Cohen, "On the Distribution of Routing Computation in Hierarchical ATM Networks," In IEEE Transactions on Networking, Vol 7. No. 6, December 1999.

12. M. El-Darieby, and J. Rolia; "Performance Modeling of a Service Provisioning Design", in “Lecture Notes in Computer Science Vol.1890, Springer Verlag, 2000.

13. M. El-Darieby, D. Petriu, and J. Rolia; "Scalability Analysis of Virtual Network-Based Service Provisioning," In IFIP/IEEE Integrated network Management (IM 2001), Seattle, WA, USA, May 2001.

14. M. Faloutsos, P. Faloutsos, C. Faloutsos, “On Power-Law Relationships of the Internet Topology” In proceedings of ACM SIGCOMM’99, Boston, MA, USA, 1999.

15. B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt. ”Routing Through Networks with Hierarchical Topology Aggregation," *Journal of High Speed Networks*, vol. 7(1), 1998.

16. OMNET++ simulator, available at: http://www.hit.bme.hu/phd/vargaa/omnetpp.htm