QRON: QoS-Aware Routing in Overlay Networks

Zhi Li, *Student Member, IEEE,* and Prasant Mohapatra, *Senior Member, IEEE*

***Abstract*—Recently, many overlay applications have emerged in the Internet, such as peer-to-peer file sharing, end host multicasting, and content distribution network. Currently, each of these applications requires their proprietary functionality support, such as network topology discovery, routing path selection, fault detection and tolerance, etc. A general unified framework may be a desirable alternative to application-specific overlays. In this paper, we introduce the concept of overlay brokers (OBs). We assume that each autonomous system in the Internet has one or more OBs. These OBs cooperate with each other to form an overlay service network (OSN) and provide overlay service support for overlay applications, such as resource allocation and negotiation, overlay routing, topology discovery, and other functionalities. The scope of our effort is the support of quality-of-service (QoS) in overlay networks. In this paper, our primary focus is on the design of QoS-aware routing protocols for overlay networks (QRONs). The goal of QRON is to find a QoS-satisfied overlay path, while trying to balance the overlay traffic among the OBs and the overlay links in the OSN. A subset of OBs, connected by the overlay paths, can form an application specific overlay network for an overlay application. The proposed QRON algorithm adopts a hierarchical methodology that enhances its scalability. Two different types of path selection algorithms are analyzed in our paper. We have simulated the protocols based on the transit-stub topologies produced by GT-ITM. Simulation results have shown that the proposed algorithms perform well in providing QoS-aware overlay routing service.**

***Index Terms*—Modified shortest distance path (MSDP), overlay brokers (OBs), overlay routing, overlay service network (OSN), proportional bandwidth shortest path (PBSP), quality-of-service (QoS)-aware routing in overlay networks (QRONs), QoS satisfaction ratio.**

I. Introduction

A SIGNIFICANT amount of effort has been dedicated to incorporate quality-of-service (QoS) in the current best-effort service model of Internet. Although there have been advances in the solution proposals for network-level QoS provisioning, the models such as Intserv [15] and DiffServ [9] are far from being deployed in the Internet. The reluctance to the adoption of these models is tied to the changes that are required in the networking infrastructure. An alternative approach for QoS provisioning for QoS-sensitive applications is to provide support mechanisms for services in the application layer while retaining the best-effort network layer. *Overlay networks* have emerged as an effective way to support new applications, as well as protocols without any changes in the underlying network layer. Thus, it can provide flexible, scalable, and reliable Internet services [4]. For example, Qbone [21] and Mbone [11] utilize overlay techniques to support QoS and multicast services, respectively, on top of the current Internet infrastructure.

An overlay network is formed by a subset of underlying physical nodes. The connections between the overlay nodes are provided by overlay links, each of which is usually composed of one or more physical links. As the overlay applications are usually built at the applications layer, it can effectively use the Internet as a lower level infrastructure to provide high level services to end users.

Some of the overlay applications rely only on the end hosts [8], [13]. However, because of the limited access bandwidth and unpredictable connectivity of many Internet hosts, it would be desirable for the overlay applications to have some preset fixed nodes that support the overlay services [6], [8], [16]. The group of these preset nodes needs to detect dynamic network status, discover overlay topology, guarantee QoS, and provide fault tolerance. With the increasing number of overlay applications, more and more of such nodes will be required across the Internet. Although a few papers have discussed such concepts of overlay networks, most of the proposed overlay routing protocols are designed for specific overlay applications. For example, Chord [25], Bayuex [33], and Brocade [31] are designed for well-defined-structure-based overlay applications. Their main goal is to locate content in a large distributed system. In [18] and [24], the authors mainly focus on overlay-based multicasting protocols.

To facilitate the support for existing and new overlay applications, especially for QoS-aware applications, a unified framework is desired in the Internet to support overlay applications. Instead of application-specific overlays, we propose to develop a general overlay service network (OSN) that can be shared by a variety of applications. We introduce the idea of overlay brokers (OBs) to achieve this goal. The OBs are strategically placed across the Internet domains. Each of the Internet domains can have one or more OBs. These OBs provide a unified platform to serve several overlay applications at the same time. We believe that this framework can provide efficient overlay services to the existing overlay applications and facilitate the implementation of new applications.

In the proposed OSN, at each of the OBs, an overlay service layer (OSL) is implemented between the transport layer and the application layer. OSL provides the common functionalities that most overlay applications need, such as topology discovery, overlay link performance estimation, overlay routing, resource allocation, etc. When a new application arrives, using the routing protocol, the OSL first forms a logical application-specific overlay topology as a subgraph of the OSN, connecting the appropriate OBs. Then, it allocates the required network resources according to the application's requirement.

Manuscript received November 15, 2002; revised May 1, 2003. This paper was supported in part by the National Science Foundation under Grant CCR-0 296 070 and Grant ANI-0 296 034.

The authors are with the Department of Computer Science, University of California, Davis, CA 95616 USA (e-mail: lizhi@cs.ucdavis.edu; prasant@cs.ucdavis.edu).

Digital Object Identifier 10.1109/JSAC.2003.818782

As the overlay network is built on top of the general Internet infrastructure, the problem of QoS-aware overlay routing is somewhat different from Internet protocol (IP)-based QoS-aware routing. Here, the QoS-aware routing path is constraint by the topology of the OSN and the resources available at the OBs. Besides considering the dynamic overlay link capacity, we also need to consider the computation capacities of the OBs, which limits the number of overlay sessions the OBs can support simultaneously. Furthermore, some applications may require computations (such as, encryption/decryption and compression/decompression) at the OBs, thus requiring the consideration for computational load in selecting the routing paths.

The focus of this paper is to propose QoS-aware routing protocols for overlay network (QRON). QRON runs at the OSL among the OBs. The main function of QRON is to search QoS-satisfied overlay paths forming overlay networks for upper layer QoS-sensitive overlay applications, and balance overlay traffic load on OBs and overlay links. We believe that such generic overlay routing protocols will greatly reduce the cost of designing QoS-aware overlay routing protocols for each specific overlay applications. With the increase in QoS-sensitive applications in the Internet, it is necessary to support QoS-aware routing service for the whole Internet. With the help of OBs, QRON can also be used to provide end-to-end QoS-aware routing services without significant changes in the Internet infrastructure.

To enhance scalability, we have relied on a hierarchical approach for QRON. In this approach, groups of contiguous (in terms of network distance) OBs are grouped to form clusters, which in turn are grouped together to form super-clusters, and so on. QRON operates hierarchically by using intercluster routing recursively. Thus, the state maintenance and overheads due to the exchange of routing messages are reduced drastically. We have proposed two approaches for selecting an appropriate path at each level of the hierarchy: modified shortest-distance path (MSDP) and proportional bandwidth shortest path (PBSP). Both of the algorithms are based on Dijkstra shortest path searching algorithm. When searching for a QoS-satisfied overlay path, MSDP and PBSP consider both the OBs' capacities and the overlay links' capacities to balance overlay traffic. Simulation results have shown that both of the algorithms (MSDP and PBSP) can effectively find and provide QoS-satisfied overlay path connecting the source OBs and destination OBs (achieving higher success rate). At the same time, they also balance the distribution of overlay traffic across the OSN.

The rest of the paper is organized as follows. In Section II, we introduce the Overlay Service Network. The basic idea of QoS-aware overlay routing protocol and the two QoS-aware overlay routing algorithms, MSDP and PBSP, are described in Section III. A series of simulations and results are discussed in Section IV. The related works are discussed in Section V. Finally, we draw the conclusions in Section VI.

II. Overlay Service Network

Most of the previous work on overlay networks can be categorized into two broad categories: overlay based on end-hosts and overlay based on fixed-nodes. Examples of overlay networks based on end-hosts include peer-to-peer (P2P) file sharing, end-host-based multicasting in the Internet, etc. [8], [13]. In these networks, the end hosts are logically connected (at the network layer) together to form overlay networks without any support from the network providers. However, each end host is required to be connected to several other end hosts at the same time. As pointed out in [18], there are two factors constraining the applicability of this scheme: lower transmission bandwidth of end hosts and the long "last mile" transmission delay for overlay applications. Most of the end hosts have low bandwidth access to Internet. They usually use cable modem, DSL, or dial-up service. A smaller portion of hosts have 10/100 Mb/s access links connected to the Internet. It is well known that the access network is where the congestion often happens. Besides, dynamic and uncontrolled environment of pure end-host based overlay networks usually cannot provide reliable service because the end host can be turned "off" or "on" any time.

