

Inter Domain Policy Routing: Overview of Architecture and Protocols

Deborah Estrin and Martha Steenstrup¹

1 Introduction

The Open Routing Working Group of the IETF has developed an architecture for policy routing among administrative domains. The Inter-Domain Policy Routing (IDPR) architecture allows routing decisions to be made on the basis of administrative policy, in addition to connectivity and type of service.

This paper summarizes the key concepts and protocols developed. For more detailed discussion, we refer the reader to [11, 18, 2]. The IDPR protocols described were designed in collaboration with: H. Bowns, L. Breslau, I. Castineyra, D. Estrin, M. Lepp, T. Li, M. Little, K. Obraczka, M. Steenstrup, G. Tsudik, and R. Woodburn.

2 IDPR Basic Concepts

IDPR uses a link-state style routing protocol, along with source routing and explicit advertisement of policy and topology information. The source routes specify the administrative domains (ADs) that compose a source-destination path; the particular links and routers transited within ADs are transparent to the source and may change during the course of a session. Source routes are computed based on topology and policy information advertised by each participating transit AD (see definitions in Section 3. Policy information specifies which traffic the particular AD will carry, e.g., an AD can limit the types of service made available to particular endpoint ADs. The routing updates (topology and policy information) are advertised to **all** AD-level routing elements so that source routes can be computed appropriately.

Elsewhere the considerations that motivate the design of IDPR protocols are detailed[4, 11, 2, 6]. One of the key characteristics of IDPR is the use of source routing. Source routing, combined with explicit expression of policies in routing updates, allows transit ADs to apply source-specific policies and allows source ADs to apply local route selection criteria. Hop by hop protocols do not efficiently support the large set of policies and types of service that we anticipate will arise in the global inter-AD internet. For example, in order to support source-specific policies, hop by hop routing would

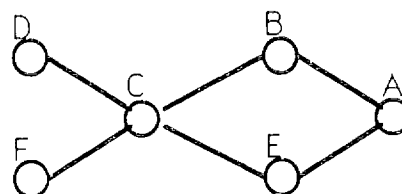


Figure 1: Example of policies that can not be supported with hop by hop routing without replicating routing tables. In this example, C can use B or E to get to A. However, D wants to use B, and F wants to use E. In hop by hop routing C must make a single decision if we assume a single routing table. Consequently, C's decision will deny either D or F a route. Examples of policies that could lead to this situation are: (1) B provides enhanced type of service support but charges on a usage sensitive basis while E does not charge but makes no service guarantees, (2) D is a university, F is a private corporation, and B is a subsidized gigabit network for university use only.

have to replicate routing tables for each possible source—a clearly infeasible solution. Figure 1 illustrates the need for source-specific policies and the inadequacy of hop by hop routing in this regard. In addition, source routing has the advantage of relaxing the consistency requirements for routing databases across the global internet.

When a packet is outgoing from one AD to another, and assuming an AD-level source route has been computed, a policy route is *setup* from the source AD to the destination AD (see Section 4.2). Successive packets between those two ADs, that meet the policy and type of service requirements specified in the setup packet, need not carry the full source route. Instead the successive packets are forwarded based upon a globally unique path ID and corresponding state information maintained in the AD boundary routers. In effect, the architecture allows routes to be installed *on demand* in

policy gateways through the setup function. For further discussion see [11, 18].

3 Terminology and Definitions

The following definitions and terminology are used throughout the remainder of the paper. Most of them are described in more detail in the subsequent sections.

