Inter-Domain Policy Routing Working Group
Internet Draft
May 1992

M. Steenstrup
BBN Systems and Technologies
Expires 30 November 1992

Inter-Domain Policy Routing Protocol Specification: Version 1

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 grousp 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 a protocol specification. Distribution of this Internet Draft is unlimited. Please send comments to idpr-wg@bbn.com.

Abstract

We present the set of protocols and procedures that constitute inter-domain policy routing (IDPR). IDPR includes the virtual gateway protocol, the flooding protocol, the route server query protocol, the route generation procedure, the path control protocol, and the data message forwarding procedure.

Contributors

The following people have contributed to the protocols and procedures described in this document: Helen Bowns, Lee Breslau, Ken Carlberg, Isidro Castineyra, Deborah Estrin, Tony Li, Mike Little, Katia Obraczka, Sam Resheff, Martha Steenstrup, Gene Tsudik, and Robert Woodburn.

Contents

1 Introduction

In this document, we specify the protocols and procedures that compose *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 services requested within the constraints stipulated for the domains transited. IDPR supports link state routing information distribution and route generation in conjunction with source specified message forwarding. Refer to [5] for a detailed justification of our approach to inter-domain policy routing.

1.1 Domain Elements

The IDPR architecture has been designed to accommodate an Internet with tens of thousands of administrative domains collectively containing hundreds of thousands of local networks. Inter-domain policy routes are constructed using information about the services offered by, and the connectivity between, administrative domains. The intra-domain details – gateways, networks, and links traversed – of an inter-domain policy route are the responsibility of intra-domain routing and are thus outside the scope of IDPR.

An *administrative domain* (AD) is a collection of contiguous hosts, gateways, networks, and links managed by a single administrative authority that defines service restrictions for transit traffic and service requirements for locally-generated traffic, and selects the addressing schemes and routing procedures that apply within the domain. Each domain has a unique numeric identifier within the Internet.

Virtual gateways (VGs) are the only IDPR-recognized connecting points between adjacent domains. Each virtual gateway is a collection of directly-connected policy gateways (see below) in two adjoining domains, whose existence has been sanctioned by the administrators of both domains. The domain administrators may agree to establish more than one virtual gateway between the two domains. For each such virtual gateway, the two administrators together assign a local numeric identifier, unique within the set of virtual gateways connecting the two domains. To produce a virtual gateway identifier unique within its domain, a domain administrator concatenates the mutually assigned local virtual gateway identifier together with the adjacent domain's identifier.

Policy gateways (PGs) are the physical gateways within a virtual gateway. Each policy gateway enforces service restrictions on IDPR transit traffic, as stipulated by the domain administrator, and forwards the traffic accordingly. Within a domain, two policy gateways are *neighbors* if they are in different virtual gateways. A single policy gateway may belong to multiple 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. Adjacent policy gateways are *directly connected* if the only Internet-addressable entities attached

to the connecting medium are policy gateways in the virtual gateways. Note that this definition implies that not only point-to-point links but also networks may serve as direct connections between adjacent policy gateways. The domain administrator assigns to each of its policy gateways a numeric identifier, unique within that domain.

A *domain component* is a subset of a domain's entities such that all entities within the subset are mutually reachable via intra-domain routes, but no entities outside the subset are reachable via intra-domain routes from entities within the subset. Normally, a domain consists of a single component, namely itself; however, when partitioned, a domain consists of multiple components. Each domain component has an identifier, unique within the Internet, composed of the domain identifier together with the identifier of the lowest-numbered operational policy gateway within the component. All operational policy gateways within a domain component can discover mutual reachability through intra-domain routing information. Hence, all such policy gateways can consistently determine, without explicit negotiation, which of them has the lowest number.

1.2 Policy

With IDPR, each domain administrator sets *transit policies* that dictate how and by whom the resources in 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 in 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, and reliability.

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

1.3 IDPR Functions

IDPR 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.

1.3.1 IDPR Entities

Several different entities are responsible for performing the IDPR functions.

Policy gateways, the only IDPR-recognized connecting points between adjacent domains, collect and distribute routing information, participate in path setup, forward data messages along established paths, and maintain forwarding information databases.

Path agents, resident within policy gateways and within route servers (see below), act on behalf of hosts to select policy routes, to set up and manage paths, and to maintain forwarding information databases. 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 the path agent.

Route servers maintain both the routing information database and the route database, and they generate policy routes using the routing information collected and the source policies requested by the path agents. A route server may reside within a policy gateway, or it may exist as an autonomous entity. Separating the route server functions from the policy gateways frees the policy gateways from both the memory intensive task of routing information and route database maintenance and the computationally intensive task of route generation. Route servers, like policy gateways, each have a unique numeric identifier within their domain, assigned by the domain administrator.

Given the size of the current Internet, each policy gateway can perform the route server functions, in addition to its message forwarding functions, with little or no degradation in message forwarding performance. Aggregating the routing functions into policy gateways simplifies implementation; one need only install IDPR protocols in policy gateways. Moreover, it simplifies communication between routing functions, as all functions reside within each policy gateway. As the Internet grows, the processing and memory required to perform the route server functions may become a burden for the policy gateways.

When this happens, each domain administrator should separate the route server functions from the policy gateways in its domain.

Mapping servers maintain the database of mappings that resolve Internet names and addresses to domain identifiers. The mapping server function will be integrated into the existing DNS name service.

Configuration servers maintain the databases of configured information that apply to IDPR entities within their domains. Configuration information for a given domain includes transit policies (i.e., service offerings), source policies (i.e., service requirements), and mappings between local IDPR entities and their names and addresses. The configuration server function will be integrated into a domain's existing network management system.

1.4 Policy Semantics

The source and transit policies supported by IDPR are intended to accommodate a wide range of services available throughout the Internet. We describe the semantics of these policies, concentrating on the access restriction aspects. To express these policies in this document, we have chosen to use a syntactic variant of Clark's policy term notation [1]. However, we provide a more succinct syntax (see [6]) for actually configuring source and transit policies.

1.4.1 Source Policies

Each source policy takes the form of a collection of sets as follows:

$\{((H_{11}, s_{11}), \ldots, (H_{1f_1}, s_{1f_1})), \ldots, ((H_{n1}, s_{n1}), \ldots, (H_{nf_n}, s_{nf_n}))\}$: The set of groups of source/destination traffic flows to which the source policy applies. Each traffic flow group $((H_{i1}, s_{i1}), \ldots, (H_{if_i}, s_{if_i}))$ contains a set of source hosts and corresponding destination hosts. Here, H_{ij} represents a host, and $s_{ij} \in \{source, destination\}$ represents an indicator of whether H_{ij} is to be considered as a source or as a destination.

$\{(AD_1, x_1), \ldots, (AD_m, x_m)\}$: The set of transit domains that the traffic flows should favor, avoid, or exclude. Here, AD_i represents a set of domains, and $x_i \in \{favor, avoid, exclude\}$ represents an indicator of whether routes including members of AD_i are to be favored, avoided if possible, or unconditionally excluded.

UCI : The user class applied to the traffic flows listed.

Requested : The set of requested services not related to access restrictions, i.e., service quality and monetary cost.

The path agent honoring such a source policy will select a route for a traffic flow from any source host H_{ij} to any destination host H_{ik}, where $1 \leq i \leq n$ and $1 \leq j,k \leq f_i$, provided that:

1. For each domain, AD_p, contained in the route, $AD_p \neq AD_k$, where $x_k = exclude$ and $1 \leq k \leq m$.
2. The route provides the services listed in the set *Requested*.

1.4.2 Transit Policies

Each transit policy takes the form of a collection of sets as follows:

$\{((H_{11}, AD_{11}, s_{11}), \ldots, (H_{1f_1}, AD_{1f_1}, s_{1f_1})), \ldots, ((H_{n1}, AD_{n1}, s_{n1}), \ldots, (H_{nf_n}, AD_{nf_n}, s_{nf_n}))\}$: The set of groups of source and destination hosts and domains to which the transit policy applies. Each host/domain group $((H_{i1}, AD_{i1}, s_{i1}), \ldots, (H_{if_i}, AD_{if_i}, s_{if_i}))$ contains a set of source and destination hosts and domains such that this transit domain will carry traffic from each source listed to each destination listed. Here, H_{ij} represents a set of hosts, AD_{ij} represents a set of domains containing H_{ij}, and $s_{ij} \subseteq \{source, destination\}$ represents an indicator of whether (H_{ij}, AD_{ij}) is to be considered as a set of sources, destinations, or both.

Time : The set of time intervals during which the transit policy applies.

UCI : The set of user classes to which the transit policy applies.

Offered : The set of offered services not related to access restrictions, i.e., service quality and monetary cost.

$\{((VG_{11}, e_{11}), \ldots, (VG_{1g_1}, e_{1g_1})), \ldots, ((VG_{m1}, e_{m1}), \ldots, (VG_{mg_m}, e_{mg_m}))\}$: The set of groups of entry and exit virtual gateways to which the transit policy applies. Each virtual gateway group $((VG_{i1}, e_{i1}), \ldots, (VG_{ig_i}, e_{ig_i}))$ contains a set of domain entry and exit points such that each entry virtual gateway can reach (barring any intra-domain routing failure) each exit virtual gateway via an intra-domain route supporting the transit policy. Here, VG_{ij} represents a virtual gateway, and $e_{ij} \subseteq \{entry, exit\}$ represents an indicator of whether VG_{ij} is to be considered as a domain entry point, exit point, or both.

The domain advertising such a transit policy will carry traffic from any host in the set H_{ij} in AD_{ij} to any host in the set H_{ik} in AD_{ik}, where $1 \leq i \leq n$ and $1 \leq j,k \leq f_i$, provided that:

1. $source \in s_{ij}$.
2. $destination \in s_{ik}$.