The second category of overlay networks use fixed nodes distributed in the Internet to facilitate overlay services. The use of fixed nodes for overlay multicast services [6], [16], and content distribution service [1] has been addressed in the literature. Most of these fixed nodes are deployed to achieve similar functions, such as building appropriate topology, tracking dynamic network condition, etc. However, with expanding and evolving overlay applications, it becomes redundant to set up special fixed nodes for each overlay application in the Internet. Rather, it is desirable to have special overlay service nodes in the Internet to facilitate a broad class of applications requiring overlay services.

A. Overlay Broker (OB)

In this paper, we propose the concept of OBs, which are specialized nodes that can be placed in the Internet by a third party [termed as overlay service provider (OSP)] to provide generic overlay service support to overlay applications. Usually, one or more OBs can be deployed in each domain. These OBs are interconnected at the application layer to form the OSN. The OBs cooperate with each other across the Internet to provide overlay services support. OBs can be placed either at the edge of a domain or in the core. The OBs subscribe high bandwidth connections to the Internet backbone. The OBs of one domain know the addresses of the OBs of the neighboring domains. This knowledge can be incorporated during deployment or through exchange of messages with the neighboring domains. The OBs are also responsible for encapsulation and decapsulation of the outgoing and incoming packets of the overlay network, respectively.

Fig. 1 shows a generic network topology composed of four autonomous systems (ASes). The figure displays the possible location and existence of the OBs. Each of the AS has one or two OBs in the figure. More number of OBs may be added incrementally in a domain to enhance scalability and fault tolerance.

The internal protocol architecture of OBs is shown in Fig. 2. The OBs have an OSL which lies between the application layer and the transportation layer. OSL supports the overlay service function to facilitate the deployment of overlay applications.

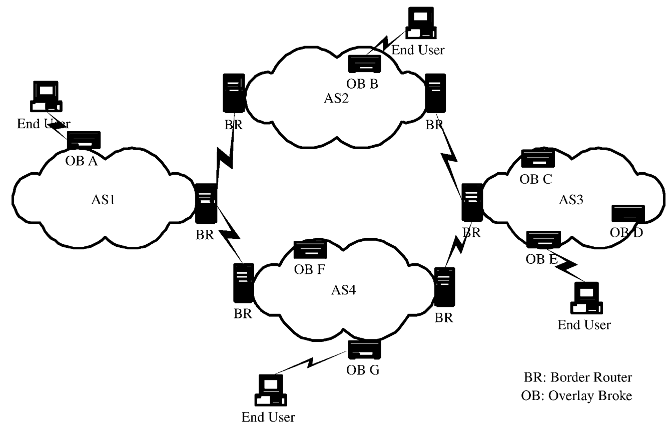

Fig. 1. Example network with OBs in each of the ASes.

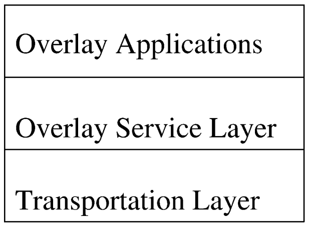

Fig. 2. OB structure.

OSL includes several modules with the following functionalities: topology discovery, OB-to-OB performance estimation, resource allocation, and service provisioning. Description of these functionalities follow in the next section.

B. Functional Modules of OSN

For a generic OSL, several functional modules are needed for the OSN. In [4], the authors have discussed several service modules. Here, we have focussed primarilyon the QoS-related functionalities.

1) Topology Discovery: With the deployment of OSN, there will be several OBs in the Internet. The global overlay topology is formed based on the following observation. If two OBs are within the same AS, there will be an overlay link connecting the two OBs. If there is an interdomain link between two domains, there is an overlay link in the overlay topology which connects the two OBs from the two domains. If more than one interdomain links connect two ASes, the overlay links will also correspond to the number of such links. The overlay link connecting two or more OBs in the overlay topology may physically pass through multiple ASes that do not have any OBs. The OBs within the same domain can form full mesh connecting each other, but the OSN interconnecting multiple domains is not a full mesh topology.

Fig. 3 shows an example of overlay topology derived from the corresponding physical topology that was shown in Fig. 1. Note that the links shown in Fig. 3 are virtual links at the application layer which may map onto multiple links at the network layer.

In this paper, the term *overlay topology* refers to the topology that connects all the OBs in the Internet. The connection is not full mesh, but is based on the configuration and connectivity of ASes. The *application-specific overlay topology* refers to the topology (connecting a subset of Internet OBs and, thus, a subgraph of OSN), which is formed by the OSL to provide service for a specific application. Using the topology discovery module, the OBs can discover and adjust the overlay topology which connects all the OBs with the aid of the physical topology underlying the overlay topology. Based on this information, OSL can utilize the overlay routing protocol to form application-specific overlay topologies which connect the appropriate OBs while meeting the applications' QoS and resource requirements.

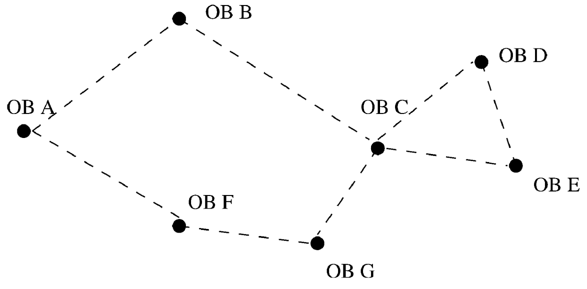

Fig. 3. Example of the formation of overlay topology.

2) Performance Measurements of Overlay Links: An overlay link is usually composed of multiple physical links. Other nonoverlay traffic would be using the same physical links. As mentioned earlier, the overlay functions work on top of the transportation layer. Thus, the OB cannot control or manage the IP-layer resources. To obtain the performance of an overlay link, we can only rely on measurements. Many efficient measurement methods have been proposed in the literature, such as, Ping and Sting [22], which can help us track the overlay link loss rate and delays. To derive the overlay available bandwidth, we can use direct measurements. We actively send traffic between two OBs and see how much of traffic can get through before the path gets saturated and starts losing packets. This method was also adopted in [19] and [32]. We can also use a more lightweight measurement method, such as cprobe [5] to obtain the available bandwidth in a path.

3) Resource Allocation: The resource allocation function mainly deals with allocating the network resources to overlay applications based on their QoS requirements and the application-specific overlay topology. With the dynamically changing network conditions, the OSN may adjust the application-specific overlay topology or the resources allocated to the overlay network.

We can use the utility-based resource allocation method which was proposed in Opus [4]. Each customer is associated with a value which is based on the service level agreement (SLA). Opus tries to make dynamic tradeoffs between service quality (associated with value) and service cost. It makes the resource allocation decisions by comparing the expected utility of a set of candidate configurations. The goal is to maximize the global utility (value). To achieve this goal, it uses models to predict the different allocation methods for service quality. Based on this model and the SLAs, we can determine the expected value of the predicted allocation methods.

4) Service Provisioning: Within each AS, the identity of the OBs are well known or can be obtained from a directory service. When a new customer wants to subscribe overlay services from the OSN, it first contacts an OB (called access OB). Based on the service type, the access OB can determine the topology that would connect the required OBs, the QoS requirement of the overlay path, as well as the computing capacity requirement of the intermediate OBs. Then, using the QoS-aware routing algorithm (which we will discuss in the following section), it can find all the necessary paths which will compose the application-specific overlay.

For end-to-end QoS provisioning, two additional components that need to be considered are: 1) QoS-satisfied path from the source node to the source OB and 2) QoS-satisfied path from the destination OB to the destination node. These two components in the path discovery process can be handled through the intradomain QoS provisioning mechanism available in the ASes (we have assumed this availability).

III. QoS-Aware Routing in Overlay Networks (QRON)

A. Problem Description

To provide overlay service, an OSP needs to subscribe underlying network services from Internet service providers (ISPs). There are two possible choices that can be envisioned. The first choice is to subscribe access service from ISPs, where the ISPs can provide the required aggregate overlay bandwidth and loss rate. However, it cannot guarantee the path performance for each source-destination pair. The second choice is to subscribe virtual private networks (VPNs)-like services connecting all the OBs. In this approach, the ISPs can set up tunnels connecting all the OBs and can, thus, guarantee the performance of each path. However, the overlay service topology would be fixed which would not be flexible enough to provide services to different kinds of overlay applications. In this paper, we assume that OBs subscribe to the first kind of network services from the ISPs.