- **Administrative Domain (AD)**: a collection of links, routers, and end-systems governed by a single administrative authority.[9].
- **Stub ADs**: contain the communicating end-systems that are the endpoints of communication; stub ADs do not provide transit services and need not advertise policies.
- **Transit ADs**: provide transit services and must advertise their policies.
- **Hybrid ADs** offer transit services (typically to fewer neighbor ADs), as well as end-system access.
- **Policy gateway (PG)**: router at the border of an AD that implements the IDPR protocol.
- **Virtual gateway (VG)**: a set of (at least two) physical policy gateways that form a connection between two neighboring ADs. Every VG has two "sides", one in each of the ADs that it connects; in its simplest form a VG consists of two PGs, one in each of the connecting ADs. The abstraction is introduced to reduce the amount of detailed (i.e., PG specific) status information that is distributed and maintained in the event that there are more than two PGs in a VG.
- **Route Server (RS)**: entity within each AD that collects topology and policy information from other ADs and computes Policy Routes (see below) based on this information.
- **Policy Term (PT)**: conditions specified by an AD for use of its transit resources. Policies may be in terms of endpoint ADs that are permitted to use the resource, type of service made available, or previous and next hop AD via which resources may be accessed, etc. For further discussion of policy types see [6]. Policy terms are carried in routing updates (see Update Protocol below).
- **Policy route (PR)**: the AD-level source route from source to destination AD. In particular, a PR is a sequence of VGs with the associated PTs that permit use of the respective AD resource.
- **Connectivity database**: the database of policy and topology information used by the Route Server as input to the route computation. The database includes configured topology and policy information, as well as most-recently reported status (i.e., up or down) of particular VGs.
- **Route database**: the database generated by route computation and kept by the route server. Policy gateways query the route server for routes from this database when outgoing packets arrive for which no PR exists.
- **Forwarding information base**: the database of active PRs maintained by PGs in order to forward packets along active PRs.
- **Setup and data protocols**: the protocols used to set up PRs and forward data packets along established PRs.
- **Update protocol**: distributes configured and status information about topology and policy. A link state protocol in which every AD advertises information about its neighbor connections to all ADs in the internet.²
- **Virtual gateway protocol**: carries information among the PGs in an AD to determine the status of VGs that should be advertised in the update protocol.
- **Configuration Server**: entity within each AD that distributes local (to the AD) configuration information to PGs and RSs. For example, the configuration server distributes the local PTs to the local PGs to be used in validation of setup requests.

The following section describes these elements in more detail.

4 Protocols

Three protocols form the core of the IDPR architecture: (1) virtual gateway protocol, (2) route setup and packet forwarding, and (3) update distribution of AD topology and policy information. A fourth function, route synthesis, is critical to the architecture but is not a protocol per se; each AD can implement its own version of route synthesis so long as it interfaces to the other protocols (e.g., update provides the inputs to route synthesis and route synthesis provides policy routes to setup). In addition to these four functions there are protocols for communication between the PG and RS, and the PG

²For most of this paper we assume a flat space of ADs across which all update information is flooded (i.e., no hierarchy).

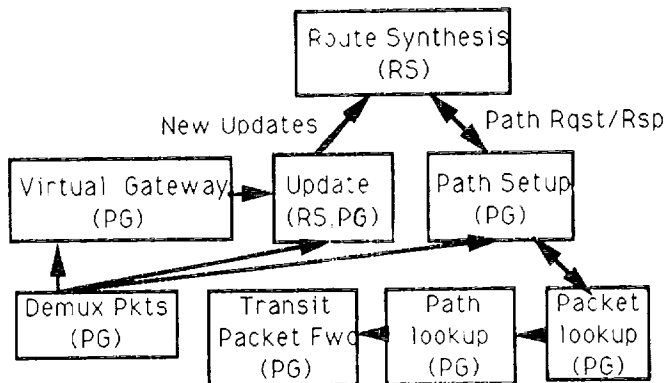


Figure 2: Simplified block diagram of the IDPR elements and their interaction.

and configuration server. The interaction of these elements is illustrated in Figure 2.

The virtual gateway, setup, and update protocols generate control packets that travel between PGs, generally in a hop-by-hop reliable fashion.³ The control information is encapsulated in the internet protocol used among PGs. Data packets travel with a smaller header indicating the path ID of the policy route being used. The operation of the individual protocols is summarized below.

4.1 Virtual Gateway Protocol

VGP is the protocol that allows groups of PGs to appear to the update and route synthesis processes as single entities. At a minimum a VG is comprised of two PGs, one in each of two directly-connected ADs. There may be more than one PG on either side of the VG, where a side refers to each AD.

VGP information has both global and local significance. The update protocol (described in Section 4.3) includes

VG reachability information garnered from VGP in dynamic routing updates; in this sense, VGP carries globally-relevant information. At the same time, policy

³The one exception to hop by hop handling of control packets arises when a route server requests an explicit update from someone other than its direct neighbor. In this case the update response is sent back from the responding RS to the requester and is not processed by neighbor PGs.

gateways within an AD use information collected from VGP to make local forwarding decisions, across a VG or across an AD. This detailed forwarding information is only distributed within the AD or VG and not to the whole internet. In addition, to the VGP derived information, each PG uses a simple up/down protocol to monitor reachability of directly connected PGs within a VG. PGs use their AD's internal IGP (or the PG up/down protocol if the internal IGP is not adequate) to monitor the reachability of other VGs (but only VGs in which the AD is a participant). VGP packets are encapsulated in the internet protocol used among PGs.