3. Traffic from H_{ij} enters the domain during one of the intervals in the set *Time*.

4. Traffic from H_{ij} carries one of the user class identifiers in the set *UCI*.

5. Traffic from H_{ij} enters via any VG_{uv} such that *entry* $\in e_{uv}$, where $1 \leq u \leq m$ and $1 \leq v \leq g_u$.

6. Traffic to H_{ik} leaves via any VG_{uw} such that *exit* $\in e_{uw}$, where $1 \leq w \leq g_u$.

1.5 IDPR Message Encapsulation

There are two kinds of IDPR messages:

1. *Data messages* containing user data generated by hosts.

2. *Control messages* containing IDPR protocol-related control information generated by policy gateways and route servers.

Within the Internet, only policy gateways and route servers are able to generate, recognize, and process IDPR messages. The existence of IDPR is invisible to all other gateways and hosts, including mapping servers and configuration servers. Mapping servers and configuration servers perform necessary but ancillary functions for IDPR, and thus they are not required to handle IDPR messages.

An IDPR entity places IDPR-specific information in each IDPR control message it originates; this information is significant only to recipient IDPR entities. Using *encapsulation* across each domain, an IDPR message tunnels from source to destination across the Internet through domains that may employ disparate intra-domain addressing schemes and routing procedures.

As an alternative to encapsulation, we had considered embedding IDPR in IP, as a set of IP options. However, this approach has the following disadvantages:

1. Only domains that support IP would be able to participate in IDPR; domains that do not support IP would be excluded.

2. Each gateway, policy or other, in a participating domain would at least have to recognize the IDPR option, even if it did not execute the IDPR protocols. However, most commercial routers are not optimized for IP options processing, and so IDPR message handling might require significant processing at each gateway.

3. For some IDPR protocols, in particular path control, the size restrictions on IP options would preclude inclusion of all of the necessary protocol-related information.

For these reasons, we decided against the IP option approach and in favor of encapsulation.

An IDPR message travels from source to destination between consecutive policy gateways. Each policy gateway encapsulates the IDPR message with information, for example an IP header, that will enable the message to reach the next policy gateway. Note that the encapsulating header and the IDPR-specific information may increase message size beyond the MTU of the given domain. However, message fragmentation and reassembly is the responsibility of the protocol, for example IP, that encapsulates IDPR messages for transport between successive policy gateways; it is not the responsibility of IDPR itself.

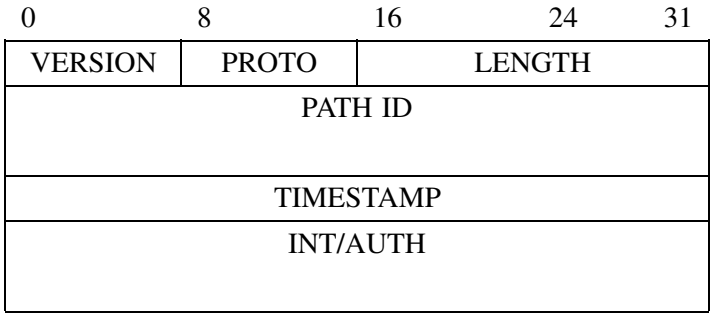
A policy gateway, when forwarding an IDPR message to a peer or a neighbor policy gateway, encapsulates the message in accordance with the addressing scheme and routing procedure of the given domain and indicates in the protocol field of the encapsulating header that the message is indeed an IDPR message. Intermediate gateways between the two policy gateways forward the IDPR message as they would any other message, using the information in the encapsulating header. Only the recipient policy gateway interprets the protocol field, strips off the encapsulating header, and processes the IDPR message.

A policy gateway, when forwarding an IDPR message to a directly-connected adjacent policy gateway, encapsulates the message in accordance with the addressing scheme of the entities within the virtual gateway and indicates in the protocol field of the encapsulating header that the message is indeed an IDPR message. The recipient policy gateway strips off the encapsulating header and processes the IDPR message. We recommend that the recipient policy gateway perform the following validation check of the encapsulating header, prior to stripping it off. Specifically, the recipient policy gateway should verify that the source address and the destination address in the encapsulating header match the adjacent policy gateway's address and its own address, respectively. Moreover, the recipient policy gateway should verify that the message arrived on the interface designated for the direct connection to the adjacent policy gateway. These checks help to ensure that IDPR traffic that crosses domain boundaries does so only over direct connections between adjacent policy gateways.

Policy gateways forward IDPR data messages according to a forwarding information database which maps path identifiers into next policy gateways, and they forward IDPR control messages according to next policy gateways selected by the particular IDPR control protocol. Distinguishing IDPR data messages and IDPR control messages at the encapsulating protocol level, instead of at the IDPR protocol level, eliminates an extra level of dispatching and hence makes IDPR message forwarding more efficient. When encapsulated within IP messages, IDPR data messages and IDPR control messages carry the IP protocol numbers 35 and 38, respectively.

1.5.1 IDPR Data Message Format

The path agents at a source domain determine which data messages generated by local hosts are to be handled by IDPR. To each data message selected for IDPR handling, a source path agent prepends the following header:



VERSION (8 bits) Version number for IDPR data messages, currently equal to 1.

PROTO (8 bits) Numeric identifier for the protocol with which to process the contents of the IDPR data message. Only the path agent at the destination interprets and acts upon the contents of the PROTO field.

LENGTH (16 bits) Length of the entire IDPR data message in bytes.

PATH ID (64 bits) Path identifier assigned by the source path agent and consisting of the numeric identifier of the path agent's domain (16 bits), the numeric identifier of the path agent's policy gateway (16 bits), and the path agent's local path identifier (32 bits) (see section 7.2).

TIMESTAMP (32 bits) Number of seconds elapsed since 1 January 1970 0:00 GMT.

INT/AUTH (variable) Computed integrity/authentication value, dependent on the type of integrity/authentication requested during path setup.

We describe the IDPR control message header in section 2.4.

1.6 Security

IDPR contains mechanisms for verifying message integrity and source authenticity and for protecting against certain types of denial of service attacks. It is particularly important to keep IDPR control messages intact, because they carry control information critical to the construction and use of viable policy routes between domains.

All IDPR messages carry a single piece of information, referred to in the IDPR documentation as the *integrity/authentication value*, which may be used not only to detect message corruption but also to verify the authenticity of the message source. The Internet coordinator[1] sanctions the set of valid algorithms which may be used to compute the integrity/authentication values. This set may include algorithms that perform only message integrity checks such as n-bit cyclic redundancy checksums (CRCs), as well as algorithms that perform both message integrity and source authentication checks such as signed hash functions of message contents.

Each domain administrator is free to select any integrity/authentication algorithm, from the set specified by the Internet coordinator, for computing the integrity/authentication values contained in its domain's messages. However, we recommend that IDPR entities in each domain be capable of executing all of the valid algorithms so that an IDPR control message originating at an entity in one domain can be properly checked by an entity in another domain.

IDPR control messages must carry a non-null integrity/authentication value. We recommend that the integrity/authentication algorithm be a digital signature, in particular an algorithm such as MD4 [15] or MD5 [16], which simultaneously verifies message integrity and source authenticity. The digital signature may be based on either public-key or private-key cryptography. Our approach to digital signature use in IDPR is based on the privacy-enhanced Internet electronic mail service [12]-[14], already available in the Internet.

We do not require IDPR data messages to carry a non-null integrity/authentication value. In fact, we recommend that a higher layer (end-to-end) protocol, and not IDPR, assume responsibility for checking the integrity and authenticity of data messages, because of the amount of computation required.

1.7 Timestamps and Clock Synchronization

Each IDPR message carries a timestamp (expressed in seconds elapsed since 1 January 1970 0:00 GMT, following the UNIX precedent) supplied by the source IDPR entity, which serves to indicate the age of the message. IDPR entities use the absolute value of the timestamp to confirm that a message is current and use the relative difference between timestamps to determine which message contains the more recent information.

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

[1]Throughout this document, we use the term *Internet coordinator* to refer to a coordinating body that makes administrative decisions about the Internet as a whole. There may actually be separate bodies responsible for separate aspects of the Internet. However, for simplicity, we use the single term Internet coordinator.

by IDPR is on the order of minutes and can be achieved manually. Thus, a synchronization protocol operating among all IDPR entities in all domains, while useful, is not necessary.

An IDPR entity can determine whether to accept or reject a message based on the discrepancy between the message's timestamp and the entity's own internal clock time. Any IDPR 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. Timestamp checks are required for control messages because of the consequences of propagating and acting upon incorrect control information. However, timestamp checks are discretionary for data messages but may be invoked during problem diagnosis, for example, when checking for suspected message replays.

We note that none of the IDPR protocols contain explicit provisions for dealing with an exhausted timestamp space. As timestamp space exhaustion will not occur until well into the next century, we expect timestamp space viability to outlast the IDPR protocols.

1.8 Network Management

In this document, we do not describe how to configure and manage IDPR. However, in this section, we do provide a list of the types of IDPR configuration information required. Also, in later sections describing the IDPR protocols, we briefly note the types of exceptional events that must be logged for network management. Complete descriptions of IDPR entity configuration and IDPR managed objects appear in [6] and [7] respectively.

To participate in inter-domain policy routing, policy gateways and route servers within a domain each require configuration information. Some of the configuration information is specifically defined within the given domain, while some of the configuration information is universally defined throughout the Internet. A domain administrator determines domain-specific information, and the Internet coordinator determines globally significant information. (Refer to [6] for detailed instructions on configuring an administrative domain to support IDPR.)

To produce valid domain configurations, the domain administrators must receive the following global information from the Internet coordinator:

1. For each Internet integrity/authentication type, the numeric identifier, syntax, and semantics. Available integrity and authentication types include but are not limited to:

   (a) public-key based signatures;

   (b) private-key based signatures;

   (c) cyclic redundancy checksums;

(d) no integrity/authentication.

