Inter-Domain Policy Routing Working Group
Internet Draft
June 1992

M. Lepp and M. Steenstrup
BBN Systems and Technologies
Expires 15 December 1992

An Architecture for Inter-Domain Policy Routing

Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress".

Please check the 1id-abstracts.txt listing contained in the internet-drafts Shadow Directories on nic.ddn.mil, nnsc.nsf.net, nic.nordu.net, ftp.nisc.sri.com, or munnari.oz.au to learn the current status of any Internet Draft.

This Internet Draft will be submitted to the RFC editor as an architecture specification. Distribution of this Internet Draft is unlimited. Please send comments to idpr-wg@bbn.com.

Abstract

We present an architecture for inter-domain policy routing (IDPR). The objective of IDPR is to construct and maintain routes, between source and destination administrative domains, that provide user traffic with the requested services within the constraints stipulated for the domains transited. The IDPR architecture is designed to accommodate an internetwork containing tens of thousands of administrative domains with heterogeneous service requirements and restrictions.

Contributors

The following people have contributed to the IDPR architecture: Bob Braden, Lee Breslau, Ross Callon, Noel Chiappa, Dave Clark, Pat Clark, Deborah Estrin, Marianne Lepp, Mike Little, Martha Steenstrup, Zaw-Sing Su, Paul Tsuchiya, and Gene Tsudik. Yakov Rehkter supplied many useful comments on a previous draft of this document.

Contents

1 Introduction **1**

1.1 The Internet Environment 2

2 Approaches to Policy Routing **3**

2.1 Policy Route Generation 3

2.1.1 Distance Vector Approach 3

2.1.2 Link State Approach 5

2.2 Routing Information Distribution 5

2.2.1 Distance Vector Approach 6

2.2.2 Link State Approach 7

2.3 Message Forwarding along Policy Routes 8

2.3.1 Hop-by-Hop Approach 8

2.3.2 Source Specified Approach 9

3 The IDPR Architecture **11**

3.1 IDPR Functions 11

3.2 IDPR Entities 11

3.2.1 Path Agents 13

3.2.2 IDPR Servers 14

3.2.3 Entity Identifiers 16

3.3 Security and Reliability 17

3.3.1 Retransmissions and Acknowledgements 17

3.3.2 Integrity Checks 17

3.3.3 Source Authentication 18

3.3.4 Timestamps 18

3.4 An Example of IDPR Operation 19

4 Accommodating a Large, Heterogeneous Internet **22**

4.1 Domain Level Routing 22

4.2 Route Generation 23

4.3 Super Domains 24

4.4 Domain Communities 26

4.5 Robustness in the Presence of Failures 26

4.5.1 Path Repair 27

4.5.2 Partitions 28

1 Introduction

As data communications technologies evolve and user populations grow, the demand for internetworking increases. Internetworks usually proliferate through interconnection of autonomous, heterogeneous networks administered by separate authorities. We use the term *administrative domain* (AD) to refer to any collection of contiguous networks, gateways, links, and hosts governed by a single administrative authority who selects the intra-domain routing procedures and addressing schemes, defines service requirements for locally-generated traffic, and specifies service restrictions for transit traffic.

Interconnection of administrative domains can broaden the range of services available in an internetwork. Hence, traffic with special service requirements is more likely to receive the service requested. However, administrators of domains offering special transit services are more likely to establish stringent access restrictions, in order to maintain control over the use of their domains' resources.

An internetwork composed of many domains with diverse service requirements and restrictions requires policy routing to transport traffic between source and destination. *Policy routing* constitutes route generation and message forwarding procedures for producing and using routes that simultaneously satisfy user service requirements and respect transit domain service restrictions.

With policy routing, each domain administrator sets *transit policies* that dictate how and by whom the resources within its domain should be used. Transit policies are usually public, and they specify offered services comprising:

Access restrictions: e.g., applied to traffic to or from certain domains or classes of users.

Quality: e.g., delay, throughput, or error characteristics.

Monetary cost: e.g., charge per byte, message, or unit time.

Each domain administrator also sets *source policies* for traffic originating within its domain. Source policies are usually private, and they specify requested services comprising:

Access restrictions: e.g., domains to favor or avoid in routes.

Quality: e.g., acceptable delay, throughput, or reliability.

Monetary cost: e.g., acceptable session cost.

In this document, we describe an architecture for inter-domain policy routing (IDPR), and we provide a set of functions which can form the basis for a suite of IDPR protocols.

1.1 The Internet Environment

The Internet currently comprises over 4000 operational networks and over 10,000 registered networks. In fact, for the last several years, the number of constituent networks has approximately doubled annually. Although we do not expect the Internet to sustain this growth rate, we must provide an architecture for IDPR that can accommodate the Internet five to ten years in the future. According to the functional requirements for inter-autonomous system (i.e., inter-domain) routing set forth in [6], the IDPR architecture and protocols must be able to handle $O(10^5)$ networks distributed over $O(10^4)$ domains.

Internet connectivity has increased along with the number of component networks. In the early 1980s, the Internet was purely hierarchical, with the ARPANET as the single backbone. The current Internet possesses a semblance of a hierarchy in the collection of backbone, regional, metropolitan, and campus domains that compose it. However, technological, economical, and political incentives have prompted the introduction of inter-domain links outside of those in the strict hierarchy. Hence, the Internet has the properties of both hierarchical and mesh connectivity.

We expect that the Internet will evolve in the following way. Over the next five years, the Internet will grow to contain $O(10)$ backbone domains, most providing connectivity between many source and destination domains and offering a wide range of qualities of service, for a fee. Most domains will connect directly or indirectly to at least one Internet backbone domain, in order to communicate with other domains. In addition, some domains may install direct links to their most favored destinations. Domains at the lower levels of the hierarchy will provide some transit service, limited to traffic between selected sources and destinations. However, the majority of Internet domains will be stubs, that is, domains that do not provide any transit service for other domains. The bulk of Internet traffic will be generated by hosts in these stub domains, and thus, the applications running in these hosts will determine the traffic service requirements. We expect application diversity encompassing electronic mail, desktop videoconferencing, scientific visualization, and distributed simulation, to list a few. Many of these applications have strict requirements on delivery, delay, and throughput.

Ensuring that Internet traffic traverses routes that provide the required services without violating domain usage restrictions will be the task of policy routing in the Internet in the next several years. Refer to [1]-[10] for more information on the role of policy routing in the Internet.

2 Approaches to Policy Routing

In this section, we provide an assessment of candidate approaches to policy routing, concentrating on the distance vector and link state alternatives for routing information distribution and route generation and on the hop-by-hop and source specified alternatives for data message forwarding. The IDPR architecture supports link state routing information distribution and route generation in conjunction with source specified message forwarding. We justify these choices for IDPR below.

2.1 Policy Route Generation

We present policy route generation from the distance vector perspective and from the link state perspective.

2.1.1 Distance Vector Approach

Distance vector route generation distributes the computation of a single route among multiple routing entities along the route. Hence, distance vector route generation is potentially susceptible to the problems of routing loop formation and slow adaptation to changes in an internetwork. However, there exist several techniques that can be applied during distance vector route generation to reduce the severity of, or even eliminate, these problems. For information on a loop-free, quickly adapting distance vector routing procedure, consult [13] and [14].