In summary, the information obtained from IGP and VGP is used by each PG to carry out setup and update procedures, as described in the rest of this section.

4.2 Packet Forwarding and Route Setup

The route setup protocol establishes a logical connection along the sequence of PGs that instantiate a policy route. Consequently, when an outbound packet arrives at a PG (i.e., the packet is exiting the AD to which the PG belongs), the source PG looks to see if the packet is part of a source-destination association that has already been bound to a policy route. In order for the association to match, it must have compatible type of service, user class identifiers, and other global conditions, in addition to the same source and destination addresses. If the association is bound, then it has already been assigned a unique path ID by the source PG. In this case, the packet is encapsulated in an IDPR data packet header, and the packet is forwarded to the next PG on the already established policy route. The header includes the path ID, which is comprised of the source AD, PG, and a unique number assigned by the source PG. If the packet is not part of a currently mapped association, the source PG compares the policy conditions of the packet with active policy routes. If it matches, then the association is added to the table and the packet is encapsulated as described above. In other words, many transport sessions may share a policy route, and consequently will use the same path ID. Finally, if no active policy route applies, the PG invokes the route setup function (see Section 5.3).

Route setup generates a setup request control packet which travels, hop-by-hop, along a sequence of PGs that correspond to the policy route. Packets are encapsulated in internet packets with the source address set to that of the sending PG and the destination address to that of the next PG. The policy route is included within the control packet so that each PG can make the next forwarding decision.

When a PG receives a setup request, it checks the

listed policy conditions and packet integrity.⁴ If all checks pass, the necessary state information is installed (e.g., path ID, next hop) and the setup packet is sent to the next PG in the policy route. The setup packet includes a policy route that lists VGs. VGP, described above, keeps each PG in a VG informed of which PGs are reachable within its VG, and which PGs represent the AD's other VGs. When a PG receives an outgoing packet (where outgoing means that it is exiting the AD to which the PG belongs), the PG must decide which PG on the other side of the VG to forward it to. When a PG receives an incoming packet, it must decide which PG in the next VG to send it to. In the first case the next PG is in a directly connected AD; in the second case the next PG is in the same AD as the sender. Information from the IGP is used to determine reachability of PGs within the same AD. Therefore, a policy route resembles a loose source route in two ways. First, the non-policy gateways traversed between next hop VGs are not bound before actual data packet transfer. In addition, even PGs are not bound until route setup time and can therefore adapt to PG up/down status, if VGs have redundant PGs.

The setup protocol serves to install routes on demand in the necessary PGs. Therefore it is well suited to an environment in which there are a large number of possible policies and associated policy routes. Such an environment makes hop-by-hop route table replication impractical and requires more global coordination. Unlike some connection-oriented protocols, these paths are shared among multiple transport sessions, and even among different host pairs, to avoid the latency of setup for transport sessions that exhibit some locality. Moreover, expedited data can be included in setup packets to accommodate datagram traffic that is traveling to uncommon destinations and that is not frequent enough to warrant maintaining state.

4.3 Update

The update protocol is a reliable link state update protocol similar to those described in [14, 16, 15]. As with setup packets, routing updates are sent from PG to PG in a hop by hop reliable manner.⁵ Update packets are encapsulated in the internet protocol used between PGs (e.g., within ADs).

The update packets include sequence numbers that allow duplicate detection. If the topology includes cycles a PG may receive an update more than once. The duplicate update is detected and dropped so that du-

plicates are not flooded further. Routing updates are not modified en route. Each update includes a signature (computed and appended in the originating AD and checked by the recipient update process within each AD) to provide integrity and authenticity.

Routing updates are generated by a representative PG in each transit AD (i.e., the lowest-numbered, live and reachable PG in the AD).⁶ The ultimate recipients of this information are the route servers in stub and hybrid ADs (i.e., any AD that must generate policy routes for traffic generated by end systems within the AD). There are two types of update messages. Configuration messages indicate the configured topology and policy information independent of the particular up/down status of each VG. Dynamic routing updates indicate which VGs are currently up, reachable, and therefore available for forwarding data packets. Configured routing updates are sent on a periodic basis, whereas dynamic updates are triggered by changes in VG status. Even dynamic updates are expected to be low frequency events (e.g., relative to intra-domain IGP routing). For example, when an intra-domain link goes down, we expect the AD's IGP to find an alternative path between any two PGs, thereby making the change transparent to IDPR. If a PR was computed with correct configured information, but without the benefit of up to date dynamic information, the setup (or data packet forwarding) process will fail with an appropriate error message alerting the originator of the setup; but no looping will occur because of the use of loop-free source routes.