Overlay routing module provides desirable QoS-aware routing service to overlay applications ensuring that the communication between the OBs meet different applications' QoS requirement. The overlay path searching process is different from the problems of network-level QoS routing and existing solutions to these problems are not readily applicable to the OSN. QRON in OSN is different from QoS-aware IP routing in the following ways. The network layer protocols can directly accesses the links and router resources. They can directly retrieve the network availability information and reserve the resources along the QoS-satisfied path. Once the resources are reserved, the flow can get guaranteed QoS services. Another approach would be to get assured QoS services by using the concept of differentiated services [9]. However, in overlay routing, the OBs cannot directly access the available resources in the overlay path. It can only rely on the measurement techniques (as we discussed earlier), where the accuracy cannot be guaranteed. The OBs cannot directly control and reserve the resources in the overlay path. Other nonoverlay traffic can pass through the overlay links anytime, which results in variations of the overlay paths' service qualities. Thus, it is possible that the parts of an overlay path may not satisfy the application requirements during the data transfer process. Another difference is that in OSN, OBs may support other application functions, such as encoding, decoding, compression, decompression, temporary storage service, etc. The processing capacities of the OBs are also important factors that needs to be considered while selecting an overlay path.

In summary, the problem can be stated as follows. How to select QoS-aware overlay paths and route data based the QoS requirements and the dynamic overlay link quality?

B. Basic Idea

We propose a routing framework called QRON for a generic OSN.[1] The primary goal of QRON is to provide a QoS-satisfied overlay path from the source OB to the destination OB in the overlay topology. To achieve this goal, we need to identify a subset of the overlay topology that provides the connectivity, as well as the required QoS between the source OB and the destination OB. Critical resources in the OSN includes not only the overlay link capacity, but also the OBs capacity. OSNs connectivity depends on the bandwidth availability of overlay links and the capacity of the OBs.

QRON uses the following approach to provide QoS-aware overlay routing services in the dynamic network environment.

1) While selecting the overlay links, QRON tries to balance the traffic among the overlay links and OBs in addition to satisfying the QoS requirement. This approach ensures that the overlay traffic will be resilient to the background nonoverlay traffic. At the same time, additional overlay traffic will have less impact on the existing overlay traffic when the overlay path quality is degraded.
2) QRON is a source-based routing protocol. To improve the efficiency of the source-based routing, QRON uses hierarchical architecture to organize the OBs. The hierachical organization provides a scalable topology for distributing the overlay link and OB state information. Each OB can then have the aggregated overall topology. When an overlay routing request arrives, the OB can utilize the aggregated topology to find an approximate path. It then contacts some of the OBs on the path to get detailed and up-to-date information about the path performance. Even though this approach incurs some control message overhead, the up-to-date path performance information helps in improving the possibility for providing QoS service satisfaction.
3) During the overlay data routing process, the nonoverlay traffic may increase suddenly and thereby could affect the normal overlay data traffic. To cope with this situation, QRON uses an adaptive routing approach. When an OB realizes that the additional overlay link capacity of an overlay link is less than the overlay traffic it is currently servicing, it begins to search several backup overlay paths connecting itself to this neighboring OB. These backup paths will make sure that the overlay traffic can bypass this overlay link if the quality degrades. If the OB cannot find enough backup paths, it will notify some of the previous hop OBs to search for backup paths to bypass the degraded overlay link.

[1]There are many QoS metrics we need to consider for QoS-aware routing services, such as bandwidth, delay, loss rate. In this paper, we only focus on bandwidth. In most cases, higher bandwidth has higher possibility for low delay, and can provide controlled loss rate service as shown in OverQoS [26].

C. Path Selection

QRON is developed considering the constraints of the available computation capacity of OBs, and the available bandwidth on the overlay links.We assume that the OBs periodically probe their own residual computation capacity by querying the operating system or by running some benchmark program. Based on a hierarchical approach (discussed in the next section), each OB maintains an aggregated overlay topology. When a new overlay routing request arrives, its required bandwidth and computational capacity are estimated and specified by the source OB.

The path selection problem can be formalized as follows. The overlay topology of the Internet can be modeled as a connected directed graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, where V is the set of OBs ($\mathrm{OB}_1 \ldots \mathrm{OB}_{N_{\mathrm{OB}}}$, where N_{OB} is the number of OBs) and E is the set of overlay links in the overlay topology.For OB_i, C_i represents its current available computation capacity. The overlay link between OB_i and OB_j is denoted as L_{ij} or (i, j), and its available bandwidth is denoted as B_{ij}. When an overlay routing request arrives at OB_S with destination OB_D, with a bandwidth request of RB and the required computation RC, the QRON algorithm should find an overlay path which can satisfy the QoS requirements while balancing the residual link capacity and the computation overheads of the OBs.

The path selection algorithm is based on Dijkstra's least-cost path routing algorithm. In addition to satisfying the QoS requirements, our goal is to balance the load on the overlay links, as well as the OBs. The key component of this algorithm is the "cost" factor, which could be defined in a variety of ways. We propose two approaches. The main difference between them lies in the definition of cost of the overlay links and the function defining the path cost. In one of the approaches, we try to choose paths based on a constraining resource component (link bandwidth, OBs capacity), while in the other approach, the cost is considered linearly dependent on all the components.

1) Modified Shortest-Distance Path (MSDP): A shortest-distance path algorithm was proposed in [17], which guarantees that the packets always travel along the lightest path (the path with the maximum available resource). Based on this notion, the algorithm can effectively balance the network load among the physical links and efficiently utilize the network resources. Suppose R_{ij} is the available bandwidth on link L_{ij}, the weight of the link is defined as $(1/R_{ij})$. Then, based on Dijkstra's shortest-path algorithm, one can find a shortest-distance path.

Based on this approach, we propose a MSDP algorithm. In MSDP, the weight or the distance function of the overlay link (i, j) is defined as

$$\mathrm{weight}(i, j) = \mathrm{MAX}\left(\frac{\mathrm{RB}}{B_{ij}}, \frac{\mathrm{RC}}{C_j}\right) \tag{1}$$

where RC denote the required computational capacity, RB is the required bandwidth, and C_j is the available computational capacity at the OB at the other end of the overlay link (i, j). The maximum value of the ratio of required resources to the available resources determines the weight. Here, the weight is dictated by the bottleneck resource component of the overlay connection between two OBs. Thus, it is ensured that the bottleneck link has the highest weight that gets added on to determine the path cost. The link with the lowest cost will have a component (B_{ij}, C_j) that has highest resource availability in terms of the required resources.

The weight function in (1) depicts that the overlay link weight not only depends on the overlay link capacity, but also on the OBs' computational capacities. This definition of overlay link weight ascertains that the new incoming request always travels the path with larger capacity and effectively balances the overlay traffic among overlay links, as well as OBs.

If an overlay path passes through the $\mathrm{OB}_1, \ldots, \mathrm{OB}_n$, the weight of the path is defined as

$$\left(\sum_{i=1}^{n-1} \mathrm{weight}(i, i+1)\right) * P(h) \tag{2}$$

where $P(h)$ defines a function that is proportional to the number of domains the overlay passes through. This factor enables us to select paths that pass through less number of ASes. $P(h)$ can be set to 1 if the overheads of passing through the ASes is not an issue. Using (2) and based on Dijkstra's shortest-path algorithm, we can find an overlay path. The overlay path based on MSDP algorithm can satisfy the QoS requirement, as well as balance the traffic among the overlay links and the OBs.

2) Proportional Bandwidth Shortest Path (PBSP): Another approach for defining the weight factor is to include the influence of all the resource components (link bandwidth and the computational capacities of the OBs). PBSP is based on this philosophy. Unlike MSDP, PBSP approach balances the load with respect to the combined influence of all the resource components. In MSDP, the bottleneck component defines the weight.