2. For each Internet user class, the numeric identifier, syntax, and semantics. Available user classes include but are not limited to:

   (a) federal (and if necessary, agency-specific such as NSF, DOD, DOE, etc.);

   (b) research;

   (c) commercial;

   (d) support.

3. For each Internet offered service that may be advertised in transit policies, the numeric identifier, syntax, and semantics. Available offered services include but are not limited to:

   (a) average message delay;

   (b) message delay variation;

   (c) average bandwidth available;

   (d) bandwidth variation;

   (e) maximum transfer unit (MTU);

   (f) charge per byte;

   (g) charge per message;

   (h) charge per unit time.

4. For transit policy applicability time periods, the syntax and semantics.

5. For each Internet requested service that may appear within a path setup message, the numeric identifier, syntax, and semantics. Available requested services include but are not limited to:

   (a) maximum path life in minutes, messages, or bytes;

   (b) integrity/authentication algorithms to be used on data messages sent over the path;

   (c) path delay;

   (d) minimum delay for path;

   (e) path delay variation;

   (f) minimum delay variation path;

   (g) path bandwidth;

   (h) maximum bandwidth path;

   (i) session monetary cost;

   (j) minimum session monetary cost path;

(k) billing address;

(l) charge number.

In an Internet-wide implementation of IDPR, the set of global configuration parameters and their syntax and semantics must be consistent across all participating domains. The Internet coordinator, responsible for establishing the full set of global configuration parameters, relies on the cooperation of the administrator of each participating domain to ensure that the global parameters are consistent with the desired transit policies and user service requirements of each domain. Moreover, as the syntax and semantics of the global parameters affects the syntax and semantics of the corresponding IDPR software, the Internet coordinator must carefully define each global parameter so that it is unlikely to require future modifications.

The Internet coordinator distributes configured global information to configuration servers in all domains participating in IDPR. Each domain administrator uses the configured global information maintained by its configuration servers to develop configurations for each IDPR entity within its domain. Each configuration server retains a copy of the configuration for each local IDPR entity and also distributes the configuration to that entity using, for example, SNMP.

1.8.1 Policy Gateway Configuration

Each policy gateway must contain sufficient configuration information to perform its IDPR functions, which subsume those of the path agent. These include: validating IDPR control messages; generating and distributing virtual gateway connectivity and routing information messages to peer, neighbor, and adjacent policy gateways; distributing routing information messages to route servers in its domain; resolving destination addresses; requesting policy routes from route servers; selecting policy routes and initiating path setup; ensuring consistency of a path with its domain's transit policies; establishing path forwarding information; and forwarding IDPR data messages along existing paths. The necessary configuration information includes the following:

1. For each integrity/authentication type, the numeric identifier, syntax, and semantics.
2. For each policy gateway and route server in the given domain, the numeric identifier and set of addresses or names.
3. For each virtual gateway connected to the given domain, the numeric identifier, the numeric identifiers of the constituent peer policy gateways, and the numeric identifier of the adjacent domain.
4. For each virtual gateway of which the given policy gateway is a member, the numeric identifiers and set of addresses of the constituent adjacent policy gateways.

5. For each policy gateway directly-connected and adjacent to the given policy gateway, the local connecting interface.

6. For each local route server to which the given policy gateway distributes routing information, the numeric identifier.

7. For each source policy applicable to hosts within the given domain, the syntax and semantics.

8. For each transit policy applicable to the domain, the numeric identifier, syntax, and semantics.

9. For each requested service that may appear within a path setup message, the numeric identifier, syntax, and semantics.

10. For each source user class, the numeric identifier, syntax, and semantics.

1.8.2 Route Server Configuration

Each route server must contain sufficient configuration information to perform its IDPR functions, which subsume those of the path agent. These include: validating IDPR control messages; deciphering and storing the contents of routing information messages; exchanging routing information with other route servers and policy gateways; generating policy routes that respect transit policy restrictions and source service requirements; distributing policy routes to path agents in policy gateways; resolving destination addresses; selecting policy routes and initiating path setup; establishing path forwarding information; and forwarding IDPR data messages along existing paths. The necessary configuration information includes the following:

1. For each integrity/authentication type, the numeric identifier, syntax, and semantics.

2. For each policy gateway and route server in the given domain, the numeric identifier and set of addresses or names.

3. For each source policy applicable to hosts within the given domain, the syntax and semantics.

4. For each offered service that may be advertised in transit policies, the numeric identifier, syntax, and semantics.

5. For each requested service that may appear within a path setup message, the numeric identifier, syntax, and semantics.

6. For each source user class, the numeric identifier, syntax, and semantics.

2 Control Message Transport Protocol

IDPR control messages convey routing-related information that directly affects the policy routes generated and the paths set up across the Internet. Errors in IDPR control messages can have widespread, deleterious effects on inter-domain policy routing, and so the IDPR protocols have been designed to minimize loss and corruption of control messages. For every control message it transmits, each IDPR protocol expects to receive notification as to whether the control message successfully reached the intended IDPR recipient. Moreover, the IDPR recipient of a control message first verifies that the message appears to be well-formed, before acting on its contents.

All IDPR protocols use the *control message transport protocol* (CMTP), a connectionless, transaction-based transport layer protocol, for communication with intended recipients of control messages. CMTP retransmits unacknowledged control messages and applies integrity and authenticity checks to received control messages.

There are three types of CMTP messages:

datagram Contains IDPR control messages.

ack Positive acknowledgement in response to a **datagram** message.

nak Negative acknowledgement in response to a **datagram** message.

Each CMTP message contains several pieces of information supplied by the sender that allow the recipient to test the integrity and authenticity of the message. The set of integrity and authenticity checks performed after CMTP message reception are collectively referred to as the *validation checks* and are described in section 2.3.

When we first designed the IDPR protocols, CMTP as a distinct protocol did not exist. Instead, CMTP-equivalent functionality was embedded in each IDPR protocol. To provide a cleaner implementation, we later decided to provide a single transport protocol that could be used by all IDPR protocols. We originally considered using an existing transport protocol, but rejected this approach for the following reasons:

1. The existing reliable transport protocols do not provide all of the validation checks, in particular the timestamp and authenticity checks, required by the IDPR protocols. Hence, if we were to use one of these protocols, we would still have to provide a separate protocol on top of the transport protocol to force retransmission of IDPR messages that failed to pass the required validation checks.

2. Many of the existing reliable transport protocols are window-based and hence can result in increased message delay and resource use when, as is the case with IDPR, multiple independent messages use

the same transport connection. A single message experiencing transmission problems and requiring retransmission can prevent the window from advancing, forcing all subsequent messages to queue behind it. Moreover, many of the window-based protocols do not support selective retransmission of failed messages but instead require retransmission of not only the failed message but also all preceding messages within the window.

2.1 Message Transmission

At the transmitting entity, when an IDPR protocol is ready to issue a control message, it passes a copy of the message to CMTP; it also passes a set of parameters to CMTP for inclusion in the CMTP header and for proper CMTP message handling. In turn, CMTP converts the control message and associated parameters into a **datagram** by prepending the appropriate header to the control message. The CMTP header contains several pieces of information to aid the message recipient in detecting errors (see section 2.4). Each IDPR protocol can specify all of the following CMTP parameters applicable to its control message:

1. IDPR protocol and message type.
2. Destination.
3. Integrity/authentication scheme.
4. Timestamp.
5. Maximum number of transmissions allotted.
6. Retransmission interval in microseconds.

One of these parameters, the timestamp, can be specified directly by CMTP as the internal clock time at which the message is transmitted. However, two of the IDPR protocols, namely flooding and path control, themselves require message generation timestamps for proper protocol operation. Thus, instead of requiring CMTP to pass back a timestamp to the IDPR protocol, we simplify the service interface between the two protocols by allowing the IDPR protocol to specify the timestamp in the first place.

Using the control message and accompanying parameters supplied by the IDPR protocol, CMTP constructs a **datagram**, adding to the header CMTP-specific parameters. In particular, CMTP assigns a transaction identifier to each **datagram** generated, used to associate acknowledgements with **datagram** messages. Each **datagram** recipient includes the received transaction identifier in its returned **ack** or **nak**, and each **datagram** sender uses the transaction identifier to match the received **ack** or **nak** with the original **datagram**.

A single **datagram**, for example, a routing information message or a path control message, may be handled by CMTP at many different policy gateways. Within a pair of consecutive IDPR entities, the **datagram** sender expects to receive an acknowledgement from the **datagram** recipient. However, only the IDPR entity that actually generated the original CMTP **datagram** has control over the transaction identifier. The intermediate policy gateways that transmit the **datagram** do not change the transaction identifier. Nevertheless, at each intermediate policy gateway, the transaction identifier must uniquely distinguish the **datagram** so that only one acknowledgement from the next policy gateway matches the original **datagram**.

The transaction identifier consists of the numeric identifiers for the domain and IDPR entity (policy gateway or route server) issuing the original **datagram**, together with a 32-bit local identifier assigned by CMTP operating within that IDPR entity. We recommend implementing the 32-bit local identifier either as a simple counter incremented for each **datagram** generated or as a fine granularity clock. The former always guarantees uniqueness of transaction identifiers; the latter guarantees uniqueness of transaction identifiers, provided the clock granularity is finer than the minimum possible interval between **datagram** generations and the clock wrapping period is longer than the maximum round-trip delay to and from any Internet destination.