Update packets express policy and topology information. One can think of the information in an update message as a listing of the policy conditions that are supported between each pair of the AD's VGs. However, assuming that many policies will be applied uniformly to a large subset of VGs in an AD, this would be inefficient. Therefore, each entry in an update is represented as a set of policy conditions that apply to the set of VGs listed. The VGs listed indicate connectivity to directly-connected neighbors; so all VGs listed in an update refer to VGs that connect to the AD generating the update. (This is analogous to a traditional link state update that advertizes information about a node's neighbors.)

The information needed to generate an update packet is collected from VGP, IGP, and the configuration server. VGP and IGP information allows the representative PG to determine which VGs are up and reachable. The con-

⁴The particular signature scheme used for integrity protection is indicated in the setup packet. For more information on the security mechanisms provided see [18, 8].

⁵However, in this case, retransmission of unreceived update packets may be over-ridden by availability of a new update packet that obsoletes the retransmitted information.

⁶In the absence of intra-AD partitions, there is one representative PG per AD. If a partition occurs then two PGs on either side will assume the PG representative role and receiving update processes will effectively treat them as separate ADs until the partition is healed. Update request and response messages are used to reconstitute a connectivity database after a partition. In the future, additional mechanisms for healing partitions may be included.

figuration server provides the PG with configured policy and topology information. The PG modifies the configured update to indicate which VGs are unreachable or down.

4.4 Route Synthesis

Route synthesis is not a protocol per se. It is the function that takes input from the update protocol in the form of global connectivity/policy database and computes AD level source routes. Route synthesis may be invoked on demand, i.e., when the first outbound packet in a transport session (for which there are no usable active PRs) arrives at the source PG. Alternatively, route synthesis may be invoked in advance to precompute some subset of routes that the AD expects to use (and thereby avoid the latency of on-demand computation). Whatever form of route computation is used, the input is the connectivity data base built with information obtained from Update. Given our assumptions about policy uniformity, we store the information as a collection of nodes that each represents either a policy condition or a VG. The output of route synthesis are policy routes and associated conditions of usage. The policy route is a sequence of VGs and the policies that allowed the VG to be used in the path. General conditions of use such as type of service, time of day, etc. are appended to the route information itself. This information is stored and provided to setup upon request.

When a route is computed on demand, the computation involves searching for a single route with specific characteristics. Therefore, much of the search tree can be pruned during computation. With precomputation, many paths are computed simultaneously and the size of the explored search tree is much larger. The actual size depends on the bounds placed on precomputation by the AD.

It is not computationally feasible to precompute all possible routes to all possible destinations in a large internet with variable policies. Other routing protocols limit the expressible or discoverable routes a priori by restricting each routing node to advertise only a single route (or single route per type of service) to each destination. IDPR is more expressive, and allows each stub AD to control which subset of routes it will discover/compute. Each stub AD is free to design or tailor route synthesis to its needs. In general we expect route synthesis to precompute some routes. Nevertheless, on-demand computation must always be supported to accommodate requirements not satisfied by precomputation (e.g., uncommon destinations, unusual TOS requirements, or unusual policy conditions). However, the problem with on demand computation for all routes is that the computation may be time consuming relative to acceptable setup latency limits, even when searching

for a route with very specific characteristics. Moreover, using on-demand computation alone is wasteful in terms of potentially unexploited parallelism. Both on-demand and precomputed routes are cached to take advantage of the locality exhibited by most communication environments. However, optimal strategies for managing these caches, in particular invalidation, deserve more careful analysis.

Because of the challenging nature of the route computation problem, further issues associated with route synthesis are the subject of ongoing research, as mentioned in Section 5.

5 Design and Implementation Issues

One area in which significant research is needed is in scaling of the IDPR architecture to very large networks (e.g., 50,000 ADs with 10% transits). The key scaling issues pertain to connectivity database size, update distribution, route synthesis, and policy gateway state.