During policy route generation, each recipient of a distance vector message assesses the acceptability of the associated route and determines the set of neighboring domains to which the message should be propagated. In the context of policy routing, both of the following conditions are necessary for route acceptability:

1. The route is consistent with at least one transit policy for each domain, not including the current routing entity's domain, contained in the route. To enable each recipient of a distance vector message to verify consistency of the associated route with the transit policies of all constituent domains, each routing entity should include its domain's identity and transit policies in each acceptable distance vector message it propagates.
2. The route is consistent with at least one source policy of at least one domain in the Internet. To enable each recipient of a distance vector message to verify consistency of the associated route with the source policies of particular domains, each domain must provide other domains with access to its source policies.

In addition, at least one of the following conditions is necessary for route acceptability:

1. The route is consistent with at least one of the transit policies of the current routing entity's domain. In this case, the routing entity accepts the distance vector message and then proceeds to compare the associated route with its other routes for the given destinations. If the routing entity decides that the new route is preferable, it updates the distance vector message with its domain's identity and transit policies and then propagates the message to the appropriate neighboring domains. We discuss distance vector message distribution in more detail in section 2.2.1.

2. The route is consistent with at least one of the source policies of the current routing entity's domain. In this case, the routing entity need not propagate the distance vector message but does retain the associated route for use by traffic, from local hosts, bound for the given destinations.

The routing entity discards any distance vector message that does not meet these necessary conditions.

With distance vector policy route generation, a routing entity may select and store multiple routes of different characteristics, such as qualities of service, to a single destination. A routing entity uses the quality of service information provided in the transit policies contained in accepted distance vector messages to discriminate between routes based on quality of service. Moreover, a routing entity may select routes that are specific to certain source domains, provided that the routing entity has access to the source policies of those domains.

In the distance vector context, the flexibility of policy route generation afforded by accounting for other domains' transit and source policies in route selection has the following disadvantages:

1. Each recipient of a distance vector message must bear the cost of verifying the consistency of the associated route with the constituent domains' transit policies.

2. Source policies must be made public. Thus, a domain must divulge potentially private information.

3. Each recipient of a distance vector message must bear the potentially high costs of selecting routes for arbitrary source domains. In particular, a routing entity must store the source policies of other domains, account for these source policies during route selection, and maintain source-specific forwarding information. Moreover, there must be a mechanism for distributing source policy information among domains. Depending on the mechanism selected, distribution of source policies may add to the costs paid be each routing entity in supporting source-specific routing.

We note, however, that failure to distribute source policies to all domains may have unfortunate consequences. In the worst case, a domain may not learn of any acceptable routes to a given destination, even though acceptable routes do exist. For example, suppose that $AD\ V$ is connected to $AD\ W$ and that $AD\ W$ can reach $AD\ Z$ through either $AD\ X$ or $AD\ Y$. Suppose also that $AD\ W$, as a recipient of distance vector messages originating in $AD\ Z$, prefers the route through $AD\ Y$ to the route through $AD\ X$. Furthermore, suppose that $AD\ W$ has no knowledge of $AD\ V$'s source policy precluding traffic from traversing $AD\ Y$. Hence, $AD\ W$ distributes to $AD\ V$ the distance vector message containing the route WYZ but not the distance vector message containing the route WXZ. $AD\ V$ is thus left with no known route to $AD\ Z$, although a viable route traversing $AD\ W$ and $AD\ X$ does exist.

2.1.2 Link State Approach

Link state route generation permits concentration of the computation of a single route within a single routing entity at the source of the route. In the policy routing context, entities within a domain generate link state messages containing information about the originating domain, including the set of transit policies that apply and the connectivity to adjacent domains, and they distribute these messages to neighboring domains. Each recipient of a link state message stores the routing information for anticipated policy route generation and also distributes it to neighboring domains. Based on the set of link state messages collected from other domains and on its domain's source and transit policies, a routing entity constructs and selects policy routes from its domain to other domains in the Internet.

We have selected link state policy route generation for IDPR for the following reasons:

1. Each domain has complete control over policy route generation from the perspective of itself as source.

2. The cost of computing a route is completely contained within the source domain. Hence, routing entities in other domains need not bear the cost of generating policy routes that their domain's local hosts may never use.

3. Source policies may be kept private and hence need not be distributed. Thus, there are no memory, computation, or link bandwidth costs incurred for distributing and storing source policies.

2.2 Routing Information Distribution

A domain's routing information and the set of domains to which that routing information is distributed each influence the set of generable policy routes that include the given domain. In particular, a domain administrator may promote the generation of routes that obey its domain's transit policies by ensuring that its domain's routing information:

1. Includes resource access restrictions.

2. Is distributed only to those domains that are permitted to use the resources.

Both of these mechanisms, distributing restrictions with and restricting distribution of a domain's routing information, can be applied in both the distance vector and link state contexts.

2.2.1 Distance Vector Approach

A routing entity may distribute its domain's resource access restrictions by including the appropriate transit policy information in each distance vector it accepts and propagates. Also, the routing entity may restrict distribution of an accepted distance vector message by limiting the set of neighboring domains to which it propagates the message. In fact, restricting distribution of routing information is inherent in the distance vector approach, as a routing entity propagates only the preferred routes among all the distance vector messages that it accepts.

Although restricting distribution of distance vector messages is easy, coordinating restricted distribution among domains may not be. Each domain may have a set of distribution restrictions that apply to all distance vector messages generated by that domain as well as sets of distribution restrictions that apply to distance vector messages generated by other domains.

As a distance vector message propagates among domains, each routing entity should exercise the distribution restrictions associated with each domain constituting the route thus far constructed. In particular, a routing entity should send an accepted distance vector message to a given neighbor, only if distribution of that message to that neighbor is not precluded by any domain contained in the route.

To enable a routing entity to exercise these distribution restrictions, each domain must permit other domains access to its routing information distribution restrictions. However, we expect that domains will prefer to keep distribution restictions, like source policies, private. There are at least two ways to make a domain's routing information distribution restrictions generally available to other domains:

1. Prior to propagation of an accepted distance vector message, a routing entity includes in the message its domain's distribution restrictions (all or only those to that apply to the given message). This method requires no additional protocol for disseminating the distribution restrictions, but it may significantly increase the size of and the processing required for each distance vector message.

2. Each domain independently disseminates its distribution restrictions to all other domains, so that each domain will be able to exercise all other domains' distribution restrictions. This method

requires an additional protocol for disseminating the distribution restrictions, and it may require a significant amount of memory at each routing entity for storing all domains' distribution restrictions.

We note that a domain administrator may describe the optimal distribution pattern of distance vector messages originating in its domain, as a directed graph rooted at its domain. Furthermore, if all domains in the directed graph honor the directionality and if the graph is also acyclic, no routing loops may form, because no two domains are able to exchange distance vector messages pertaining to the same destination. However, an acyclic graph also means that some domains may be unable to discover alternate paths when connectivity between adjacent domains fails, as we show below.