Before transmitting a **datagram**, CMTP computes the length of the entire message, taking into account the prescribed integrity/authentication scheme, and then computes the integrity/authentication value over the whole message. CMTP includes both of these quantities, which are crucial for checking message integrity and authenticity at the recipient, in the **datagram** header. After sending a **datagram**, CMTP saves a copy and sets an associated retransmission timer, as directed by the IDPR protocol parameters. If the retransmission timer fires and CMTP has received neither an **ack** nor a **nak** for the **datagram**, CMTP then retransmits the datagram, provided this retransmission does not exceed the transmission allotment. Whenever a **datagram** exhausts its transmission allotment, CMTP discards the **datagram**, informs the IDPR protocol that the control message transmission was not successful, and logs the event for network management. In this case, the IDPR protocol may either resubmit its control message to CMTP, specifying an alternate destination, or discard the control message altogether.

2.2 Message Reception

At the receiving entity, when CMTP obtains a **datagram**, it takes one of the following actions, depending upon the outcome of the message validation checks:

1. The **datagram** passes the CMTP validation checks. CMTP then delivers the **datagram** with enclosed IDPR control message, to the appropriate IDPR protocol, which in turn applies its own integrity checks to the control message before acting on the contents. The recipient IDPR protocol,

except in one case,[2] directs CMTP to generate an **ack** and return the **ack** to the sender. In addition, the IDPR protocol may pass control information to CMTP for inclusion in the **ack**, depending on the contents of the original control message. For example, a route server unable to fill a request for routing information may inform the requesting IDPR entity to place its request elsewhere, through an **ack** for the initial request.

2. The **datagram** fails at least one of the CMTP validation checks. CMTP then generates a **nak**, returns the **nak** to the sender, and discards the **datagram**, regardless of the type of IDPR control message contained in the **datagram**. The **nak** indicates the nature of the validation failure and serves to help the sender establish communication with the recipient. In particular, the CMTP **nak** provides a mechanism for negotiation of IDPR version and integrity/authentication scheme, two parameters crucial for establishing communication between IDPR entities.

Upon receiving an **ack** or a **nak**, CMTP immediately discards the message if at least one of the validation checks fails or if it is unable to locate the associated **datagram**. CMTP logs the latter event for network management. Otherwise, if all of the validation checks pass and if it is able to locate the associated **datagram**, CMTP clears the associated retransmission timer and then takes one of the following actions, depending upon the message type:

1. The message is an **ack**. CMTP discards the associated **datagram** and delivers the **ack**, which may contain IDPR control information, to the appropriate IDPR protocol.

2. The message is a **nak**. If the associated **datagram** has exhausted its transmission allotment, CMTP discards the **datagram**, informs the appropriate IDPR protocol that the control message transmission was not successful, and logs the event for network management. Otherwise, if the associated **datagram** has not yet exhausted its transmission allotment, CMTP first checks its copy of the **datagram** against the failure indication contained in the **nak**. If its **datagram** copy appears to be intact, CMTP retransmits the **datagram** and sets the associated retransmission timer. However, if its **datagram** copy appears to be corrupted, CMTP discards the **datagram**, informs the IDPR protocol that the control message transmission was not successful, and logs the event for network management.

[2]The up/down protocol (see section 3.2) determines reachability of adjacent policy gateways and does not use CMTP **ack** messages to notify the sender of message reception. Instead, the protocol messages themselves carry implicit information about message reception at the adjacent policy gateway.

2.3 Message Validation

On every CMTP message received, CMTP performs a set of validation checks to test message integrity and authenticity. The order in which these tests are executed is important. CMTP must first determine if it can parse enough of the message to compute the integrity/authentication value. (Refer to section 2.4 for a description of CMTP message formats.) Then, CMTP must immediately compute the integrity/authentication value before checking other header information. An incorrect integrity/authentication value means that the message is corrupted, and so it is likely that CMTP header information is incorrect. Checking specific header fields before computing the integrity/authentication value not only may waste time and resources, but also may lead to incorrect diagnoses of a validation failure.

The CMTP validation checks are as follows:

1. CMTP verifies that it can recognize both the control message version and type contained in the header. Failure to recognize either one of these values means that CMTP cannot continue to parse the message.

2. CMTP verifies that it can recognize and accept the integrity/authentication type contained in the header; no integrity/authentication is not an acceptable type for CMTP.

3. CMTP computes the integrity/authentication value and verifies that it equals the integrity/authentication value contained in the header. For key-based integrity/authentication schemes, CMTP may use the source domain identifier contained in the CMTP header to index the correct key. Failure to index a key means that CMTP cannot compute the integrity/authentication value.

4. CMTP computes the message length in bytes and verifies that it equals the length value contained in the header.

5. CMTP verifies that the message timestamp is in the acceptable range. The message should be no more recent than `cmtp_new` (5) minutes ahead of the entity's current internal clock time.[3] The `cmtp_new` value allows some clock drift between IDPR entities. Moreover, each IDPR protocol has its own limit on the maximum age of its control messages. The message should be no less recent than a prescribed number of minutes behind the entity's current internal clock time. Hence, each IDPR protocol performs its own message timestamp check in addition to that performed by CMTP.

6. CMTP verifies that it can recognize the IDPR protocol designated for the enclosed control message.

[3] In this document, when we present an IDPR system configuration parameter, such as `cmtp_new`, we usually follow it with a recommended value in parentheses.

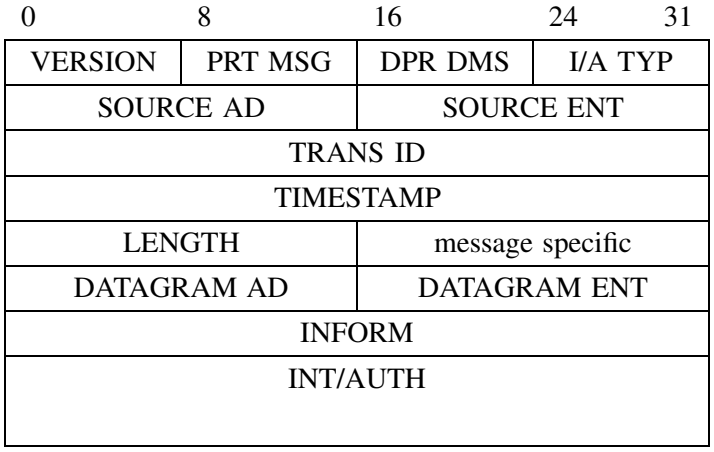
Whenever CMTP encounters a failure while performing any of these validation checks, it logs the event for network management. If the failure occurs on a **datagram**, CMTP immediately generates a **nak** containing the reason for the failure, returns the **nak** to the sender, and discards the **datagram** message. If the failure occurs on an **ack** or a **nak**, CMTP discards the **ack** or **nak** message.

2.4 CMTP Message Formats

In designing the format of IDPR control messages, we have attempted to strike a balance between efficiency of link bandwidth usage and efficiency of message processing. In general, we have chosen compact representations for IDPR information in order to minimize the link bandwidth consumed by IDPR-specific information. However, we have also organized IDPR information in order to speed message processing, which does not always result in minimum link bandwidth usage.

To limit link bandwidth usage, we currently use fixed-length identifier fields in IDPR messages; domains, virtual gateways, policy gateways, and route servers are all represented by fixed-length identifiers. To simplify message processing, we currently align fields containing an even number of bytes on even-byte boundaries within a message. In the future, if the Internet adopts the use of super domains, we will offer hierarchical, variable-length identifier fields in an updated version of IDPR.

The header of each CMTP message contains the following information:



VERSION (8 bits) Version number for IDPR control messages, currently equal to 1.

PRT (4 bits) Numeric identifier for the control message transport protocol, equal to 0 for CMTP.

MSG (4 bits) Numeric identifier for the CMTP message type, equal to 0 for a **datagram**, 1 for an **ack**, and 2 for a **nak**.

DPR (4 bits) Numeric identifier for the original **datagram**'s IDPR protocol type.

DMS (4 bits) Numeric identifier for the original **datagram**'s IDPR message type.

I/A TYP (8 bits) Numeric identifier for the integrity/authentication scheme used. CMTP requires the use of an integrity/authentication scheme; this value must not be set equal to 0, indicating no integrity/authentication in use.

SOURCE AD (16 bits) Numeric identifier for the domain containing the IDPR entity that generated the message.

SOURCE ENT (16 bits) Numeric identifier for the IDPR entity that generated the message.

TRANSACTION ID (32 bits) Local transaction identifier assigned by the IDPR entity that generated the original **datagram**.

TIMESTAMP (32 bits) Number of seconds elapsed since 1 January 1970 0:00 GMT.

LENGTH (16 bits) Length of the entire IDPR control message, including the CMTP header, in bytes.

message specific (16 bits) Dependent upon CMTP message type.

For **datagram** and **ack** messages:

RESERVED (16 bits) Reserved for future use and currently set equal to 0.

For **nak** messages:

ERR TYP (8 bits) Numeric identifier for the type of CMTP validation failure encountered. Validation failures include the following types:

1. Unrecognized IDPR control message version number.
2. Unrecognized CMTP message type.
3. Unrecognized integrity/authentication type.
4. Unacceptable integrity/authentication type.
5. Unable to locate key using source domain.
6. Incorrect integrity/authentication value.
7. Incorrect message length.
8. Message timestamp out of range.
9. Unrecognized IDPR protocol designated for the enclosed control message.

ERR INFO (8 bits) CMTP supplies the following additional information for the designated types of validation failures:

Type 1: Acceptable IDPR version number.

Types 2 and 3: Acceptable integrity/authentication type.

DATAGRAM AD (16 bits) Numeric identifier for the domain containing the IDPR entity that generated the original **datagram**. Present only in **ack** and **nak** messages.

DATAGRAM ENT (16 bits) Numeric identifier for the IDPR entity that generated the original **datagram**. Present only in **ack** and **nak** messages.