5.1 Connectivity database

The connectivity database maintains global information and therefore raises concerns regarding scale. The size of the database depends upon AD connectivity, as well as policies. In particular, the route server that maintains the database faces memory requirements that are a function of the number of transit ADs, the average number of neighbors for a transit AD, the number of policies expressed, and the uniformity with which the policies are applied. For example, if we assumed that each of the 5,000 transit ADs includes an average of 10 policy term entries in an update, and an average of 50 VGs per transit AD, then approximately 2.5 Megabytes of storage is needed for the complete connectivity database. For further discussion, see [7].

A possible method for reducing connectivity database size is to introduce additional level(s) of abstraction or hierarchy, i.e., group ADs into super ADs and only flood update information among entities at the same level. However, in IDPR it is only meaningful to group ADs when policies are somewhat similar.

When ADs are grouped, the policies advertised to other AD groups must represent policies acceptable to constituent ADs. Alternatively, the AD group must advertise multiple VGs to each neighbor group in order to indicate the different policies supported. If intra-group policies differ significantly, then the AD group will have to advertise a large collection of policies for a large number of VGs and the amount of information distributed will not be reduced. Moreover, from the perspective

of logical connectivity (i.e., taking into account policy as well as topology), the more dissimilar the policies among constituent ADs, the more likely it is that the AD group will appear partitioned to external groups—an undesirable situation. Moreover, if other transit ADs (or groups) have policies that distinguish among members of an AD group, then those ADs will still write policies at the lower granularity and policy routes must be synthesized and set up with the finer granularity (i.e., not for the AD group as a whole).

5.2 Update distribution

Global flooding of update information also presents scaling problems. To analyze update overhead we need to make additional assumptions about the number of PGs in an AD and the number of inter-AD links supported by each of the PGs. For the purpose of this approximation we will assume 10 PGs per transit AD, where each PG is connected to 5 inter-AD links and each AD generates approximately one update per day. Although under these assumptions only 5,000 routing updates are generated per day, a PG may receive duplicates because of the flooding technique used. Each PG will receive at most one copy of each update from each of the PGs to which it is connected in neighboring ADs (5 in this example). In the worst case each PG also receives one copy of the update from each of the other PGs in its own AD (9 in this example). Therefore, in this pessimistic example, each PG would see at most 14 duplicates of each new update—averaging to 50 per minute or less than 1 per second. This represents an aggregate of approximately 3 Kbytes per second of data over intra-AD resources. However, because internetwork structure more closely resembles a tree than a mesh, the general case should be much better. In other words, one PG in one VG typically will receive an update before any others in a given AD. Consequently, an AD would not experience updates from all VGs simultaneously and duplicates would be eliminated.⁷

A possible method for reducing update distribution overhead is to limit the hop count on flooded packets. However, this assumes a topological locality of traffic that is not necessarily characteristic of much internet traffic. Consequently, we are considering use of mechanisms for distribution of routing updates within logical communities.

5.3 Route computation

As described earlier, complete precomputation of policy routes would be computationally infeasible in a large

⁷We note that even the worst case analysis results in a tolerable number considering that the burden is spread across the AD as a whole, not carried over a single intra-AD link.

internet with multiple policies and types of service per AD. However, we wish to exploit the parallelism available when computing multiple routes simultaneously, and to reduce setup latency, by doing some precomputation

On demand computation can use a simple breadth first search to locate a policy route that meets the conditions specified by setup. SPF algorithms such as [5] may be used instead if some aspect(s) of policy are encoded in metrics before computation.

The same approach can be used for precomputation, however, multiple paths are explored and few paths are pruned. We are exploring the use of several techniques to address this problem. One approach is to precompute K best paths based on topology (e.g., hop count) alone, and consequently check the policies of each route's constituent ADs to eliminate those that are not legal. This approach will fail if K is too small to discover usable or desirable routes. Alternatively, if K is set too large, it will not be computationally practical.

Other heuristics may be incorporated into route synthesis. For example the search tree may be pruned by rejecting ADs that express excessively restrictive or unusual policies (e.g., a TOS that is rarely used, or a user-class specific policy that will be applicable only rarely). Alternatively, an AD may precompute all or many paths with a small number of hops and extend them on demand when appropriate paths to desired destinations can not be found.

This is an area of ongoing research. Currently the internet is probably small enough to support with on demand computation alone.

5.4 PG state

As part of the setup protocol there is state information established and maintained in each PG along a policy route. The state information is the forwarding information base used to forward data packet; consequently it can not be treated as "soft state" as defined by Clark.[3].