The PBSP algorithm is based on the following criteria. If link (i, j)'s capacity B_{ij} is larger than link (m, n)'s capacity $B_{mn}(B_{ij} > B_{mn})$, then the probability of choosing link (i, j) (denoted as P_{ij}) should be larger than choosing link (m, n) (denoted as P_{mn}). That is, $P_{ij} > P_{mn}$. In other words, the goal of PBSP is to maximize the residual bandwidth at any link for any path. Suppose a routing request comes with bandwidth requirement RB ($B_{ij} > \mathrm{RB}$ and $B_{mn} > \mathrm{RB}$). If $(B_{ij} - \mathrm{RB})/B_{ij} > (B_{mn} - \mathrm{RB})/B_{mn}$, then $B_{ij} > B_{mn}$ and $P_{ij} > P_{mn}$. Based on the criteria, we define the probability of choosing link $(\mathrm{i}, \mathrm{j}), P_{ij}$, as $(B_{ij} - \mathrm{RB})/B_{ij}$. Thus, the weight of the link can be defined as $((B_{ij})/(B_{ij} - RB))$.

Similarly, when considering both the OBs' capacities, the weight of the link can be defined as:

$$\mathrm{weight}(i, j) = \frac{B_{ij}}{B_{ij} - \mathrm{RB}} * \frac{C_j}{C_j - \mathrm{RC}}. \tag{3}$$

If there are n OBs in the path (from OB_1 to OB_n), the weight of the path is defined as:

$$\left(\sum_{i=1}^{n-1} (\mathrm{weight}(i, i+1))\right) * P(h). \tag{4}$$

Then, based on Dijkstra's shortest-path algorithm, we can find a shortest-distance path. The overlay path based on MSDP algorithm can satisfy the QoS requirement, as well as balance the traffic among the overlay links and OBs.

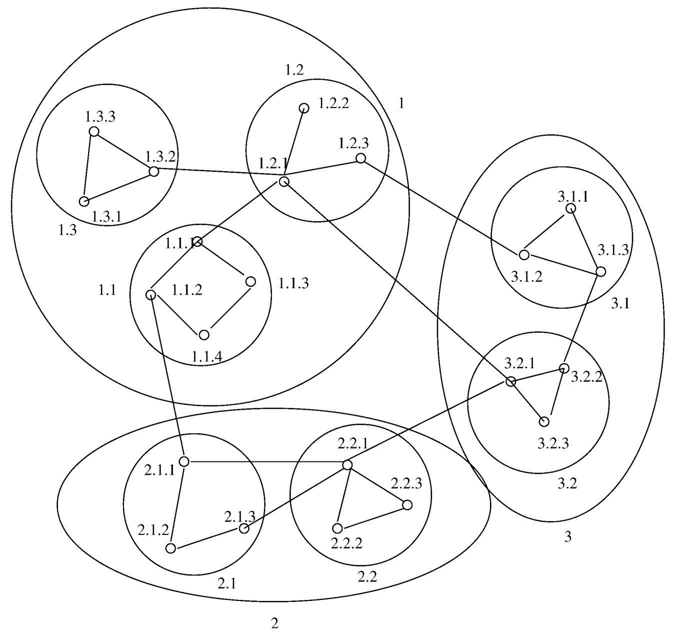

Fig. 4. Example of the hierarchical structure of overlay topology.

D. Hierarchical Organization

To adopt the approaches described and derived in the previous subsection, we assume that each node knows the residual capacity of all the links and nodes in the OSL. This knowledge at the nodes can be acquired by requiring that each node periodically broadcast its residual computation capacity and the residual bandwidth of the attached overlay links. Each and every OB in the OSN will, thus, need to maintain the state information of all other OBs. Therefore, this method is not scalable to large OSNs. In this section, we propose a hierarchical organization of the OSN to facilitate a scalable QRON algorithm.

We assume that the OBs are organized in a logical hierarchical structure. An example of such a structure is shown in Fig. 4. The hierarchical structure can be set up manually or self-constructed by the OBs using some methods, such as the top-down hierarchy (TDH) approach described in [27]. Our approach of clustering is based on the following guidelines: 1) the OBs within the same AS are clustered together; 2) physically closer OBs/clusters are clustered together; and 3) if two OBs/clusters have multiple overlay links connecting them, they are clustered together. The hierarchy is created by logically grouping OBs into clusters, grouping clusters into next-level-clusters, and so on. The highest level of these cluster is denoted as level-1, which is composed of a group of level-2 clusters, and so on. In Fig. 4, the OBs themselves form the level-3 clusters. A group of contiguous (from the network standpoint) OBs form the level-2 clusters. A group of level-2 clusters form the level-1 cluster. Thus, in the example shown in Fig. 4, there are 22 level-3 clusters, 7 level-2 clusters, and 3 level-1 clusters.

If there are n levels, the OBs themselves are denoted as level-n. Cluster ids at the ith level are denoted as an *i-tuple* notation, similar to the one used in [28]. The level-1 clusters have single number ids. The ids of level-2 clusters are in the form of $x.y$, and that of level-3 are $x.y.z$. The id $x.y.z$ of a level-3 cluster indicates that the id of the corresponding level-1 and level-2 clusters are x and $x.y$, respectively. The OBs maintain the topology and cost information of the views of level-1 clusters, level-2 clusters, $\ldots$, level-n clusters. To provide the above information to all the OBs, we adopt the following mechanism. Each OB periodically broadcast its computation capacity and the attached overlay link capacities information. This information is broadcast only within its own cluster. After collecting these information, each OB can have the topology information of its own cluster level. Based on this information, the gateway OBs will begin to aggregate the cluster into simplified full-mesh topology connecting all the gateway OBs. The methodology is adopted as follows. The gateway OB first gets the widest overlay paths (with the maximal available overlay bandwidth and OB computation capacity) for each pair of gateway OBs within this cluster. If the widest overlay path

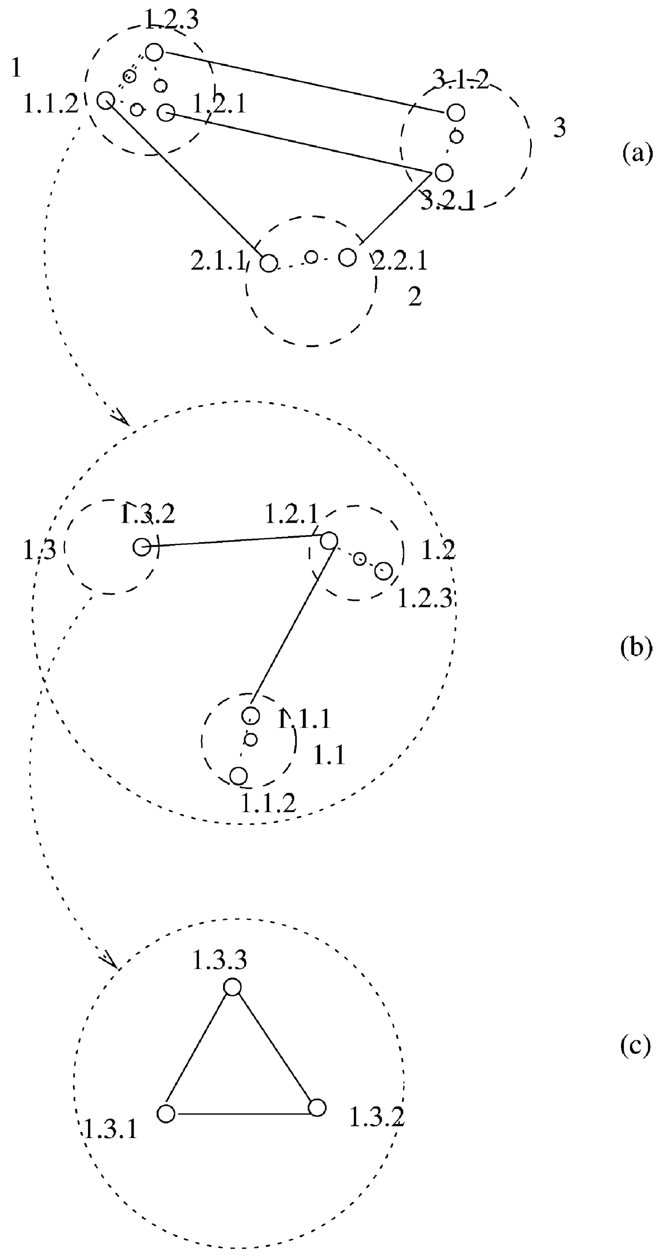

Fig. 5. Example of different levels of overlay topology.