INFORM (optional, variable) Information to be interpreted by the IDPR protocol that issued the original **datagram**. Present only in **ack** messages and dependent on the original **datagram**'s IDPR protocol type.

INT/AUTH (variable) Computed integrity/authentication value, dependent on type of integrity/authentication scheme used.

3 Virtual Gateway Protocol

Every policy gateway within a domain participates in gathering information about connectivity within and between virtual gateways of which it is a member and in distributing this information to other virtual gateways in its domain. We refer to these functions collectively as the *virtual gateway protocol* (VGP).

The information collected through VGP has both local and global significance for IDPR. Virtual gateway connectivity information, distributed to policy gateways within a single domain, aids those policy gateways in selecting routes across and between virtual gateways connecting their domain to adjacent domains. Inter-domain connectivity information, distributed throughout the Internet in routing information messages, aids route servers in constructing feasible policy routes.

Provided that a domain contains simple virtual gateway and transit policy configurations, one need only implement a small subset of the VGP functions. The connectivity among policy gateways within a virtual gateway and the heterogeneity of transit policies within a domain determine which VGP functions must be implemented, as we explain toward the end of this section.

3.1 Message Scope

Policy gateways generate VGP messages containing information about perceived changes in virtual gateway connectivity and distribute these messages to other policy gateways within the same domain and within the same virtual gateway. We classify VGP messages into three distinct categories: pair-PG, intra-VG, and inter-VG, depending upon the scope of message distribution.

Policy gateways use CMTP for reliable transport of VGP messages. The issuing policy gateway must communicate to CMTP the maximum number of transmissions per VGP message, `vgp_ret`, and the interval between VGP message retransmissions, `vgp_int` microseconds. The recipient policy gateway must determine VGP message acceptability; conditions of acceptability depend on the type of VGP message, as we describe below.

Policy gateways store, act upon, and in the case of inter-VG messages, forward the information contained in acceptable VGP messages. VGP messages that pass the CMTP validation checks but fail a specific VGP message acceptability check are considered to be unacceptable and are hence discarded by recipient policy gateways. A policy gateway that receives an unacceptable VGP message also logs the event for network management.

3.1.1 Pair-PG Messages

Pair-PG message communication occurs between the two members of a pair of adjacent, peer, or neighbor policy gateways. With IDPR, the only pair-PG messages are those periodically generated by the up/down protocol and used to monitor mutual reachability between policy gateways.

A pair-PG message is *acceptable* if:

1. It passes the CMTP validation checks.
2. Its timestamp is less than `vgp_old` (300) seconds behind the recipient's internal clock time.
3. Its destination policy gateway identifier coincides with the identifier of the recipient policy gateway.
4. Its source policy gateway identifier coincides with the identifier of a policy gateway configured for the recipient's domain or associated virtual gateway.

3.1.2 Intra-VG Messages

Intra-VG message communication occurs between one policy gateway and all of its peers. Whenever a policy gateway discovers that its connectivity to an adjacent or neighbor policy gateway has changed, it issues an intra-VG message indicating the connectivity change to all of its reachable peers. Whenever a policy gateway detects that a previously unreachable peer is now reachable, it issues, to that peer, intra-VG messages indicating connectivity to adjacent and neighbor policy gateways. If the issuing policy gateway fails to receive an analogous intra-VG message from the newly reachable peer within twice the configured VGP retransmission interval, `vgp_int` microseconds, it actively requests the intra-VG message from that peer. These message exchanges ensure that peers maintain a consistent view of each others' connectivity to adjacent and neighbor policy gateways.

An intra-VG message is *acceptable* if:

1. It passes the CMTP validation checks.
2. Its timestamp is less than `vgp_old` (300) seconds behind the recipient's internal clock time.
3. Its virtual gateway identifier coincides with that of a virtual gateway configured for the recipient's domain.

3.1.3 Inter-VG Messages

Inter-VG message communication occurs between one policy gateway and all of its neighbors. Whenever the lowest-numbered operational policy gateway in a set of mutually reachable peers discovers that its virtual gateway's connectivity to the adjacent domain or to another virtual gateway has changed, it issues an inter-VG message indicating the connectivity change to all of its neighbors. Specifically, the policy gateway distributes an inter-VG message to a VG-representative policy gateway (see section 3.1.4 below) in each virtual gateway in the domain. Each VG representative in turn propagates the inter-VG message to each of its peers.

Whenever the lowest-numbered operational policy gateway in a set of mutually peers detects that one or more previously unreachable peers are now reachable, it issues, to the lowest-numbered operational policy gateway in all other virtual gateways, requests for inter-VG information indicating connectivity to adjacent domains and to other virtual gateways. The recipient policy gateways return the requested inter-VG messages to the issuing policy gateway, which in turn distributes the messages to the newly reachable peers. These message exchanges ensure that virtual gateways maintain a consistent view of each others' connectivity, while consuming minimal domain resources in distributing connectivity information.

An inter-VG message contains information about the entire virtual gateway, not just about the issuing policy gateway. Thus, when virtual gateway connectivity changes happen in rapid succession, recipients of the resultant inter-VG messages should be able to determine the most recent message and that message must contain the current virtual gateway connectivity information. To ensure that the connectivity information distributed is consistent and unambiguous, we designate a single policy gateway, namely the lowest-numbered operational peer, for generating and distributing inter-VG messages. It is a simple procedure for a set of mutually reachable peers to determine the lowest-numbered member.

To understand why a single member of a virtual gateway must issue inter-VG messages, consider the following example. Suppose that two peers in a virtual gateway each detect a different connectivity change and generate a separate inter-VG message. Recipients may not be able to determine which message is more recent, as policy gateway internal clocks may not be synchronized to the necessary granularity. Moreover, even if the clocks were synchronized so that recipients could determine message recency, it is possible for each peer to issue its inter-VG message before receiving current information from the other. As a result, neither inter-VG message contains the correct connectivity for the virtual gateway. However, these problems are eliminated if all inter-VG messages are generated by a single peer within a virtual gateway, in particular the lowest-numbered operational policy gateway.

An inter-VG message is *acceptable* if:

1. It passes the CMTP validation checks.

2. Its timestamp is less than `vgp_old` (300) seconds behind the recipient's internal clock time.

3. Its virtual gateway identifier coincides with that of a virtual gateway configured for the recipient's domain.

4. Its source policy gateway identifier represents the lowest numbered operational member of the issuing virtual gateway, reachable from the recipient.

Distribution of intra-VG messages among peers often triggers generation and distribution of inter-VG messages among virtual gateways. Usually, the lowest-numbered operational policy gateway in a virtual gateway generates and distributes an inter-VG message immediately after detecting a change in virtual gateway connectivity, through receipt or generation of an intra-VG message. However, if this policy gateway is also waiting for an intra-VG message from a newly reachable peer, it does not immediately generate and distribute the inter-VG message.

Waiting for intra-VG messages enables the lowest-numbered operational policy gateway in a virtual gateway to gather the most recent connectivity information for inclusion in the inter-VG message. However, under unusual circumstances, the policy gateway may fail to receive an intra-VG message from a newly reachable peer, even after actively requesting such a message. To accommodate this case, VGP uses an upper bound of four times the configured retransmission interval, `vgp_int` microseconds, on the amount of time to wait before generating and distributing an inter-VG message, when receipt of an intra-VG message is pending.

3.1.4 VG Representatives

When distributing an inter-VG message, the issuing policy gateway selects as recipients one neighbor, the *VG representative*, from each virtual gateway in the domain. To be selected as a VG representative, a policy gateway must be reachable from the issuing policy gateway via intra-domain routing. The issuing policy gateway gives preference to neighbors that are members of more than one virtual gateway. Such a neighbor acts as a VG representative for all virtual gateways of which it is a member and restricts inter-VG message distribution as follows: any policy gateway that is a peer in more than one of the represented virtual gateways receives at most one copy of the inter-VG message. This message distribution strategy minimizes the number of message copies required for disseminating inter-VG information.

3.2 Up/Down Protocol

Directly-connected adjacent policy gateways execute the *up/down protocol* to determine mutual reachability. Pairs of peer or neighbor policy gateways can determine mutual reachability through information

provided by the intra-domain routing procedure or through execution of the up/down protocol. In general, we do not recommend implementing the up/down protocol between each pair of policy gateways in a domain, as it results in $O(n^2)$ (where n is the number of policy gateways within the domain) communications complexity. However, if the intra-domain routing procedure is slow to detect connectivity changes or is unable to report reachability at the IDPR entity level, the reachability information obtained through the up/down protocol may well be worth the extra communications cost. In the remainder of this section, we decribe the up/down protocol from the perspective of adjacent policy gateways, but we note that the identical protocol can be applied to peer and neighbor policy gateways as well.

The up/down protocol determines whether the direct connection between adjacent policy gateways is acceptable for data traffic transport. A direct connection is presumed to be *down* (unacceptable for data traffic transport) until the up/down protocol declares it to be *up* (acceptable for data traffic transport). We say that a virtual gateway is *up* if there exists at least one pair of adjacent policy gateways whose direct connection is acceptable for data traffic transport, and that a virtual gateway is *down* if there exists no such pair of adjacent policy gateways.

When executing the up/down protocol, policy gateways exchange **up/down** messages every `ud_per` (1) second. All policy gateways use the same default period of `ud_per` initially and then negotiate a preferred period through exchange of **up/down** messages. A policy gateway reports its desired value for `ud_per` within its **up/down** messages. It then chooses the larger of its desired value and that of the adjacent policy gateway as the period for exchanging subsequent **up/down** messages. Policy gateways also exchange, in **up/down** messages, information about the identity of their respective domain components. This information assists the policy gateways in selecting routes across virtual gateways to partitioned domains.