The amount of state information maintained by a PG depends upon where it sits in the internet and how much traffic it sees. The memory requirements to store and maintain this state are significant for the PGs in very large transit ADs. Moreover, the more fine grained the policies expressed by transits and stub ADs, the larger the number of policy routes that individual stubs will set up, and the greater the amount of state required. Caching strategies should be effective given some locality of communication. We are currently collecting data to determine the extent of locality experienced in existing internet backbones, and are analyzing the utility of various caching schemes.

5.5 Other Design and Implementation Issues

In addition to issues of scaling, the IDPR architecture raises several issues and challenges to current datagram internet design and practice: (1) route setup introduces state requirements in border gateways, (2) VG abstraction is used to represent groups of PGs, (3) The gateway and route synthesis processes are logically (and often physically) separated, and (4) encapsulation is used to tunnel through ADs.

6 Conclusions and status

In summary, the IDPR architecture is based on source routing, setup and link state advertisements with explicit policy terms. This approach supports flexible expression of policy with limited computational burden on transit ADs and a means of avoiding routing loops without strong consistency requirements.

The IDPR protocols are currently being implemented and we began testing of the partial prototype in October of 1990. Participants are BBN, USC, and SAIC.

We will be conducting experiments in the laboratory throughout the fall and hope to experiment with multiple sites in early December (e.g., via DARTnet). In parallel we are investigating the various scaling issues discussed through analysis and simulation.

Future versions of the protocol will separate the RS and PG functions, explore hierarchical configuration for the connectivity database in RSs, and address the need for a community update mechanism, techniques for route synthesis, and interoperability with other (lower overhead and lower functionality) inter-AD protocols, such as BGP.[13, 1]

References

- [1] ANSI, *Intermediate System to Intermediate System Inter-domain Routing Information Exchange Protocol*, Document Number X3S3.3/90-132, June 1990.
- [2] L. Breslau and D. Estrin, *Design of Inter-Administrative Domain Routing Protocols*, ACM SIGCOMM, September 1990. Philadelphia, PA.
- [3] D. Clark, *Design Philosophy of the DARPA Internet Protocols*, Proceedings of ACM Sigcomm, pp. 106-114, August 1988, Palo Alto, CA.
- [4] D. Clark, *Policy Routing in Internet Protocols*, RFC 1102, SRI Network Information Center, May 1989.
- [5] E. W. Dijkstra, *A Note on Two Problems in Connection with Graphs*, Numerische Mathematik, 1959, pp 269-271.
- [6] D. Estrin, *Policy Requirements for Inter Administrative Domain Routing*, To appear in *Computer Networks and ISDN Systems*, 1991. A previous version was distributed as RFC 1125, SRI Network Information Center, November 1989.
- [7] D. Estrin, K. Obraczka, *Connectivity Database Overhead for Inter-Domain Policy Routing*, University of Southern California, Technical Report TR 90-17, August 1990. To appear in Proceedings of IEEE Infocom 1991.
- [8] D. Estrin and G. Tsudik, *Secure Control of Inter-network Transit Traffic*, To Appear in *Computer Networks and ISDN Systems*, 1991.
- [9] S. Hares, D. Katz, *Administrative Domains and Routing Domains, a Model for Routing in the Internet*, RFC 1136, SRI Network Information Center, December 1989.
- [10] ISO, *OSI Routing Framework*, ISO/TF 9575, 1989.
- [11] M. Lepp and M. Steenstrup. *An Architecture for Inter-Domain Policy Routing* BBN Technical Report No. 7345, July 1990.
- [12] M. Little, *Goals and Functional Requirements for Inter-Autonomous System Routing*, RFC 1126, SRI Network Information Center, October 1989.
- [13] K. Lougheed and Y. Rekhter, *Border Gateway Protocol*, RFC 1163, SRI Network Information Center, June 1990.

- [14] J.M. McQuillan , I. Richer, and E. Rosen, *The New Routing Algorithm for the ARPANET*, **IEEE Transactions on Communications**, Volume COM-28:711-719, 1980.
- [15] J. Moy, *The Open Shortest Path First (OSPF) Specification*, **RFC 1131**, **SRI Network Information Center**, October 1989.
- [16] R. Perlman, *Fault-Tolerant Broadcast of Routing Information*, **Computer Networks**, Volume 7, December 1983, pp 395-405. .
- [17] E. Rosen, *Exterior Gateway Protocol (EGP)*, **RFC 827**, **SRI Network Information Center**, October 1982.
- [18] M. Steenstrup, *Inter-Domain Policy Routing Protocol Specification and Usage: Version 1*, **BBN Technical Report No. 7346**, July 1990.