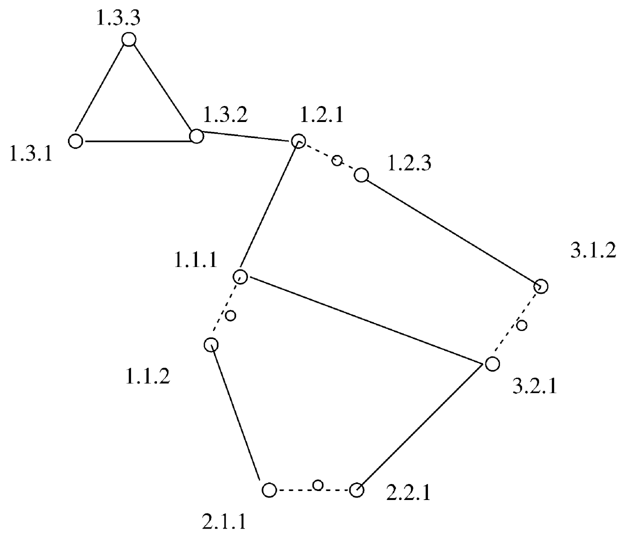

Fig. 6. Example of combined cluster topology.

connecting two gateway OBs is composed of an overlay link, there is a virtual overlay link between these two OBs in the full mesh. Otherwise, there is a virtual path connecting the two OBs. The virtual path is composed of two virtual overlay links and the virtual OBs. Their capacities are set as the capacity of the widest path. The gateway OBs of this level broadcast their aggregated information to the other OBs, and the process is iterated until we reach the highest level of clusters. Note that not all the OBs are involved in the intercluster information exchanges, thereby reducing the overheads.

We describe the QRON algorithm by first analyzing how the algorithm works for an example network, followed by the presentation of the formal algorithm. Consider an application that requires routing information from the source OB 1.3.3 to the destination OB 3.2.3 in Fig. 4. As mentioned earlier, the source OB 1.3.3 maintains the topology information on a hierarchical basis as shown in Fig. 5. The dotted lines and dotted nodes are virtual overlay links and virtual OBs, respectively. After combining the three levels of cluster topologies, it can have a combined cluster topology as shown in Fig. 6. The request for the overlay service first locates an OB which share the lowest level cluster as the destination OB, i.e., 3.2.1 in this example. Based on the path selection algorithms, discussed in the previous subsection, it can find a path from 1.3.3 to 3.2.1, such as 1.3.1 $\rightarrow$ 1.3.2 $\rightarrow$ 1.2.1 $\rightarrow$ 1.1.1 $\rightarrow$ 1.1.2 $\rightarrow$ 2.1.1 $\rightarrow$ 2.2.1 $\rightarrow$ 3.2.1. Notice that the path 1.1.1 $\rightarrow$ 1.1.2, and 2.1.1 $\rightarrow$ 2.2.1 are composed of virtual overlay links. Then, we need to send the routing request to 1.1.1 and 2.1.1 with destination 1.1.2 and 2.2.1, respectively. At the same time, OB 1.3.1 also sends a routing request to 3.2.1 with destination 3.2.3. If all the routing request are honored with a response of the detailed path information, OB 1.3.3 can provide a full overlay path to the user. If OB 1.1.1 cannot find a path, we will remove the virtual path 1.1.1 $\rightarrow$ 1.1.2 from the combined cluster topology and repeat the previous step. If OB 3.2.1 fails to find a path to 3.2.3, we will remove the virtual link from the the combined cluster topology. Then, it will find another OB which shares the lowest level cluster with 3.2.3 (3.1.2) and repeat the process.

Since the path search process is not exhaustive, it can be easily shown by counter examples that the hierarchical approach may not necessarily result in finding the optimal path, which is true. Our approach is to minimize the complexity and state maintenance overhead, while determining a near-optimal path.

Note that even though this routing method can incur additional delays, the OB tries to get the up-to-date information which can better adapt to the dynamic overlay link variations. At the same time, it also can achieve good scalability.

The routing algorithm can be formalized as follows. Let there be n levels of clusters. The nth level clusters are the singular OBs. If the source OB is denoted as $S_1.S_2.\dots S_n$ and the destination OB as $D_1.D_2.\dots D_n$, then the desired path can be determined using Algorithm 1.

Algorithm 1

```
$PATH(S_1.S_2.\dots S_n, D_1.D_2.\dots D_n)$
1. Rank all the OBs in the combined cluster topology into the list
   OBList based on descending order of sharing the lowest level of
   cluster with $D_1.D_2.\dots D_n$
2. While $(\mathrm{OBList!} = \mathrm{NULL})$ and (destination cluster is connected)
3.   Remove the first OB from OBList, $X_1.X_2.\dots X_n$
4.   Find a path from $S_1.S_2.\dots S_n$ to $X_1.X_2.\dots X_n$ using either MSDP
     or PBSP.
5.   While the path exits(denoted as $S_1.S_2.\dots S_n \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow$
     $\dots X_1.X_2.\dots X_n$)
6.     Determine all the virtual overlay paths from this path.
7.     Send routing request to the begining node of the virtual overlay
       link with the destination as the end node of the overlay link.
8.     If $(X_1.X_2.\dots X_n! = D_1.D_2.\dots D_n)$
9.         Send routing request to $X_1.X_2.\dots X_n$ with destination
           $D_1.D_2.\dots D_n$
10.    Collect the reponses.
11.      If all of them return successful response,
12.    Return with the full path.
13.    Otherwise,
14.      For those OBs which failed to provide positive response,
15.          remove the attached virtual overlay links from the
             combined cluster topology.
16.        If $X_1.X_2.\dots X_n$'s attached virtual overlay link exists
17.          Find another path $S_1.S_2.\dots S_n$ to $X_1.X_2.\dots X_n$
```

IV. Performance Evaluation

In this section, we evaluate the proposed QRON algorithms using simulations. The goal of the simulation-based study is to evaluate the following performance of the two variants of QRON algorithms (MSDP and PBSP).

1) Balancing overlay traffic among the overlay links.
2) Balancing the overlay traffic overhead among the OBs.
3) Finding and providing QoS-satisfied paths connecting the source OBs and destination OBs.
4) Overlay path penalty compared with IP-layer routing protocols. This aspect is measured in term of link stress penalty, which is defined as the ratio of the number of hops an overlay path passed to the number of links passed if using least-cost IP-layer routing protocol.

The results obtained for QRON using both MSDP and PBSP approaches are compared with that of the shortest-path routing (SPR) algorithm.Note that the SPR refers to the shortest path in the overlay network, not in the IP layer.

A. Simulation Setup

We have implemented a session-level event-driven simulator to evaluate and compare the performance of QRON and SPR algorithms. The simulations are based on the Georgia Technology Internetwork Topology Model (GT-ITM) [14], which is used to generate the network topology. Unless specifically mentioned, the topology we used in this section has 1000 nodes that are evenly distributed across 100 domains. In our simulation, we assume that there is an OB in each domain. When simulating the two algorithms (MSDP and PBSP), the OBs form a two-level hierarchical topology at the application layer. Each of the clusters in the hierarchy has an average of ten members (subclusters or OBs).

The dynamics of the overlay routing is modeled as follows. The overlay routing request arrives at a random OB according to a Poisson distribution with rate λ. The destination domain is randomly chosen. The holding time of the overlay session is exponentially distributed with a mean of 2 min. Similar to [23], the offered load of the overlay routing request is defined as $\rho = (\lambda * h/\mu * (\sum L_i))$, where h is the mean overlay path hops (number of OBs in the path), and $\sum L_i$ is the sum of the overlay link capacities in the corresponding overlay topology. During the process of simulation, we vary the value of μ to test QRONs' performance under different offered loads.The physical links' bandwidths during the simulation are randomly selected between 40 and 240 units with delay 2 ms, while the OBs' capacities are uniformly set as 800 units. The nonoverlay traffic occupy around 50% of each physical link's capacity. The nonoverlay traffic varies its volume $\pm$20% every 500 ms. The OBs exchange their state information every 1000 ms. We assume that the error of available bandwidth measurement result is within $\pm$10%. For each overlay routing request, we use an overlay routing protocol to set up an overlay path connecting the source OB and the destination OB with a bandwidth requirement range of 1–6 units. The computation capacity requirement is varied between 6–10 units. The value of $P(h)$ is set to 1.