We reconsider the example from section 2.1.1. Suppose that the distance vector distribution graph for $AD\ Z$ is such that all distance vectors originating in $AD\ Z$ flow toward $AD\ V$. In particular, distance vectors from $AD\ Z$ enter $AD\ W$ from $AD\ X$ and $AD\ Y$ and leave $AD\ W$ for $AD\ V$. Now, suppose that the link between the $AD\ Z$ and $AD\ X$ breaks. $AD\ X$ no longer has knowledge of any viable route to $AD\ Z$, although such a route exists through $AD\ W$. To ensure discovery of alternate routes during connectivity failures, the distance vector distribution graph for $AD\ Z$ must contain bidirectional links between $AD\ W$ and $AD\ X$ and between $AD\ W$ and $AD\ Y$.

2.2.2 Link State Approach

With link state routing information distribution, all recipients of a domain's link state message gain knowledge of that domain's transit policies and hence service restrictions. For reasons of efficiency or privacy, a domain may also restrict the set of domains to which its link state messages should be distributed. Thus, a domain has complete control over distributing restrictions with and restricting distribution of its routing information.

A domain's link state messages automatically travel to all other domains if no distribution restrictions are imposed. Moreover, to ensure that distribution restrictions, when imposed, are applied, the domain may use source specified forwarding of its link state messages, such that the messages are distributed and interpreted only by the destination domains for which they were intended. Thus, only those domains receive the given domain's link state messages and hence gain knowledge of that domain's service offerings.

We have selected link state routing information distribution for IDPR for the following reasons:

1. A domain has complete control over the distribution of its own routing information.
2. Routing information distribution restrictions may be kept private and hence need not be distributed. Thus, there are no memory, computation, or link bandwidth costs incurred for distributing and

storing distribution restrictions.

2.3 Message Forwarding along Policy Routes

To transport data messages along a selected policy route, a routing entity may use either hop-by-hop or source specified message forwarding.

2.3.1 Hop-by-Hop Approach

With hop-by-hop message forwarding, each routing entity makes an independent forwarding decision based on a message's source, destination, and requested services and on information contained in the entity's forwarding information database. Hop-by-hop message forwarding follows a source-selected policy route only if all routing entities along the route have consistent routing information and make consistent use of this information when generating and selecting policy routes and when establishing forwarding information. In particular, all domains along the route must have consistent information about the source domain's source policies and consistent, but not necessarily complete, information about transit policies and domain adjacencies within the Internet. In general, this implies that each domain should have knowledge of all other domains' source policies, transit policies, and domain adjacencies.

When hop-by-hop message forwarding is applied in the presence of inconsistent routing information, the actual route traversed by data messages not only may differ from the route selected by the source but also may contain loops. In the policy routing context, private source policies and restricted distribution of routing information are two potential causes of routing information inconsistencies among domains. Moreover, we expect routing information inconsistencies among domains in a large Internet, independent of whether the Internet supports policy routing, as some domains may not want or be able to store routing information from the entire Internet.

A Clarification. In a previous draft, we presented the following example which results in persistent routing loops, when hop-by-hop message forwarding is used in conjunction with distance vector routing information distribution and route selection. Consider the sequence of events:

1. $AD\ X$ receives a distance vector message containing a route to $AD\ Z$, which does not include $AD\ Y$. $AD\ X$ selects and distributes this route as its primary route to $AD\ Z$.

2. $AD\ Y$ receives a distance vector message containing a route to $AD\ Z$, which does not include $AD\ X$. $AD\ Y$ selects and distributes this route as its primary route to $AD\ Z$.

3. $AD\ X$ eventually receives the distance vector message containing the route to $AD\ Z$, which includes $AD\ Y$ but not $AD\ X$. $AD\ X$ prefers this route over its previous route to $AD\ Z$ and selects this new route as its primary route to $AD\ Z$.

4. $AD\ Y$ eventually receives the distance vector message containing the route to $AD\ Z$, which includes $AD\ X$ but not $AD\ Y$. $AD\ Y$ prefers this route over its previous route to $AD\ Z$ and selects this new route as its primary route to $AD\ Z$.

Thus, $AD\ X$ selects a route to $AD\ Z$ that includes $AD\ Y$, and $AD\ Y$ selects a route to $AD\ Z$ that includes $AD\ X$.

Suppose that all domains along the route selected by $AD\ X$, except for $AD\ Y$, make forwarding decisions consistent with $AD\ X$'s route, and that all domains along the route selected by $AD\ Y$, except for $AD\ X$, make forwarding decisions consistent with $AD\ Y$'s route. Neither $AD\ X$'s selected route nor $AD\ Y$'s selected route contains a loop. Nevertheless, data messages destined for $AD\ Z$ and forwarded to either $AD\ X$ or $AD\ Y$ will continue to circulate between $AD\ X$ and $AD\ Y$, until there is a route change. The reason is that $AD\ X$ and $AD\ Y$ have conflicting notions of the route to $AD\ Z$, with each domain existing as a hop on the other's route.

We note that while BGP-3 [8] is susceptible to such routing loops, BGP-4 [9] is not. We thank Tony Li and Yakov Rekhter for their help in clarifying this difference between BGP-3 and BGP-4.

2.3.2 Source Specified Approach

With source specified message forwarding, the source domain dictates the data message forwarding decisions to the routing entities in each intermediate domain, which then forward data messages according to the source specification. Thus, the source domain ensures that any data message originating within it follows its selected routes.

For source specified message forwarding, each data message must carry either an entire source specified route or a path identifier. Including the complete route in each data message incurs a per message transmission and processing cost for transporting and interpreting the source route. Using path identifiers does not incur these costs. However, to use path identifiers, the source domain must initiate, prior to data message forwarding, a path setup procedure that forms an association between the path identifier and the next hop, in the routing entities in each domain along the path. Thus, path setup may impose an initial delay before data message forwarding can begin.

We have selected source specified message forwarding for IDPR data messages for the following reasons:

1. Message forwarding respects the source policies of the source domain, regardless of whether intermediate domains along the route have knowledge of these source policies.

2. Message forwarding is loop-free, regardless of whether the all domains along the route maintain consistent routing information.

Also, we have chosen path identifiers over complete routes, to affect source specified message forwarding, because of the reduced transmission and processing cost per data message.

3 The IDPR Architecture

We now present the architecture for IDPR, including a description of the IDPR functions, the entities that perform these functions, and the features of IDPR that aid in accommodating Internet growth.

3.1 IDPR Functions

Inter-domain policy routing comprises the following functions:

1. Collecting and distributing routing information including domain transit policies and inter-domain connectivity.
2. Generating and selecting policy routes based on the routing information distributed and on the source policies configured or requested.
3. Setting up paths across the Internet using the policy routes generated.
4. Forwarding messages across and between domains along the established paths.
5. Maintaining databases of routing information, inter-domain policy routes, forwarding information, and configuration information.

3.2 IDPR Entities

From the perspective of IDPR, the Internet comprises administrative domains connected by virtual gateways, which are in turn connected by intra-domain routes supporting the transit policies configured by the domain administrators. Each domain administrator defines the set of transit policies that apply across its domain and the virtual gateways between which each transit policy applies. Several different transit policies may be configured for the intra-domain routes connecting a pair of virtual gateways. Moreover, a transit policy between two virtual gateways may be directional. That is, the transit policy may apply to traffic flowing in one direction, between the virtual gateways, but not in the other direction.