Each **up/down** message is transported using CMTP and hence is covered by the CMTP validation checks. However, unlike other IDPR control messages, **up/down** messages do not require reliable transport. Specifically, the up/down protocol requires only a single transmission per **up/down** message and never directs CMTP to return an **ack**. As pair-PG messages, **up/down** messages are acceptable under the conditions described in section 3.1.1.

Each policy gateway assesses the state of its direct connection, to the adjacent policy gateway, by counting the number of acceptable **up/down** messages received within a set of consecutive periods. A policy gateway communicates its perception of the state of the direct connection through its **up/down** messages. Initially, a policy gateway indicates the down state in each of its **up/down** messages. Only when the direct connection appears to be up from its perspective does a policy gateway indicate the up state in its **up/down** messages.

A policy gateway can begin to transport data traffic over a direct connection only after both of the

following conditions are satisfied:

1. The policy gateway receives from the adjacent policy gateway at least j acceptable **up/down** messages within the last m consecutive periods. From the recipient policy gateway's perspective, this event constitutes a state transition of the direct connection from down to up. Hence, the policy gateway indicates the up state in its subsequent **up/down** messages.

2. The **up/down** message most recently received from the adjacent policy gateway indicates the up state, signifying that the adjacent policy gateway considers the direct connection to be up as well.

A policy gateway must cease to transport data traffic over a direct connection whenever either of the following conditions is satisfied:

1. The policy gateway receives from the adjacent policy gateway at most k acceptable **up/down** messages within the last n consecutive periods.

2. The **up/down** message most recently received from the adjacent policy gateway indicates the down state, signifying that the adjacent policy gateway considers the direct connection to be down.

From the recipient policy gateway's perspective, either of these events constitutes a state transition of the direct connection from up to down. Hence, the policy gateway indicates the down state in its subsequent **up/down** messages.

3.2.1 Implementation

We recommend implementing the up/down protocol using a sliding window. Each window slot indicates the **up/down** message activity during a given period, containing either a *hit* for receipt of an acceptable **up/down** message or a *miss* for failure to receive an acceptable **up/down** message, within the given period. In addition to the sliding window, the implementation should include a tally of hits recorded during the current period and a tally of misses recorded over the current window.

When the direct connection moves to the down state, the initial values of the up/down protocol parameters must be set as follows:

- The sliding window size is equal to m.
- Each window slot contains a miss.
- The current period hit tally is equal to 0.

- The current window miss tally is equal to m.

When the direct connection moves to the up state, the initial values of the up/down protocol parameters must be set as follows:

- The sliding window size is equal to n.
- Each window slot contains a hit.
- The current period hit tally is equal to 0.
- The current window miss tally is equal to 0.

At the conclusion of each period, a policy gateway computes the miss tally and determines whether there has been a state transition of the direct connection to the adjacent policy gateway. In the down state, a miss tally of no more than $m - j$ signals a transition to the up state. In the up state, a miss tally of no less than $n - k$ signals a transition to the down state.

Computing the correct miss tally involves several steps. First, the policy gateway prepares to slide the window by one slot so that the oldest slot disappears, making room for the newest slot. However, before sliding the window, the policy gateway checks the contents of the oldest window slot. If this slot contains a miss, the policy gateway decrements the miss tally by 1, as this slot is no longer part of the current window.

After sliding the window, the policy gateway initially records a miss in the newest window slot and then determines what the proper slot contents should be. If the hit tally for the current period equals 0, a miss is the correct value for the newest slot, and so the policy gateway increments the miss tally by 1. Otherwise, if the hit tally for the current period is greater than 0, the policy gateway applies the hits to any slot containing a miss, beginning with the newest and progressing to the oldest such slot. For each such slot, the policy gateway records a hit in that slot and decrements the hit tally by 1. If the selected slot is not the newest slot, the hit cancels out an actual miss, and so the policy gateway decrements the miss tally by 1 as well. The policy gateway continues to apply each remaining hit tallied to any slot containing a miss, until either all such hits are exhausted or all such slots are accounted for. Before beginning the next up/down period, the policy gateway resets the hit tally to 0.

Although we expect the hit tally, within any given period, to be no greater than 1, we do anticipate the occasional period in which a policy gateway receives more than one **up/down** message from an adjacent policy gateway. The most common reasons for this occurrence are message delay and clock drift. When an **up/down** message is delayed, the receiving policy gateway observes a miss in one period followed by two hits in the next period, one of which cancels the previous miss. However, excess hits remaining in

the tally after miss cancellation indicate a problem, such as clock drift. Thus, whenever a policy gateway accumulates excess hits, it logs the event for network management.

When clock drift occurs between two adjacent policy gateways, it causes the period of one policy gateway to grow with respect to the period of the other policy gateway. Let p_X be the period for PG X, let p_Y be the period for PG Y, and let g and h be the smallest positive integers such that $gp_X = hp_Y$. Suppose that $p_X < p_Y$ because of clock drift. In this case, PG X observes $g - h$ misses in g consecutive periods, while PG Y observes $g - h$ surplus hits in h consecutive periods. As long as $\frac{g-h}{g} < \frac{n-k}{n}$ and $\frac{g-h}{g} \leq \frac{m-j}{m}$, the clock drift itself will not cause the direct connection to enter or remain in the down state.

3.3 Policy Gateway Connectivity

Policy gateways collect connectivity information through the intra-domain routing procedure and through VGP, and they distribute connectivity changes through VGP in both intra-VG messages to peers and inter-VG messages to neighbors. Locally, this connectivity information assists policy gateways in selecting routes, not only across a virtual gateway to an adjacent domain but also across a domain between two virtual gateways. Moreover, changes in connectivity between domains are distributed, in routing information messages, to route servers throughout the Internet.

3.3.1 Within a Virtual Gateway

Each policy gateway within a virtual gateway constantly monitors its connectivity to all adjacent and to all peer policy gateways. To determine the state of its direct connection to an adjacent policy gateway, a policy gateway uses reachability information supplied by the up/down protocol. To determine the state of its intra-domain routes to a peer policy gateway, a policy gateway uses reachability information supplied by either the intra-domain routing procedure or the up/down protocol.

When a policy gateway detects a change, in state or adjacent domain component, associated with its direct connection to an adjacent policy gateway, or when a policy gateway detects that a previously unreachable peer is now reachable, it generates a **PG connect** message. In the first case, it distributes a copy to each peer reachable via intra-domain routing, and in the second case, it distributes a copy to the newly reachable peer. A **PG connect** message is an intra-VG message that includes information about each adjacent policy gateway directly connected to the issuing policy gateway. Specifically, the **PG connect** message contains the adjacent policy gateway's identifier, status (reachable or unreachable), and domain component identifier. If a **PG connect** message contains a request, each peer that receives the message responds to the sender with its own **PG connect** message.

All mutually reachable peers monitor policy gateway connectivity within their virtual gateway, through the up/down protocol, the intra-domain routing procedure, and the exchange of **PG connect** messages. Within a given virtual gateway, each constituent policy gateway maintains the following information about each configured adjacent policy gateway:

1. The identifier for the adjacent policy gateway.
2. The status of the adjacent policy gateway: reachable/unreachable, directly connected/not directly connected.
3. The local exit interfaces used to reach the adjacent policy gateway, provided it is reachable.
4. The identifier for the adjacent policy gateway's domain component.
5. The set of peers to which the adjacent policy gateway is directly-connected.

Hence, all mutually reachable peers can detect changes in connectivity across the virtual gateway to adjacent domain components.

When the lowest-numbered operational policy gateway within a virtual gateway detects a change in the set of adjacent domain components reachable through direct connections across the given virtual gateway, it generates a **VG connect** message and distributes a copy to a VG representative in all other virtual gateways connected to its domain. A **VG connect** message is an inter-VG message that includes information about each peer's connectivity across the given virtual gateway. Specifically, the **VG connect** message contains, for each peer, its identifier and the identifiers of the domain components reachable through its direct connections to adjacent policy gateways. Moreover, the **VG connect** message gives each recipient enough information to determine the state, up or down, of the issuing virtual gateway.

The issuing policy gateway, namely the lowest-numbered operational peer, may have to wait up to four times `vgp_int` microseconds after detecting the connectivity change, before generating and distributing the **VG connect** message, as described in section 3.1.3. Each recipient VG representative in turn distributes a copy of the **VG connect** message to each of its peers reachable via intra-domain routing. If a **VG connect** message contains a request, then in each recipient virtual gateway, the lowest-numbered operational peer that receives the message responds to the original sender with its own **VG connect** message.

3.3.2 Between Virtual Gateways

At present, we expect transit policies to be uniform over all intra-domain routes between any pair of policy gateways within a domain. However, when tariffed qualities of service become prevalent offerings

for intra-domain routing, we can no longer expect uniformity of transit policies throughout a domain. To monitor the transit policies supported on intra-domain routes between virtual gateways requires both a policy-sensitive intra-domain routing procedure and a VGP exchange of policy information between neighbor policy gateways.

Each policy gateway within a domain constantly monitors its connectivity to all peer and neighbor policy gateways, including the transit policies supported on intra-domain routes to these policy gateways. To determine the state of its intra-domain connection to a peer or neighbor policy gateway, a policy gateway uses reachability information supplied by either the intra-domain routing procedure or the up/down protocol. To determine the transit policies supported on intra-domain routes to a peer or neighbor policy gateway, a policy gateway uses policy-sensitive reachability information supplied by the intra-domain routing procedure. We note that when transit policies are uniform over a domain, reachability and policy-sensitive reachability are equivalent.

Within a virtual gateway, each constituent policy gateway maintains the following information about each configured peer and neighbor policy gateway:

1. The identifier for the peer or neighbor policy gateway.
2. The identifiers corresponding to the transit policies configured to be supported by intra-domain routes to the peer or neighbor policy gateway.
3. For each transit policy, the status of the peer or neighbor policy gateway: reachable/unreachable.
4. For each transit policy, the local exit interfaces used to reach the peer or neighbor policy gateway, provided it is reachable.
5. The identifiers for the adjacent domain components reachable through direct connections from the peer or neighbor policy gateway, obtained through **VG connect** messages.