To simulate different dynamic network situations, we repeat our simulations on the following two different network scenarios.

- Scenario 1: 80% of the overlay routing requests' source and destination pairs are from 50% OBs, while others are uniformly distributed among all the other OBs.
- Scenario 2: 80% of the overlay routing requests' source and destination pairs are from 25% OBs, while others are uniformly distributed among all the other OBs.

In the Internet, most of the interdomain traffic is concentrated across a smaller subset of ASes. Scenario 2 is more reflective of the real Internet environment. Scenario 1 is considered as a conservative estimate to show the performance in a worse-case scenario.

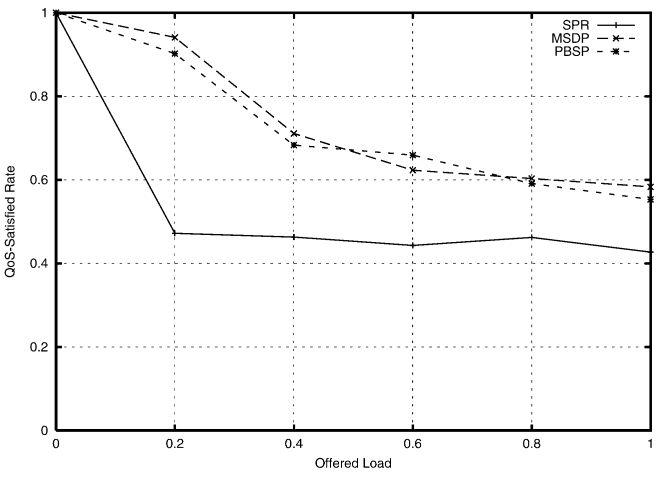

Fig. 7. QSR comparison of Scenario 1.

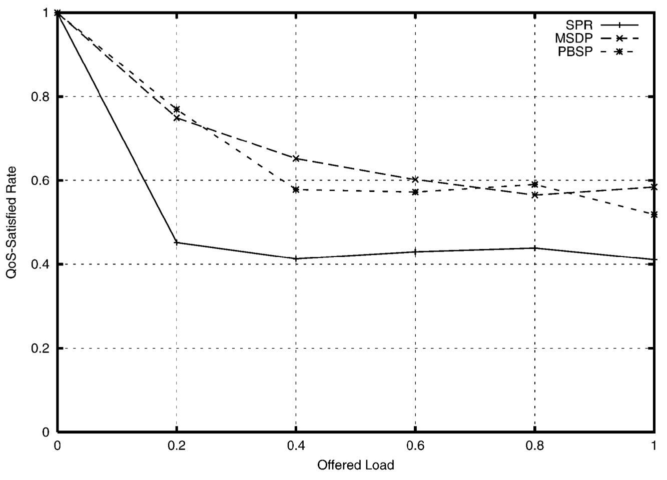

Fig. 8. QSR comparison of Scenario 2.

B. Simulation Results and Discussions

For each of the above scenario, the OSN uses each of the algorithms (QRON with MSDP, QRON with PBSP, SPR) to find and provide overlay paths to the 1000 overlay sessions. During the simulation, we measured the following performance metrics: QoS satisfaction rate, link stress penalty, residual overlay link capacity deviation, and OB residual capacity deviation.

1) QoS-Satisfaction Rate (QSR): Because of the unbalanced distribution of Internet traffic, in many situations, the shortest-path-based routing protocol cannot provide a QoS-satisfied path connecting the source and destination domains. To quantify this factor, QSR is defined as

$$\mathrm{QSR} = \frac{\text{Number of QoS satisfied overlay paths}}{\text{Number of overlay routing requests}}. \quad (5)$$

Fig. 7 shows the QSR of MSDP and PBSP compared with SPR in Scenario 1, while Fig. 8 shows the results for Scenario 2. From the two figures, we can observe that both MSDP and PBSP can greatly improve the QSR. In addition to finding QoS-satisfying overlay paths, QRON also helps in finding paths that are not affected significantly by the nonoverlay traffic. The advantage is more prominent in Scenario 2, because the traffic distribution is much more unbalanced. Here, MSDP and PBSP

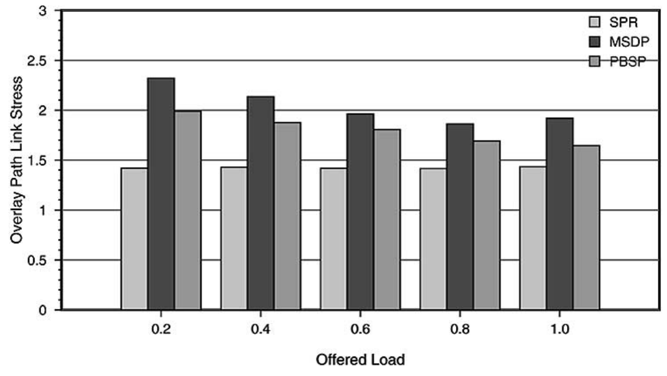

Fig. 9. Average overlay path link stress (Scenario 1).

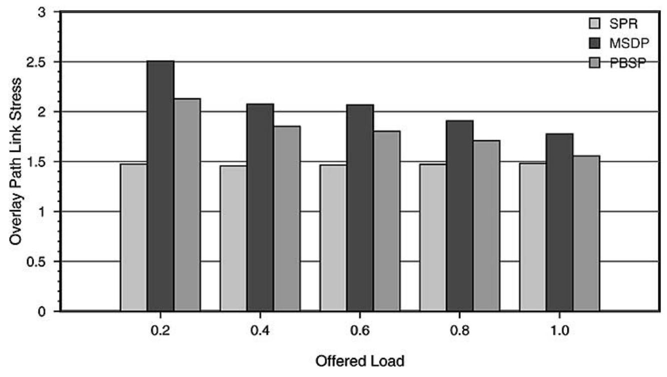

Fig. 10. Average overlay path link stress (Scenario 2).

demonstrate about 20%–30% benefit of QSR for a wide range of loads. When comparing MSDP and PBSP, we observe that the former is little better than the latter. Another trend that we observe from the figures is that with the increase in offered load, the QSRs decrease and the QSR difference between QRONs and SPR is also reduced. This is because when all the overlay links are nearly overloaded, the chance of finding a feasible overlay path also diminishes.

2) Link Stress Penalty: To satisfy the user's QoS requirement, the QoS-satisfied overlay paths are usually longer than the corresponding IP-layer least-cost paths. Link stress is used to evaluate this penalty of QoS-aware overlay routing protocols. It is defined as the number of IP-layer hops crossed by the QoS-satisfied overlay path divided by the number of links crossed via the corresponding IP-layer path. Thus, lower link stress penalty is better.

Figs. 9 and 10 show the simulation result of the two protocols' average link stress penalty in the two scenarios. Results show that the link stress penalty for the two protocols are within the range of 1.5–3.0 under the simulated network topology and dynamic network situation. This overhead is not significant considering the gain in QSR and in other performance parameters. For the same scenario, MSDP has higher link stress penalty than PBSP, which implies that MSDP chooses longer paths compared with PBSP for satisfying the QoS constraints.

3) Balancing Overlay Link Capacities: One of the most important objective of QRONs is to balance the traffic among the overlay links, which ensures that the overlay links are less affected by the dynamics of the nonoverlay traffic. The simulation results in this section illustrate this aspect. The goal of QRONs

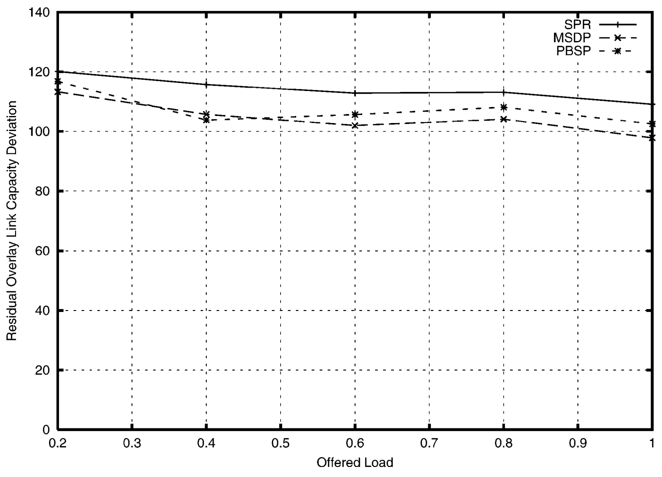

Fig. 11. RLCD comparison of Scenario 1.