Virtual gateways (VGs) are the only connecting points recognized by IDPR between adjacent administrative domains. Each virtual gateway is actually a collection of directly-connected policy gateways (see below) in two adjacent domains, whose existence has been sanctioned by the administrators of both domains. Domain administrators may agree to establish more than one virtual gateway between their domains. For example, if two domains are to be connected at two geographically distant locations, the domain administrators may wish to preserve these connecting points as distinct at the inter-domain level, by establishing two distinct virtual gateways.

Policy gateways (PGs) are the physical gateways within a virtual gateway. Each policy gateway forwards transit traffic according to the service restrictions stipulated by its domain's transit policies applicable to its virtual gateway. A single policy gateway may belong to multiple virtual gateways. Within a domain, two policy gateways are *neighbors* if they are in different virtual gateways. Within a virtual gateway, two policy gateways are *peers* if they are in the same domain and are *adjacent* if they are in different domains. Peer policy gateways must be able to communicate over intra-domain routes that support the transit policies that apply to their virtual gateways. Adjacent policy gateways are *directly connected* if they are the only Internet addressable entities attached to the connecting medium. Note that this definition implies that not only point-to-point links but also multiaccess networks may serve as direct connections between adjacent policy gateways.

Combining multiple policy gateways into a single virtual gateway affords three advantages:

1. A reduction in the amount of IDPR routing information that must be distributed and maintained throughout the Internet.
2. An increase in the reliability of IDPR routes through redundancy of physical connections between domains.
3. An opportunity for load sharing of IDPR traffic among policy gateways.

Several different entities are responsible for performing the IDPR functions:

1. Policy gateways collect and distribute routing information, participate in path setup, forward data messages along established paths, and maintain forwarding information databases.
2. *Path agents* act on behalf of hosts to select policy routes, to set up and manage paths, and to maintain forwarding information databases.
3. Special-purpose servers maintain all other IDPR databases as follows:
   (a) Each *route server* is responsible for both its database of routing information, including domain connectivity and transit policy information, and its database of policy routes. Also, each route server generates policy routes on behalf of its domain, using entries from its routing information database and source policy information supplied through configuration or obtained directly from the path agents.
   (b) Each *mapping server* is responsible for its database of mappings that resolve Internet names and addresses to administrative domains.
   (c) Each *configuration server* is responsible for its database of configured information that applies to policy gateways, path agents, and route servers in the given administrative domain. The

configuration information for a given domain includes source and transit policies and mappings between local IDPR entities and their Internet addresses.

To maximize IDPR's manageability, one should embed all of IDPR's required functionality within the IDPR protocols and procedures. However, to minimize duplication of implementation effort, one should take advantage of required functionality already provided by mechanisms external to IDPR. Two such cases are the mapping server functionality and the configuration server functionality. The functions of the mapping server can be integrated into an existing name service such as DNS, and the functions of the configuration server can be integrated into the domain's existing network management system.

Within the Internet, only policy gateways, path agents, and route servers must be able to generate, recognize, and process IDPR messages. The existence of IDPR is invisible to all other gateways and hosts. Mapping servers and configuration servers perform necessary but ancillary functions for IDPR, and they are not required to execute the IDPR protocols.

3.2.1 Path Agents

Any Internet host can reap the benefits of IDPR, as long as there exists a path agent configured to act on its behalf and a means by which the host's messages can reach that path agent. Path agents select and set up policy routes for hosts, accounting for service requirements. To obtain a host's service requirements, a path agent may either consult its IDPR source policy configuration information or extract service requirements directly from the host's data messages, provided such information is available in these data messages.

Separating the path agent functions from the hosts means that host software need not be modified to support IDPR. Moreover, it means that a path agent can aggregate onto a single policy route traffic from several different hosts, as long as the source domains, destination domains, and service requirements are the same for all of these host traffic flows. Policy gateways are the natural choice for the entities that perform the path agent functions on behalf of hosts, as policy gateways are the only inter-domain connecting points recognized by IDPR.

Each domain administrator determines the set of hosts that its domain's path agents will handle. We expect that a domain administrator will normally configure path agents in its domain to act on behalf of its domain's hosts only. However, a path agent can be configured to act on behalf of any Internet host. This flexibility permits one domain to act as an IDPR proxy for another domain. For example, a small stub domain may wish to have policy routing available to a few of its hosts but may not want to set up its domain to support all of the IDPR functionality. The administrator of the stub domain can negotiate the proxy function with the administrator of another domain, who agrees that its domain will provide policy routes on behalf of the stub domain's hosts.

If the source domain supports IDPR and limits all domain egress points to policy gateways, then each message generated by a host in the source domain and destined for a host in another domain must pass through at least one policy gateway, and hence path agent, in the source domain. A host need not know how to reach any policy gateways in its domain; it need only know how to reach a gateway on its own local network. Gateways within the source domain direct inter-domain host traffic toward policy gateways, using default routes or routes derived from other inter-domain routing procedures.

If the source domain does not support IDPR and requires an IDPR proxy domain to provide its hosts with policy routing, the administrator of the source domain must carefully choose the proxy domain. All intervening gateways between hosts in the source domain and path agents in the proxy domain forward traffic according to default routes or routes derived from other inter-domain routing procedures. In order for traffic from hosts in the source domain to reach the proxy domain with no special intervention, the proxy domain must lie on an existing non-IDPR inter-domain route from the source to the destination domain. Hence, to minimize the knowledge a domain administrator must have about inter-domain routes when selecting a proxy domain, we recommend that a domain administrator select its proxy domain from the set of adjacent domains.

In either case, the first policy gateway to receive messages from an inter-domain traffic flow originating at the source domain acts as the path agent for the host generating that flow.

3.2.2 IDPR Servers

IDPR servers are the entities that manage the IDPR databases and that respond to queries for information from policy gateways or other servers. Each IDPR server may be a dedicated device, physically separate from the policy gateway, or it may be part of the functionality of the policy gateway itself. Separating the server functions from the policy gateways reduces the processing and memory requirements for and increases the data traffic carrying capacity of the policy gateways.

The following IDPR databases: routing information, route, mapping, and configuration, may be distributed hierarchically, with partial redundancy throughout the Internet. This arrangement implies a hierarchy of the associated servers, where a server's position in the hierarchy determines the extent of its database. At the bottom of the hierarchy are the *local servers* that maintain information pertinent to a single domain; at the top of the hierarchy are the *global servers* that maintain information pertinent to all domains in the Internet. There may be zero or more levels in between the local and global levels.

Hierarchical database organization relieves most IDPR servers of the burden of maintaining information about large portions of the Internet, most of which their clients will never request. Distributed database organization, with redundancy, allows clients to spread queries among IDPR servers, thus reducing the load on any one server. Furthermore, failure to communicate with a given IDPR server does not mean the loss of the entire service, as a client may obtain the information from another server. We note that some IDPR databases, such as the mapping database, may grow so large that it is not feasible to store the entire database at any single server.

IDPR routing information databases need not be completely consistent for proper policy route generation and use, because message forwarding along policy routes is completely specified by the source path agent. The absence of a requirement for consistency among IDPR routing information databases implies that there is no requirement for strict synchronization of these databases. Such synchronization is costly in terms of the message processing and transmission bandwidth required. Nevertheless, each IDPR route server should have a query/response mechanism for making its routing information database consistent with that of another route server, if necessary. A route server uses this mechanism to update its routing information database following detection of a gap or potential error in database contents, for example, when the route server returns to service after disconnection from the Internet.