Using this information, a policy gateway can detect changes in its connectivity to a neighboring domain component, with respect to a given transit policy and through a given neighbor. Moreover, combining the information obtained for all neighbors within a given virtual gateway, the policy gateway can detect changes in its connectivity, with respect to a given transit policy, to another virtual gateway and to adjacent domain components reachable through that virtual gateway.

All policy gateways mutually reachable via intra-domain routes supporting a configured transit policy need not exchange information about perceived changes in connectivity, with respect to the given transit policy. In this case, each policy gateway can infer another's policy-sensitive reachability to a third, through mutual peer intra-domain reachability information provided by the intra-domain routing procedure. However, whenever two or more policy gateways are no longer mutually reachable with respect to

a given transit policy, these policy gateways can no longer infer each other's reachability to other policy gateways, with respect to that transit policy. In this case, these policy gateways must exchange explicit information about changes in connectivity to other policy gateways, with respect to that transit policy.

When a policy gateway detects a change in its connectivity to another virtual gateway, with respect to a configured transit policy, or to an adjacent domain component reachable through that virtual gateway, or when a policy gateway detects that a previously unreachable peer is now reachable, it generates a **PG policy** message. In the first case, it distributes a copy to each peer reachable via intra-domain routing but not currently reachable via any intra-domain routes of the given transit policy, and in the second case, it distributes a copy to the newly reachable peer. A **PG policy** message is an intra-VG message that includes information about each configured transit policy and each virtual gateway configured to be reachable from the issuing policy gateway via intra-domain routes of the given transit policy. Specifically, the **PG policy** message contains, for each configured transit policy:

1. The identifier of the transit policy.
2. The identifiers of the virtual gateways associated with the given transit policy and currently reachable, with respect to that transit policy, from the issuing policy gateway.
3. The identifiers of the domain components reachable from and adjacent to the members of the given virtual gateways.

If a **PG policy** message contains a request, each peer that receives the message responds to the original sender with its own **PG policy** message.

In addition to connectivity between itself and its neighbors, each policy gateway also monitors the connectivity, between domain components adjacent to its virtual gateway and domain components adjacent to other virtual gateways, through its domain and with respect to the configured transit policies. For each member of each of its virtual gateways, a policy gateway monitors:

1. The set of adjacent domain components currently reachable through direct connections across the given virtual gateway. The policy gateway obtains this information through **PG connect** messages from reachable peers and through **up/down** messages from adjacent policy gateways.
2. For each configured transit policy, the set of virtual gateways currently reachable from the given virtual gateway with respect to that transit policy and the set of neighboring domain components currently reachable through direct connections across those virtual gateways. The policy gateway obtains this information through **PG policy** messages from peers, **VG connect** messages from neighbors, and the intra-domain routing procedure. Using this information, a policy gateway can

detect connectivity changes, through its domain and with respect to a given transit policy, between neighboring domain components.

When the lowest-numbered operational policy gateway within a virtual gateway detects a change in the connectivity between a domain component adjacent to its virtual gateway and a domain component adjacent to another virtual gateway in its domain, with respect to a configured transit policy, it generates a **VG policy** message and distributes a copy to a VG representative in selected virtual gateways connected to its domain. In particular, the lowest-numbered operational policy gateway distributes a **VG policy** message to a VG representative in every other virtual gateway containing a member reachable via intra-domain routing but not currently reachable via any routes of the given transit policy. A **VG policy** message is an inter-VG message that includes information about the connectivity between domain components adjacent to the issuing virtual gateway and domain components adjacent to the other virtual gateways in the domain, with respect to configured transit policies. Specifically, the **VG policy** message contains, for each transit policy:

1. The identifier of the transit policy.
2. The identifiers of the virtual gateways associated with the given transit policy and currently reachable, with respect to that transit policy, from the issuing virtual gateway.
3. The identifiers of the domain components reachable from and adjacent to the members of the given virtual gateways.

The issuing policy gateway, namely the lowest-numbered operational peer, may have to wait up to four times `vgp_int` microseconds after detecting the connectivity change, before generating and distributing the **VG policy** message, as described in section 3.1.3. Each recipient VG representative in turn distributes a copy of the **VG policy** message to each of its peers reachable via intra-domain routing. If a **VG policy** message contains a request, then in each recipient virtual gateway, the lowest-numbered operational peer that receives the message responds to the original sender with its own **VG policy** message.

3.3.3 Communication Complexity

We offer an example, to provide an estimate of the number of VGP messages exchanged within a domain, $AD\ X$, after a detected change in policy gateway connectivity. Suppose that an adjacent domain, $AD\ Y$, partitions such that the partition is detectable through the exchange of **up/down** messages across a virtual gateway connecting $AD\ X$ and $AD\ Y$. Let V be the number of virtual gateways in $AD\ X$, and let P be the number of peer policy gateways within each virtual gateway. Within $AD\ X$, the detected partition will result in the following VGP message exchanges:

1. $P - 1$ policy gateways each receive one **PG connect** message. The policy gateway detecting the adjacent domain partition generates a **PG connect** message and distributes it to each peer in the virtual gateway.

2. $P(V-1)$ policy gateways each receive one **VG connect** message. The lowest-numbered operational policy gateway in the virtual gateway detecting the partition of the adjacent domain generates a **VG connect** message and distributes it to a VG representative in all other virtual gateways connected to the domain. In turn, each VG representative distributes the **VG connect** message to each peer within its virtual gateway.

3. $P(V-1)$ policy gateways each receive at most $P-1$ **PG policy** messages, and only if the domain has more than a single, uniform transit policy. Each policy gateway in each virtual gateway generates a **PG policy** message and distributes it to all reachable peers not currently reachable with respect to the given transit policy.

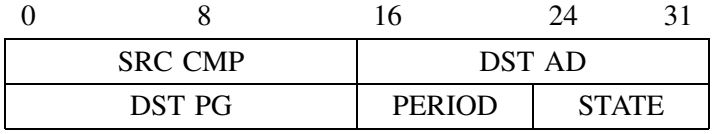
4. PV policy gateways each receive at most $V - 1$ **VG policy** messages, and only if the domain has more than a single, uniform transit policy. The lowest-numbered operational policy gateway in each virtual gateway generates a **VG policy** message and distributes it to a VG representative in all other virtual gateways containing at least one reachable member not currently reachable with respect to the given transit policy. In turn, each VG representative distributes a **VG policy** message to each peer within its virtual gateway.

3.4 VGP Message Formats

The virtual gateway protocol number is equal to 0. We describe the contents of each type of VGP message below.

3.4.1 Up/Down

The **up/down** message type is equal to 0.



SRC CMP (16 bits) Numeric identifier for the domain component containing the issuing policy gateway.

DST AD (16 bits) Numeric identifier for the destination domain.

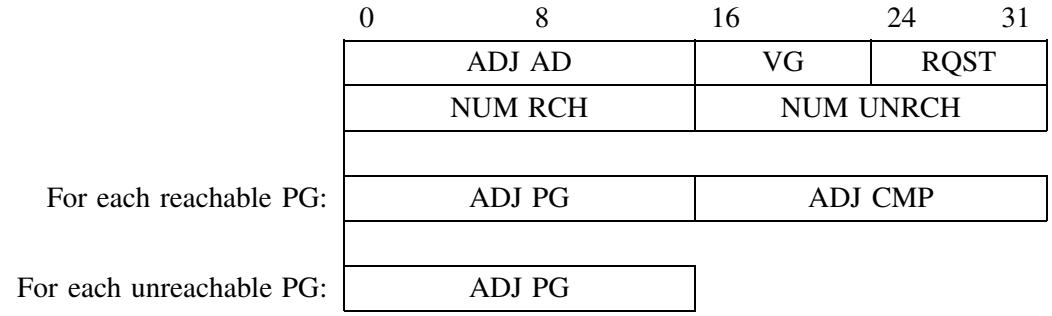
DST PG (16 bits) Numeric identifier for the destination policy gateway.

PERIOD (8 bits) Length in seconds of the **up/down** message generation period.

STATE (8 bits) Perceived state (1 up, 0 down) of the direct connection in the direction from the destination policy gateway to the issuing policy gateway, contained in the right-most bit.

3.4.2 PG Connect

The **PG connect** message type is equal to 1. **PG connect** messages are not required for any virtual gateway containing exactly two policy gateways.



ADJ AD (16 bits) Numeric identifier for the adjacent domain.

VG (8 bits) Numeric identifier for the virtual gateway associated with the adjacent domain.

RQST (8 bits) Request for a **PG connect** message (1 request, 0 no request) from each recipient peer, contained in the right-most bit.

NUM RCH (16 bits) Number of adjacent policy gateways within the virtual gateway, which are directly-connected to and currently reachable from the issuing policy gateway.

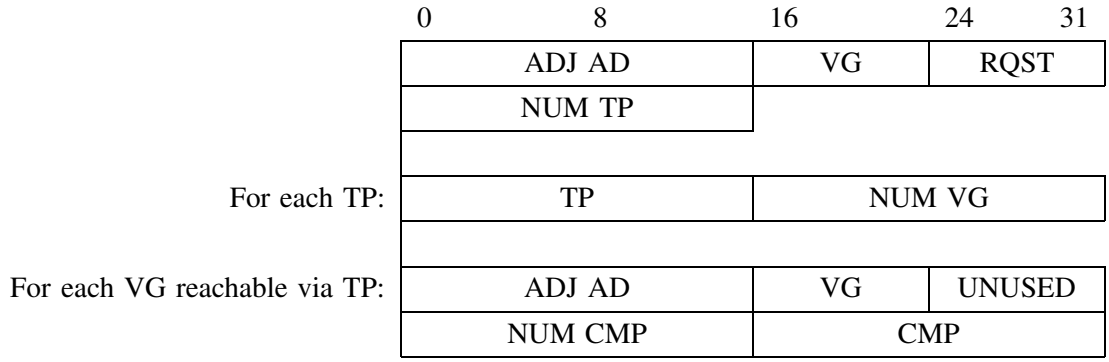
NUM UNRCH (16 bits) Number of adjacent policy gateways within the virtual gateway, which are directly-connected to but not currently reachable from the issuing policy gateway.