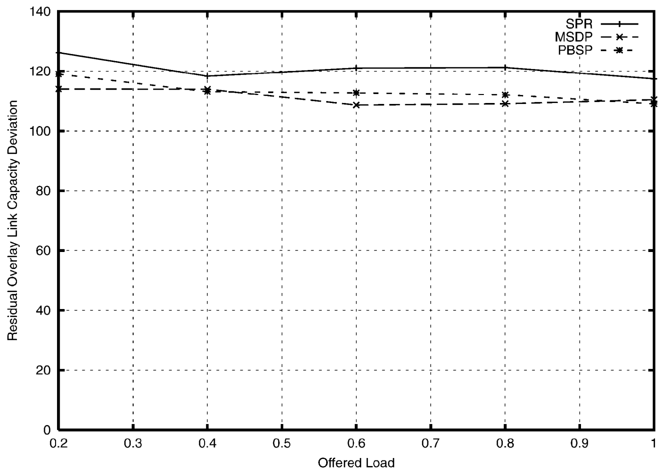

Fig. 12. RLCD comparison of Scenario 2.

is to have similar residual capacities at all links. We use the residual link capacity deviation (RLCD) to evaluate this characteristic. RLCD is defined as follows:

$$\text{RLCD} = \sqrt{\frac{\sum_{i=1}^{N}(B_i - B)^2}{N}}. \quad (6)$$

In the above expression, N is the number of overlay links, B is the average link residual capacity, while B_i is the ith overlay link residual capacity. This equation reflects how the traffic is balanced globally.

Figs. 11 and 12 depict the RLCD variations under the two scenarios after the arrivals of 1000 session requests. The RLCD of the QRONs is much less than that of SPR. From the graphs, we can also observe that MSDP and PBSP can balance the traffic much better among the overlay links with the increase in offered load and when the traffic is more unbalanced (Scenario 2). When comparing between the two QRONs, MSDP has a lower value of RLCD indicating that MSDP distributes the traffic more evenly compared with PBSP.

4) Balancing Overlay Broker Overheads: Another function of QRON is to balance the overlay routing overhead among all the OBs. QRON algorithms can achieve this goal by balancing the OBs' residual computation capacities.

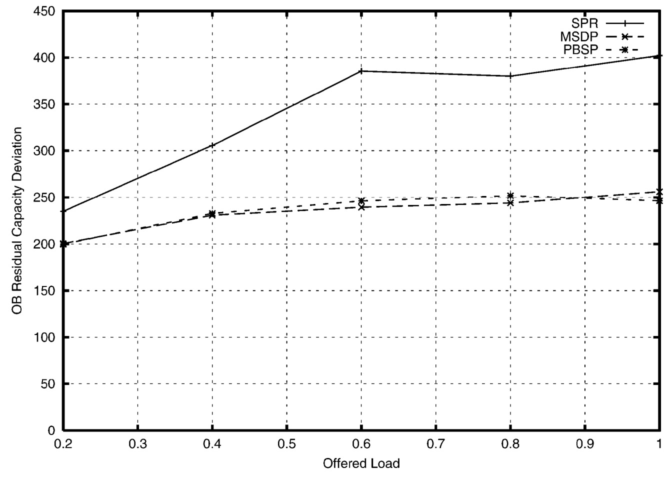

Fig. 13. Overlay broker residual capacity deviation (Scenario 1).

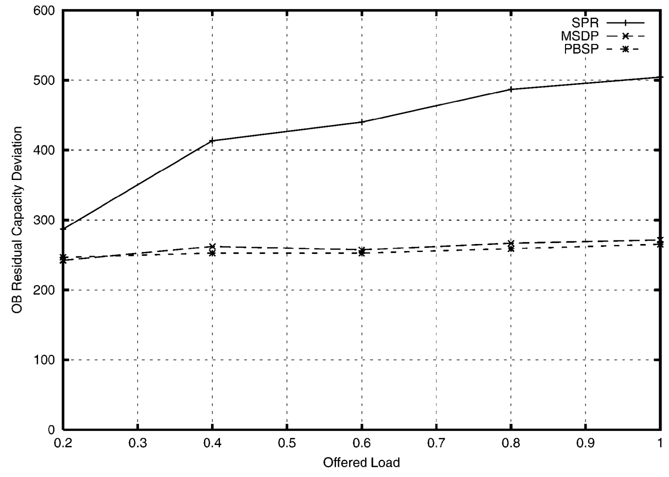

Fig. 14. Overlay broker residual capacity deviation (Scenario 2).

We use the following equation to evaluate the performance of OB load balancing:

$$\text{OB Residual Capacity Deviation} = \sqrt{\frac{\sum_{i=1}^{N_{\text{OB}}} (C_i - C)^2}{N_{\text{OB}}}}. \tag{7}$$

In the above equation, C_i is the capacity of OB_i, C is the average value of capacity, while N_{OB} is the number of OBs.

Figs. 13 and 14 show the OBs residual capacity deviations after the arrivals of 1000 overlay sessions under the two scenarios. We observe that under different session distribution situations and for different offered load, the OB residual capacity deviation remains more or less constant while SPRs balancing functions become worse with the increase in traffic load. That means, both MSDP and PBSP can achieve good performance in terms of balancing the overheads among OBs with respect to the load variations. When comparing the two protocols, PBSP can balance the load a little better than MSDP.

V. Related Work

There has been a moderate amount of work in the areas of overlay networks. Recently the efforts on this topic has been very active. Most of the efforts on overlay networks can be subsumed under two broad categories. The first category proposes overlay networks for specific applications in the Internet. The second category aims at developing generic overlay service networks in the Internet that can be used for a variety of applications.

The efforts on application-specific overlay networks have targeted for wide-use applications such as multicasting [8], [30], content distribution networks [1], and peer-to-peer file sharing [25]. Most of the proposed approaches on application-layer multicasting adopt the idea of using strategically placed fixed nodes to support overlay multicast service. The goal of the overcast model [16] is to provide wide-area content distribution. It is designed to provide bandwidth sensitive multicast services while utilizing the network bandwidth efficiently. In [6], the authors focus on how to provide scalable multicast services to real-time heterogeneous receivers. Its main goal is to reconcile the heterogeneous capabilities and network connections of various clients with the need for reliability. In [24], the author mainly focus on how to balance the multicast traffic among multicast service nodes while maintaining low end-to-end latency. However, the paper did not consider balancing the traffic among the peer links or searching QoS-satisfied overlay paths for applications. resilient overlay network (RON) [3] is also based on strategically placed nodes in the Internet domains. It is proposed to quickly detect and recover path outages and degraded performance. RON can effectively cope with the problem compared with BGP, which usually takes longer time to converge to a new valid route.

Our effort belong to the second category where we propose a general overlay service network that can be used for a variety of application-layer services. A few other works are related to this paper and also belong to the second category. Yoid [12] is a generic overlay architecture which is designed to support a variety of overlay applications that are as diverse as netnews, streaming broadcasts, and bulk email distribution. Another similar effort is the Planet-lab [20] experiment whose goal is to build a global testbed for developing and accessing new network services. It will serve as both a research testbed and a deployment platform. The main research issues being address in this project include: 1) defining virtual machine running on each node and 2) building the management services used to control the testbed. A similar approach was proposed in Opus [4], which is a large-scale overlay utility service that provides a common platform and the necessary abstractions to service multiple distributed applications. It automatically configure server network overlays with the goal of dynamically meeting the performance and availability requirements of competing applications. Their overlay topology construction is to build application overlays to enable flexible and dynamic tradeoffs between overlay cost and the associated performance and reliability. X-Bone [29] is a system for the automated deployment of overlay network. It operates at the IP layer and based on IP tunnel technique. The main focus is to manage and allocate overlay link and router resource to different overlays and avoid resource contention among the overlays.