A route server in one domain wishing to communicate with a route server in another domain must establish a policy route to the other route server's domain. To generate and establish a policy route, the route server must have sufficient routing information and it must know the other route server's domain. As route servers usually intercommunicate in order to obtain routing information, one might assume an ensuing deadlock in which a route server requires routing and mapping information to establish a policy route but must establish a policy route in order to obtain that routing and mapping information. However, such a deadlock should seldom persist, if the following IDPR functionality is in place:

1. There should be a mechanism that allows a route server to gain access to the identities of the other route servers within its domain, during route server initialization. Using this information, the route server may query these other route servers in order to update its routing information database.

2. There should also be a mechanism that allows a route server to gain access to its domain's adjacencies, during route server initialization. Using this information, the route server may establish policy routes to the adjacent domains in order to query their route servers for routing information when none is available within its own domain.

3. Once operational, a route server should collect all the routing information to which it has access. A domain usually does not restrict distribution of its routing information but instead distributes its routing information to all other Internet domains. Hence, a route server in a given domain is likely to receive routing information from most Internet domains.

4. There should be a mechanism that allows an operational route server to obtain the identities of external route servers from which it can obtain routing information and of the domains containing

these route servers. Furthermore, this mechanism should not require mapping server queries. Rather, each domain should distribute in its routing information messages the identities of all route servers, within its domain, that may be queried by clients outside of its domain.

When a host in one domain wishes to communicate with a host in another domain, the path agent in the source domain must establish a policy route to a path agent in the destination domain. However, the source path agent must first query a mapping server, to determine the identity of the destination domain. The queried mapping server may in turn contact other mapping servers to obtain a reply. As with route server communication, one might assume an ensuing deadlock in which a mapping server requires routing and mapping information to establish a policy route but must establish a policy route in order to obtain that routing and mapping information.

We have previously described how to minimize the potential for deadlock in obtaining routing information. To minimize the potential for deadlock in obtaining mapping information, there should be a mechanism that allows a mapping server to gain access to the identities of other mapping servers and the domains in which they reside, during mapping server initialization. Hence, no external mapping server queries are required to obtain this information.

3.2.3 Entity Identifiers

Each domain has a unique identifier within the Internet, specifically an ordinal number in the enumeration of Internet domains, determined by an Internet coordinator responsible for maintaining such information.

Each virtual gateway has a unique local identifier within a domain, derived from the adjacent domain's identifier together with the virtual gateway's ordinal number within an enumeration of the virtual gateways connecting the two domains. The administrators of both domains mutually agree upon the enumeration of the virtual gateways within their shared set of virtual gateways; selecting a single virtual gateway enumeration that applies in both domains eliminates the need to maintain a mapping between separate virtual gateway ordinal numbers in each domain.

Each policy gateway and route server has a unique local identifier within its domain, specifically an ordinal number in the domain administrator's enumeration of IDPR entities within its domain. This local identifier, when combined with the domain identifier, produces a unique identifier for the policy gateway or route server within the Internet.

3.3 Security and Reliability

The correctness of control information, and in particular routing-related information, distributed throughout the Internet is a critical factor affecting the Internet's ability to transport data. As the number and heterogeneity of Internet domains increases, so too does the potential for both information corruption and denial of service attacks. Thus, we have imbued the IDPR architecture with a variety of mechanisms to:

1. Promote timely delivery of control information.
2. Minimize acceptance and distribution of corrupted control information.
3. Verify authenticity of a source of control information.
4. Reduce the chances for certain types of denial of service attacks.

Consult [11] for a general security architecture for routing and [12] for a security architecture for inter-domain routing.

3.3.1 Retransmissions and Acknowledgements

All IDPR entities must make an effort to accept and distribute only correct IDPR control messages. Each IDPR entity that transmits an IDPR control message expects an acknowledgement from the recipient and must retransmit the message up to a maximum number of times when an acknowledgement is not forthcoming. An IDPR entity that receives an IDPR control message must verify message content integrity and source authenticity before accepting, acknowledging, and possibly redistributing the message.

3.3.2 Integrity Checks

Integrity checks on message contents promote the detection of corrupted information. Each IDPR entity that receives an IDPR control message must perform several integrity checks on the contents. Individual IDPR protocols may apply more stringent integrity checks than those listed below. The required checks include confirmation of:

1. Recognized message version.
2. Consistent message length.
3. Valid message checksum.

Each IDPR entity may also apply these integrity checks to IDPR data messages. Although the IDPR architecture only requires data message integrity checks at the last IDPR entity on a path, it does not preclude intermediate policy gateways from performing these checks as well.

3.3.3 Source Authentication

Authentication of a message's source promotes the detection of a rogue entity masquerading as another legitimate entity. Each IDPR entity that receives an IDPR control message must verify the authenticity of the message source. We recommend that the source of the message supply a digital signature for authentication by message recipients. The digital signature should cover the entire message contents, so that it can serve as the message checksum as well as the source authentication information.

Each IDPR entity may also authenticate the source of IDPR data messages; however, the IDPR architecture does not require source authentication of data messages. Instead, we recommend that higher level (end-to-end) protocols, not IDPR, assume the responsibility for data message source authentication, because of the amount of computation involved in verifying a digital signature.

3.3.4 Timestamps

Message timestamps promote the detection of out-of-date messages as well as message replays. Each IDPR entity that receives an IDPR control message must check that the message is timely. The IDPR control message must carry a timestamp supplied by the source, which serves to indicate the age of the message. IDPR entities use the absolute value of a timestamp to confirm that the message is current and use the relative difference between timestamps to determine which message contains the most recent information. Hence, all IDPR entities must possess internal clocks that are synchronized to some degree, in order for the absolute value of a message timestamp to be meaningful. The synchronization granularity required by the IDPR architecture is on the order of minutes and can be achieved manually. Any IDPR control message whose timestamp lies outside of the acceptable range may contain stale or corrupted information or may have been issued by a source whose internal clock has lost synchronization with the message recipient's internal clock.

IDPR data messages also carry timestamps; however, the IDPR architecture does not require timestamp acceptability checks on IDPR data messages. Instead, we recommend that IDPR entities only check IDPR data message timestamps during problem diagnosis, for example, when checking for suspected message replays.

3.4 An Example of IDPR Operation

We illustrate how IDPR works by stepping through an example. In this example, we assume that all domains support IDPR and that all domain egress points are policy gateways.

Suppose host H_X in domain $AD\ X$ wants to communicate with host H_Y in domain $AD\ Y$. H_X need not know the identity of its own domain or of H_Y's domain in order to send messages to H_Y. Instead, H_X simply forwards a message bound for H_Y to one of the gateways on its local network, according to its local forwarding information. If the recipient gateway is a policy gateway, the resident path agent determines how to forward the message outside of the domain. Otherwise, the recipient gateway forwards the message to another gateway in $AD\ X$, according to its local forwarding information. Eventually, the message will arrive at a policy gateway in $AD\ X$.

The path agent resident in the recipient policy gateway uses the message header, including source and destination addresses and any requested service information (for example, type of service), in order to determine whether it is an intra-domain or inter-domain message, and if inter-domain, whether it requires an IDPR policy route. Specifically, the path agent attempts to locate a forwarding information database entry for the given traffic flow. The forwarding information database will already contain entries for all of the following:

1. All intra-domain traffic flows. Intra-domain forwarding information is integrated into the forwarding database as soon as it is received.

2. Inter-domain traffic flows that do not require IDPR policy routes. Non-IDPR inter-domain forwarding information is integrated into the forwarding database as soon as it is received.

3. IDPR inter-domain traffic flows for which a path has already been set up. IDPR forwarding information is integrated into the forwarding database only during path setup.

The path agent uses the message header contents to guide the search for a forwarding information database entry for a traffic flow; we suggest a radix search to locate a database entry. When the search terminates, it either produces a forwarding information database entry or a directive to generate such an entry for an IDPR traffic flow. If the search terminates in an existing database entry, the path agent forwards the message according to that entry.

Suppose that the search terminates indicating that the traffic flow between H_X and H_Y requires an IDPR route and that no forwarding information database entry yet exists for this flow. In this case, the path agent first determines the source and destination domains associated with the message's source and destination addresses, before attempting to obtain a policy route. The path agent relies on the mapping servers to supply the domain information, but it caches all mapping server responses locally to limit the number of future queries. When attempting to resolve an address to a domain, the path agent always checks its local cache before contacting a mapping server.

After obtaining the source and destination domain information, the path agent attempts to obtain a policy route to carry the traffic from H_X to H_Y. The path agent relies on the route servers to supply policy routes, but it caches all route server responses locally to limit the number of future queries. When attempting to locate a suitable policy route, the path agent consults its local cache before contacting a route server. A policy route contained in the cache is suitable provided that its associated source domain is $AD\ X$, its associated destination domain is $AD\ Y$, and it satisfies the service requirements specified in the data message or through configuration.

If no suitable cache entry exists, the path agent queries the route server, providing it with the source and destination domains together with the service requests carried in the data message or specified through configuration. Upon receiving a policy route query, a route server consults its route database. If it cannot locate a suitable route in its route database, the route server attempts to generate at least one route to domain $AD\ Y$, consistent with the requested services for H_X.

The response to a successful route query consists of a set of candidate routes, from which the path agent makes its selection. We expect that a path agent will normally choose a single route from a candidate set. Nevertheless, the IDPR architecture does not preclude a path agent from selecting multiple routes from the candidate set. A path agent may desire multiple routes to support features such as fault tolerance or load balancing; however, the IDPR architecture does not specify how the path agent should use multiple routes. In any case, a route server always returns a response to a path agent's query, even if it is not successful in locating a suitable policy route.

If the policy route is a new route provided by the route server, there will be no existing path for the route and thus the path agent must set up such a path. However, if the policy route is an existing route extracted from the path agent's cache, there may well be an existing path for the route, set up to accommodate a different host traffic flow. The IDPR architecture permits multiple host traffic flows to use the same path, provided that all flows sharing the path travel between the same endpoint domains and have the same service requirements. Nevertheless, the IDPR architecture does not preclude a path agent from setting up distinct paths along the same policy route to preserve the distinction between host traffic flows.

The path agent associates an identifier with the path, which will be included in each message that travels down the path and will be used by the policy gateways along the path in order to determine how to forward the message. If the path already exists, the path agent uses the preexisting identifier. However, for new paths, the path agent chooses a path identifier that is different from those of all other paths that it manages. The path agent also updates its forwarding information database to reference the path identifier and modifies its search procedure to yield the correct forwarding information database entry given the data message header.

For new paths, the path agent initiates path setup, communicating the policy route, in terms of requested services, constituent domains, relevant transit policies, and the connecting virtual gateways, to policy gateways in intermediate domains. Using this information, an intermediate policy gateway determines whether to accept or refuse the path and to which policy gateway to forward the path setup information. The path setup procedure allows policy gateways to set up a path in both directions simultaneously. Each intermediate policy gateway, after path acceptance, updates its forwarding information database to include an entry that associates the path identifier with the appropriate previous and next hop policy gateways. Paths remain in place until they are torn down because of failure, expiration, or when resources are scarce, preemption in favor of other paths.

When a policy gateway in $AD\ Y$ accepts a path, it notifies the source path agent in $AD\ X$. We expect that the source path agent will normally wait until a path has been successfully established before using it to transport data traffic. However, the IDPR architecture does not preclude a path agent from forwarding data messages along a path prior to confirmation of successful path establishment. In this case, the source path agent transmits data messages along the path with full knowledge that the path may not yet have been successfully established at all intermediate policy gateways and thus that these data messages will be immediately discarded by any policy gateway not yet able to recognize the path identifier.

We note that data communication between H_X and H_Y may occur over two separate IDPR paths: one from $AD\ X$ to $AD\ Y$ and one from $AD\ Y$ to $AD\ X$. The reasons are that within a domain, hosts know nothing about policy gateways nor IDPR paths, and policy gateways know nothing about other policy gateways' existing IDPR paths. Thus, in $AD\ Y$, the policy gateway that terminates the path from $AD\ X$ may not be the same as the policy gateway that receives traffic from H_Y destined for H_X. In this case, receipt of traffic from H_Y forces the second policy gateway to set up a new path from $AD\ Y$ to $AD\ X$.

4 Accommodating a Large, Heterogeneous Internet

The IDPR architecture must be able to accommodate an Internet containing $O(10^4)$ domains, supporting diverse source and transit policies. Thus, we have endowed the IDPR architecture with many features that allow it to function effectively in such an environment.

4.1 Domain Level Routing

The IDPR architecture provides policy routing among administrative domains. In order to construct policy routes, route servers require routing information at the domain level only; no intra-domain details need be included in IDPR routing information. The size of the routing information database maintained by a route server depends not on the number of Internet gateways, networks, and links, but on how these gateways, networks, and links are grouped into domains and on what services they offer. Therefore, the number of entries in an IDPR routing information database depends on the number of domains and the number and size of the transit policies supported by these domains.

Policy gateways distribute IDPR routing information only when detectable inter-domain changes occur and may also elect to distribute routing information periodically (for example, on the order of once per day) as a backup. We expect that a pair of policy gateways within a domain will normally be connected such that when the primary intra-domain route fails, the intra-domain routing procedure will be able to construct an alternate route. Thus, an intra-domain failure is unlikely to be visible at the inter-domain level and hence unlikely to force an inter-domain routing change. Therefore, we expect that policy gateways will not often generate and distribute IDPR routing information messages.

IDPR entities rely on intra-domain routing procedures operating within domains to transport inter-domain messages across domains. Hence, IDPR messages must appear well-formed according to the intra-domain routing and addressing procedures in each domain traversed. Recall that source authentication information (refer to section 3.3.3 above) may cover the entire IDPR message. Thus, the IDPR portion of such a message cannot be modified at intermediate domains along the path without causing source authenticity checks to fail. Therefore, at domain boundaries, IDPR messages require encapsulation and decapsulation according to the routing procedures and addressing schemes operating with the given domain. Only policy gateways and route servers must be capable of handling IDPR-specific messages; other gateways and hosts simply treat the encapsulated IDPR messages like any other message. Thus, for the Internet to support IDPR, only a small proportion of Internet entities require special IDPR software.