Two other recent efforts that share similar goals of the second category are OverQoS [26] and SON [10]. OverQoS is an architecture proposed to provide Internet QoS using overlay networks. The third-party providers can utilize OverQoS to provide QoS to the customers using the controlled loss virtual link (CLVL) technique, which ensures that the loss rate observed by aggregation is very small as long as the aggregate rate does not exceed a certain value. OverQoS can be employed to provide differentiated rate allocations, statistical bandwidth and loss assurance, and can enable explicit-rate congestion control algorithms. Service overlay networks (SON) is designed to use overlay technique to provide value-added Internet services. Similar to our framework, a SON can purchase bandwidth with certain QoS guarantees from ISPs to build a logical end-to-end service delivery overlay. This work focus on the bandwidth provisioning problem: provide QoS and meet traffic demands while minimize the bandwidth cost. The authors have formulated the problem considering various factors: SLA, service QoS, traffic demand distribution, and bandwidth cost.

Our effort is complementary to most of the above approaches. We share common goals as OverQoS, OPUS, SON, and Yoid. However, in this paper, we focus on facilitating functionalities that will help achieve the overall goals. Thus, our work is mainly focused on QRONs, whose job is to set up QoS-satisfied overlay connections between OBs, balance the overlay traffic and the load on the OBs. None of the other work has addressed this aspect adequately. In addition, we address critical issues such as scalability, low overheads, and minimal state maintenance algorithms.

VI. Conclusion

In this paper, we proposed the idea of using OSN to support emerging overlay applications. We assume that each Internet AS has one or more OBs. These OBs cooperatewith each other to facilitate the deployment of overlay service.We discussed the necessary modules in OSN to support QoS-sensitive overlay applications. The focus of the paper is QRON. QRON is designed as a generic overlay routing protocol, by which OBs can form QoS-satisfied application-specific overlay topologies.

The goal of QRON is to find QoS-satisfied paths and adaptively route the overlay traffic in spite of the unpredictable overlay link performance. To achieve this goal, QRON utilizes the following methods.

1) QRON uses the MSDP or the PBSP algorithm to balance the overlay traffic among OBs and overlay links. These schemes ensure that additional traffic has less impact on overlay traffic, especially when nonoverlay traffic volume vary unexpectedly.
2) QRON uses a hierarchical organization of the OBs. This organization not only can provide scalability in distributing network state information, but also can find a path with up-to-date information.
3) QRON uses adaptive routing during data transfer. An OB begins to find a partial backup path when it realizes that its current overlay link is undergoing service degradation. It can, thus quickly bypass the link during the data transfer process. The simulation results show that the QRON algorithms can effectively find and provide QoS-satisfied overlay paths and balance the overlay traffic burden among the OBs, as well as the overlay links.

References

[1] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," presented at the ACM SIGCOMM Internet Measurement Workshop, San Francisco, CA, Nov. 2001.
[2] Y. Amir, B. Awerbuch, C. Danilov, and J. Stanton, "Global flow control for wide area overlay networks: A cost-benefit approach," *Proc. IEEE OpenArch'02*, pp. 155–166, June 2002.
[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay network," in *Proc. ACM SOSP'01*, Oct. 2001, pp. 131–145.
[4] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, and A. Vahdat, "Opus: An overlay peer utility service," in *Proc. IEEE OpenArch'02*, June 2002, pp. 167–178.
[5] R. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks," presented at the Performance'96, the Int. Conf. Performance Theory, Measurement and Evaluation of Computer and Communication Systems, Lausanne, Switzerland, Oct. 1996.
[6] Y. Chawathe, S. Mccanne, and E. A. Brewer, "RMX: Reliable multicast for heterogeneous networks," in *Proc. INFOCOM'00*, Mar. 2000, pp. 795–804.
[7] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions," *IEEE Network Mag.*, vol. 12, pp. 64–79, Nov./Dec. 1998.
[8] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. ACM SIGMETRICS*, June 2000, pp. 1–12.
[9] Differentiated services (DiffServ) [Online]. Available: http://www.ietf.org/html.charters/diffserv-charter.html
[10] Z. Duan, Z. Zhang, and Y. T. Hou, "Bandwidth provisioning for service overlay networks," presented at the SPIE ITCOM Scalability and Traffic Control in IP Networks (II)'02, Boston, MA, July 2002.
[11] H. Eriksson, "MBone: The multicast backbone," *Commun. ACM*, vol. 37, no. 8, pp. 54–60, Aug. 1994.
[12] P. Francis. *Yoid: Extending the Internet Multicast Architecture* [Online]. Available: http://www.aciri.org/yoid/docs/index.htm
[13] Gnutella [Online]. Available: http://www.Gnutella.com
[14] GT-ITM: Modeling Topology of Large Internetworks [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/
[15] *Integrated Services (IntServ)*, http://www.ietf.org/html.charters/intserv-charter.html.
[16] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr., "Overcast: Reliable multicasting with an overlay network," in *Proc. 4th USENIX OSDI*, Oct. 2000, pp. 197–212.
[17] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proc. 5th IEEE ICNP*, Oct. 1997, pp. 191–202.
[18] N. M. Malouch, Z. Liu, D. Rubenstein, and S. Sahu, "A graph theoretic approach to bounding delay in proxy-assisted end-system multicast," presented at the IWQoS'02, Miami Beach, FL, May 2002.
[19] V. Paxson, "End-to-end Internet packet dynamics," *IEEE/ACM Trans. Networking*, vol. 7, pp. 277–292, June 1999.
[20] Planet Lab. [Online]. Available: www.planet-lab.org
[21] Qbone [Online]. Available: http://qbone.internet2.edu/
[22] S. Savage, "Sting: A TCP-based nework performance measurement tools," in *Proc. 2nd USENIX Symp. Internet Technologies and Systems*, Oct. 1999, pp. 71–79.
[23] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the overheads of source-directed quality-of-service routing," in *Proc. 6th IEEE ICNP*, Oct. 1998, pp. 42–51.
[24] S. Y. Shi and J. S. Turner, "Routing in overlay multicast networks," in *Proc. IEEE INFOCOM*, June 2002, pp. 1200–1208.
[25] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 149–160.
[26] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "Over QoS: Offering Internet QoS using overlays," presented at the HotNet-I Workshop, Princeton, NJ, Oct. 2002.
[27] D. G. Thaler and C. V. Ravishankar, "Distributed top-down hierarchy construction," in *Proc. IEEE INFOCOM*, Mar. 1998, pp. 693–701.
[28] P. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," *Comput. Commun. Rev.*, vol. 18, pp. 35–42, Aug. 1998.
[29] XBone [Online]. Available: http://www.isi.edu/xbone
[30] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," presented at the INFOCOM'02, New York, June 2002.

[31] B. Y. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz, "Brocade: Landmark routing on overlay networks," presented at the 1st Int. Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, MA, Mar. 2002.

[32] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the constancy of internet path peroperties," in *Proc. 1st ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001, pp. 197–211.

[33] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination," presented at the 11th NOSSDAV, Port Jefferson, NY, June 2001.

Zhi Li (S'00) received the B.Eng. degree from North China University of Technology, in 1997, and M.Eng. degree from Tsinghua University, Beijing, China, in 2000. He is currently working toward the Ph.D. degree in the Networks Laboratory, Department of Computer Science, University of California, Davis.

His research interests include computer networks, multicasting, Internet QoS, overlay network, and interdomain routing.

Prasant Mohapatra (S'88–M'89–SM'98) received the Ph.D. degree in computer engineering from the Pennsylvania State University, Philadelphia, in 1993.

He is currently a Professor in the Department of Computer Science, University of California, Davis. In the past, he was on the Faculty at Iowa State University, Ames, and Michigan State University, East Lansing. He has also held Visiting Scientist positions at Intel Corporation, Portland, OR, and Panasonic Technologies, Princeton, NJ. His research interests are in the areas of wireless mobile networks, Internet protocols and QoS, and Internet servers. His research has been funded though grants from the National Science Foundation, Intel Corporation, Panasonic Technologies, Hewlett Packard, and EMC Corporation.

Dr. Mohapatra was on the Editorial Board of the IEEE TRANSACTIONS ON COMPUTERS and has been on the program/organizational committees of several international conferences. He was the Program Chair for the PAWS Workshop during 2000–2001 and the Program Vice-Chair for the International Conference on Parallel Processing (2001) and INFOCOM 2004. He was also the Co-Editor of the January 2003 issue of the IEEE NETWORK.