With domain level routes, many different traffic flows may use not only the same policy route but also the same path, as long as their source domains, destination domains, and service requirements are compatible. The size of the forwarding information database maintained by a policy gateway depends not on the number of Internet hosts but on how these hosts are grouped into domains, which hosts intercommunicate, and on how much distinction a source domain wishes to preserve among its traffic flows. Therefore, the number of entries in an IDPR forwarding information database depends on the number of domains and the number of source policies supported by those domains. Moreover, memory associated with failed, expired, or disused paths can be reclaimed for new paths, and thus forwarding information for many paths can be accommodated in a policy gateway's forwarding information database.

4.2 Route Generation

Route generation is the most computationally complex part of IDPR, because of the number of domains and the number and heterogeneity of policies that it must accommodate. Route servers must generate policy routes that satisfy the requested services of the source domain and respect the offered services of the transit domains.

We distinguish requested qualities of service and route generation with respect to them as follows:

1. Optimal requested services include minimum route delay, minimum route delay variation, minimum session monetary cost, and maximum available route bandwidth. In the worst case, the computational complexity of generating a route that is optimal with respect to a given requested service is $O(N + L)$ for breadth-first (BF) search and $O((N + L) \log N)$ for Dijkstra's shortest path first (SPF) search, where N is the number of nodes and L is the number of links in the search graph. Multi-criteria optimization, for example finding a route with minimal delay variation and minimal session monetary cost, may be defined in several ways. One approach to multi-criteria optimization is to assign each link a single value equal to a weighted sum of the values of the individual offered qualities of service and generate a route that is optimal with respect to this new criterion. However, selecting the weights that yield the desired route generation behavior is itself an optimization procedure and hence not trivial.
2. Requested service limits include upper bounds on route delay, route delay variation, and session monetary cost and lower bounds on available route bandwidth. Generating a route that must satisfy more than one quality of service constraint, for example route delay of no more than X seconds and available route bandwidth of no less than Y bits per second, is an NP-complete problem.

To contain the combinatorial explosion of processing and memory costs associated with route generation, we supply the following guidelines for generation of suitable policy routes:

1. Each route server should only generate policy routes from the perspective of its own domain as source; it need not generate policy routes for arbitrary source/destination domain pairs. Thus, we

can distribute the computational burden over all route servers.

2. Route servers should precompute routes for which they anticipate requests and should generate routes on demand only in order to satisfy unanticipated route requests. Hence, a single route server can distribute its computational burden over time.

3. Route servers should cache the results of route generation, in order to minimize the computation associated with responding to future route requests.

4. To handle multi-criteria optimization in route selection, a route server should generate routes that are optimal with respect to the first specified optimal requested service listed in the source policy. The route server should resolve ties between otherwise equivalent routes by evaluating these routes according to the other optimal requested services, in the order in which they are specified. With respect to the route server's routing information database, the selected route is optimal according to the first optimal requested service but is not necessarily optimal according to any other optimal requested service.

5. To handle requested service limits, a route server should always select the first route generated that satisfies all of the requested service limits.

6. To handle a mixture of requested service limits and optimal requested services, a route server should generate routes that satisfy all of the requested service limits. The route server should resolve ties between otherwise equivalent routes by evaluating these routes as described in the multi-criteria optimization case.

7. All else being equal, a route server should always prefer minimum-hop routes, because they minimize the amount of network resources consumed by the routes.

All domains need not execute the identical route generation procedure. Each domain administrator is free to specify the IDPR route generation procedure for route servers in its own domain, making the procedure as simple or as complex as desired.

4.3 Super Domains

A *super domain* is itself an administrative domain, comprising a set of contiguous domains with similar transit policies and formed through consensus of the administrators of the consituent domains. Super domains provide a mechanism for reducing the amount of IDPR routing information distributed throughout the Internet. Given a set of n contiguous domains with consistent transit policies, the amount of routing information associated with the set is approximately n times smaller when the set is considered as a single super domain than when it is considered as n individual domains.

When forming a super domain from constituent domains whose transit policies do not form a consistent set, one must determine which transit policies to distribute in the routing information for the super domain. The range of possibilities is bounded by the following two alternatives, each of which reduces the amount of routing information associated with the set of constituent domains:

1. The transit policies supported by the super domain are derived from the union of the access restrictions and the intersection of the qualities of service, over all constituent domains. In this case, the formation of the super domain reduces the number of services offered by the constituent domains, but guarantees that none of these domains' access restrictions are violated.

2. The transit policies supported by the super domain are derived from the intersection of the access restrictions and the union of the qualities of service. In this case, the formation of the super domain increases the number of services offered by the constituent domains, but requires these domains to relax their access restrictions.

Thus, we recommend that domain administrators refrain from arbitrarily grouping domains into super domains, unless they fully understand the consequences.

The existence of super domains imposes a hierarchy on domains within the Internet. For model consistency, we assume that there is a single super domain at the top of the hierarchy, which contains the set of all high-level domains. A domain's identity is defined relative to the domain hierarchy. Specifically, a domain's identity may be defined in terms of the domains containing it, the domains it contains, or both.

For any domain $AD\ X$, the universe of distribution for its routing information usually extends only to those domains contained in $AD\ X$'s immediate super domain and at the same level of the hierarchy as $AD\ X$. However, the IDPR architecture does not preclude $AD\ X$ from distributing its routing information to domains at arbitrarily high levels in the hierarchy, as long as these domains share a common super domain with $AD\ X$. For example, the administrator of an individual domain within a super domain may wish to have one of its transit policies advertised outside of the immediate super domain, so that other domains can take advantage of a quality of service not offered by the super domain itself. In this case, the super domain and the consituent domain may distribute routing information at the same level in the domain hierarchy, even though one domain actually contains the other.

We note that the existence of super domains may restrict the number of routes available to source domains with access restrictions. For example, suppose that a source domain $AD\ X$ has source policies that preclude its traffic from traversing a domain $AD\ Y$ and that $AD\ Y$ is contained in a super domain $AD\ Z$. If domains within $AD\ Z$ do not advertise routing information separately, then route servers within $AD\ X$ do not have enough routing information to construct routes that traverse $AD\ Z$ but that avoid $AD\ Y$. Hence, route servers in $AD\ X$ must generate routes that avoid $AD\ Z$ altogether.

4.4 Domain Communities

A *domain community* is a group of domains to which a given domain distributes routing information, and hence domain communities may be used to limit routing information distribution. Domain communities not only reduce the costs associated with distributing and storing routing information but also allow concealment of routing information from domains outside of the community. Unlike a super domain, a domain community is not necessarily an administrative domain. However, formation of a domain community may or may not involve the consent of the administrators of the member domains, and the definition of the community may be implicit or explicit.

Each domain administrator determines the extent of distribution of its domain's routing information and hence unilaterally defines a domain community. By default, this community encompasses all Internet domains. However, the domain administrator may restrict community membership by describing the community as a neighborhood (defined in terms of domain hops) or as a list of member domains.

A group of domain administrators may mutually agree on distribution of their domains' routing information among their domains and hence multilaterally define a domain community. By default, this community encompasses all Internet domains. However, the domain administrators may restrict community membership by describing the community as a list of member domains. In fact, this domain community may serve as a multicast group for routing information distribution.

4.5 Robustness in the Presence of Failures

The IDPR architecture possesses the following features that make it resistent to failures in the Internet:

1. Multiple connections between adjacent policy gateways in a virtual gateway and between peer and neighbor policy gateways across an administrative domain minimize the number of single component failures that are visible at the inter-domain level.

2. Policy gateways distribute IDPR routing information immediately after detecting a connectivity failure at the inter-domain level, and route servers immediately incorporate this information into their routing information databases. This ensures that new policy routes will not include those domains involved in the connectivity failure.

3. The routing information database query/response mechanism ensures rapid updating of the routing information database for a previously failed route server following the route server's reconnection

to the Internet.

4. To minimize user service disruption following a failure in the primary path, policy gateways attempt local path repair immediately after detecting a connectivity failure. Moreover, path agents may maintain standby alternate paths that can become the primary path if necessary.

5. Policy gateways within a domain continuously monitor domain connectivity and hence can detect and identify domain partitions. Moreover, IDPR can continue to operate properly in the presence of partitioned domains.

4.5.1 Path Repair

Failure of one or more entities on a given policy route may render the route unusable. If the failure is within a domain, IDPR relies on the intra-domain routing procedure to find an alternate route across the domain, which leaves the path unaffected. If the failure is in a virtual gateway, policy gateways must bear the responsibility of repairing the path. Policy gateways nearest to the failure are the first to recognize its existence and hence can react most quickly to repair the path.

Relinquishing control over path repair to policy gateways in other domains may be unacceptable to some domain administrators. The reason is that these policy gateways cannot guarantee construction of a path that satisfies the source policies of the source domain, as they have no knowledge of other domains' source policies.

Nevertheless, limited local path repair is feasible, without distributing either source policy information throughout the Internet or detailed path information among policy gateways in a domain or in a virtual gateway. We say that a path is *locally repairable* if there exists an alternate route between two policy gateways on the path, separated by at most one policy gateway. This definition covers path repair in the presence of failed routes between consecutive policy gateways as well as failed policy gateways themselves.

A policy gateway attempts local path repair, proceeding in the forward direction of the path, upon detecting that the next policy gateway on a path is no longer reachable. The policy gateway must retain enough of the original path setup information to repair the path locally. Using the path setup information, the policy gateway attempts to locate a route around the unreachable policy gateway. Specifically, the policy gateway attempts to establish contact with either:

1. A peer of the unreachable policy gateway.

2. A peer of itself, if the unreachable policy gateway is an adjacent policy gateway and if the given policy gateway no longer has direct connections to any adjacent policy gateways.

The contacted policy gateway attempts to locate the next policy gateway following the unreachable policy gateway, on the original path. If successful, the contacted policy gateway informs the requesting policy gateway. In this case, the requesting, contacted, and next policy gateways update their forwarding information databases to conform to the new part of the path. If not successful, the contacted policy gateway initiates teardown of the original path; in this case, the source path agent must find a new route to the destination.

4.5.2 Partitions

A *domain partition* exists whenever there are at least two entities within the domain that can no longer communicate over any intra-domain route. Domain partitions not only disrupt intra-domain communication but also may interfere with inter-domain communication, particularly when the partitioned domain is a transit domain. Therefore, we have designed the IDPR architecture to permit effective use of partitioned domains and hence maximize Internet connectivity in the presence of domain partitions.

When a domain is partitioned, it becomes a set of multiple distinct components. A *domain component* is a subset of the domain's entities such that all entities within the subset are mutually reachable via intra-domain routes, but no entities in the complement of the subset are reachable via intra-domain routes from entities within the subset. Each domain component has a unique identifier, namely the identifier of the domain together with the ordinal number of the lowest-numbered operational policy gateway within the domain component. No negotiation among policy gateways is necessary to determine the domain component's lowest-numbered operational policy gateway. Instead, within each domain component, all policy gateway members discover mutual reachability through intra-domain reachability information. Therefore, all members have a consistent view of which is the lowest-numbered operational policy gateway in the component.

IDPR entities can detect and compensate for all domain partitions that isolate at least two groups of policy gateways from each other. They cannot, however, detect any domain partition that isolates groups of hosts only. Note that a domain partition may segregate portions of a virtual gateway, such that peer policy gateways lie in separate domain components. Although itself partitioned, the virtual gateway does not assume any additional identities. However, from the perspective of the adjacent domain, the virtual gateway now connects to two separate domain components.

Policy gateways use partition information to select routes across virtual gateways to the correct domain components. They also distribute partition information to route servers as part of the IDPR routing information. Thus, route servers know which domains are partitioned. However, route servers do not know which hosts reside in which components of a partitioned domain; tracking this information would require extensive computation and communication. Instead, when a route server discovers that the destination of a requested route is a partitioned domain, it attempts to generate a suitable policy route to each component of the destination domain. Generation of multiple routes, on detection of a partitioned destination domain, maximizes the chances of obtaining at least one policy route that can be used for communication between the source and destination hosts.

References

[1] Y. Rehkter. EGP and policy based routing in the new NSFNET backbone. *IETF RFC 1092*. February 1989.

[2] D. Clark. Policy routing in internet protocols. *IETF RFC 1102*. May 1989.

[3] H-W. Braun. Models of policy based routing. *IETF RFC 1104*. June 1989.

[4] B. Leiner. Policy issues in interconnecting networks. *IETF RFC 1124*. September 1989.

[5] D. Estrin. Requirements for policy based routing in the research internet. *IETF RFC 1125*. November 1989.

[6] M. Little. Goals and functional requirements for inter-autonomous system routing. *IETF RFC 1126*. July 1989.

[7] J. Honig, D. Katz, M. Mathis, Y. Rekhter, and J. Yu. Application of the border gateway protocol in the internet. *IETF RFC 1164*. June 1990.

[8] K. Lougheed and Y. Rekhter. A border gateway protocol 3 (BGP-3). *IETF RFC 1267.* October 1991.

[9] Y. Rekhter and T. Li, editors. A border gateway protocol 4 (BGP-4). *IETF Internet Draft.* April 1992.

[10] ANSI. Intermediate system to intermediate system inter-domain routeing information exchange protocol. *ANSI Document X3S3.3/90-132,* June 1990.

[11] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. Ph.D. Thesis. Department of Electrical Engineering and Computer Science. MIT. August 1988.

[12] D. Estrin and G. Tsudik. *Secure Control of Transit Internetwork Traffic.* TR-89-15. Computer Science Department. University of Southern California. To appear in *Computer Networks and ISDN Systems*, 1991.

[13] J.J. Garcia-Luna-Aceves. A unified approach for loop-free routing using link states or distance vectors. *ACM Computer Communication Review*, Vol. 19, No. 4, SIGCOMM 1989, pp. 212-223.

[14] W.T. Zaumen and J.J. Garcia-Luna-Aceves. Dynamics of distributed shortest-path routing algorithms. *ACM Computer Communication Review*, Vol. 21, No. 4, SIGCOMM 1991, pp. 31-42.