ADJ PG (16 bits) Numeric identifier for a directly-connected adjacent policy gateway.

ADJ CMP (16 bits) Numeric identifier for the domain component containing the reachable, directly-connected adjacent policy gateway.

3.4.3 PG Policy

The **PG policy** message type is equal to 2. **PG policy** messages are not required for any virtual gateway containing exactly two policy gateways or for any domain with a single, uniform transit policy.



ADJ AD (16 bits) Numeric identifier for the adjacent domain.

VG (8 bits) Numeric identifier for the virtual gateway associated with the adjacent domain.

RQST (8 bits) Request for a **PG policy** message (1 request, 0 no request) from each recipient peer, contained in the right-most bit.

NUM TP (8 bits) Number of transit policies configured to include the virtual gateway.

TP (16 bits) Numeric identifier for a transit policy associated with the virtual gateway.

NUM VG (16 bits) Number of virtual gateways reachable from the issuing policy gateway, via intra-domain routes supporting the transit policy.

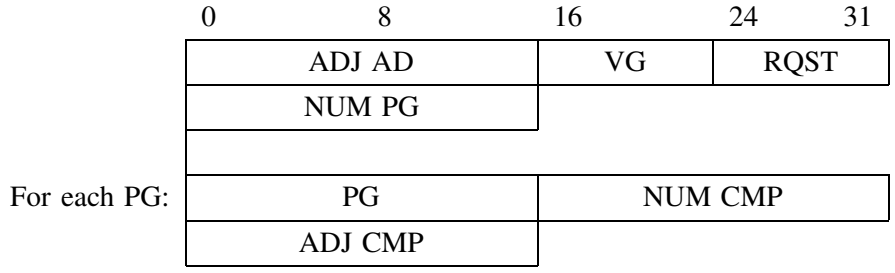
UNUSED (8 bits) Not currently used; must be set equal to 0.

NUM CMP (16 bits) Number of adjacent domain components reachable via direct connections through the virtual gateway.

CMP (16 bits) Numeric identifier for a reachable adjacent domain component.

3.4.4 VG Connect

The **VG connect** message type is equal to 3.



ADJ AD (16 bits) Numeric identifier for the adjacent domain.

VG (8 bits) Numeric identifier for the virtual gateway associated with the adjacent domain.

RQST (8 bits) Request for a **VG connect** message (1 request, 0 no request) from a recipient in all other virtual gateways, contained in the right-most bit.

NUM PG (16 bits) Number of mutually-reachable peer policy gateways in the virtual gateway.

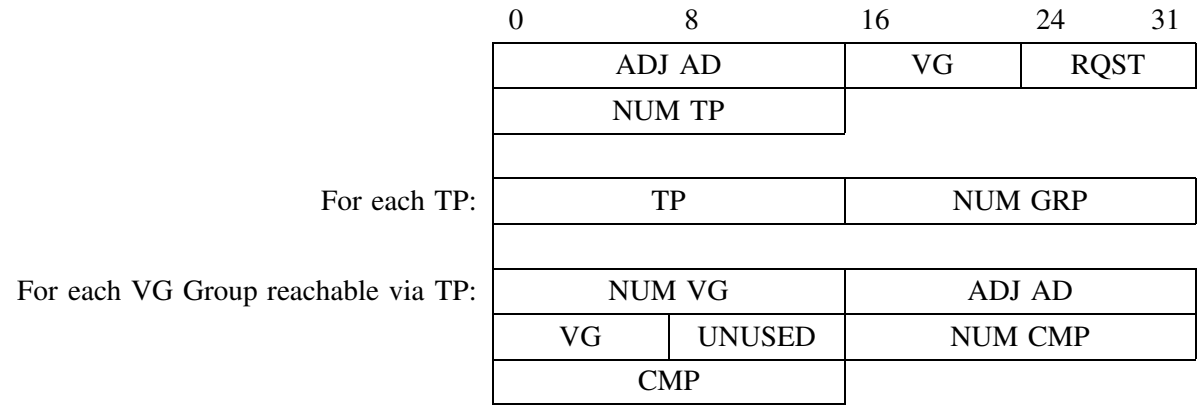
PG (16 bits) Numeric identifier for a peer policy gateway.

NUM CMP (16 bits) Number of components of the adjacent domain reachable via direct connections from the policy gateway.

ADJ CMP (16 bits) Numeric identifier for a reachable adjacent domain component.

3.4.5 VG Policy

The **VG policy** message type is equal to 4. **VG policy** messages are not required for any domain with a single, uniform transit policy.



ADJ AD (16 bits) Numeric identifier for the adjacent domain.

VG (8 bits) Numeric identifier for a virtual gateway associated with the adjacent domain.

RQST (8 bits) Request for a **VG policy** message (1 request, 0 no request) from a recipient in all other virtual gateways, contained in the right-most bit.

NUM TP (16 bits) Number of transit policies configured to include the virtual gateway.

TP (16 bits) Numeric identifier for a transit policy associated with the virtual gateway.

NUM GRP (16 bits) Number of groups of virtual gateways, such that all members in a group are reachable from the issuing virtual gateway via intra-domain routes supporting the given transit policy.

NUM VG (16 bits) Number of virtual gateways in a virtual gateway group.

UNUSED (8 bits) Not currently used; must be set equal to 0.

NUM CMP (16 bits) Number of adjacent domain components reachable via direct connections through the virtual gateway.

CMP (16 bits) Numeric identifier for a reachable adjacent domain component.

Normally, each **VG policy** message will contain a single virtual gateway group. However, if the issuing virtual gateway becomes partitioned such that peers are mutually reachable with respect to some transit policies but not others, virtual gateway groups may be necessary. For example, let $PG\ X$ and $PG\ Y$ be two peers in VG_1. Suppose that $PG\ X$ and $PG\ Y$ are reachable with respect to transit policy A but not with respect to transit policy B. Furthermore, suppose that $PG\ X$ can reach members of VG_2 via intra-domain routes of transit policy B and that $PG\ Y$ can reach members of VG_3 via intra-domain routes of transit policy B. Then the entry in the **VG policy** message issued by VG_1 will include, for transit policy B, two groups of virtual gateways, one containing VG_2 and one containing VG_3.

3.4.6 Negative Acknowledgements

When a policy gateway receives an unacceptable VGP message that passes the CMTP validation checks, it includes, in its CMTP **ack**, an appropriate VGP negative acknowledgement. This information is placed in the INFORM field of the CMTP **ack** (described in section 2.4); the numeric identifier for each type of VGP negative acknowledgement is contained in the left-most 8 bits of the INFORM field. Negative acknowledgements associated with VGP include the following types:

1. Unrecognized VGP message type. Numeric identifier for the unrecognized message type (8 bits).
2. Out-of-date VGP message.
3. Unrecognized virtual gateway source. Numeric identifier for the unrecognized virtual gateway including adjacent administrative domain (16 bits) and local identifier (8 bits).

4 Routing Information Distribution

Each domain participating in IDPR generates and distributes its routing information messages to route servers throughout the Internet. IDPR routing information messages contain information about the transit policies in effect across the given domain and the virtual gateway connectivity to adjacent domains. Route servers in turn use IDPR routing information to generate policy routes between source and destination domains.

There are two different procedures for distributing IDPR routing information: the flooding protocol and the route server query protocol. With the flooding protocol, a representative policy gateway in each domain floods its routing information messages to all other domains. With the route server query protocol, a policy gateway or route server requests routing information from another route server, which in turn responds with routing information from its database. The route server query protocol can be used for quickly updating the routing information maintained by a policy gateway or route server that has just been connected or reconnected to the Internet. In this section, we describe the flooding protocol only; in section 5, we describe the route server query protocol.

Policy gateways and route servers use CMTP for reliable transport of IDPR routing information messages flooded between peer, neighbor, and adjacent policy gateways and between those policy gateways and route servers. The issuing policy gateway must communicate to CMTP the maximum number of transmissions per routing information message, `flood_ret`, and the interval between routing information message retransmissions, `flood_int` microseconds. The recipient policy gateway or route server must determine routing information message acceptability, as we describe in section 4.2.3 below.

4.1 AD Representatives

We designate a single policy gateway, the *AD representative*, for generating and distributing IDPR routing information about its domain, to ensure that the routing information distributed is consistent and unambiguous and to minimize the communication required for routing information distribution. There is usually only a single AD representative per domain, namely the lowest-numbered operational policy gateway in the domain. Within a domain, policy gateways need no explicit election procedure to determine the AD representative. Instead, all members of a set of policy gateways mutually reachable via intra-domain routes can agree on set membership and therefore on which member has the lowest number.

A partitioned domain has as many AD representatives as it does domain components. In fact, the numeric identifier for an AD representative is identical to the numeric identifier for a domain component. One cannot normally predict when and where a domain partition will occur, and thus any policy gateway within a domain may become an AD representative at any time. To prepare for the role of AD repre-

sentative in the event of a domain partition, every policy gateway must continually monitor its domain's IDPR routing information, through VGP and through the intra-domain routing procedure.

4.2 Flooding Protocol

An AD representative policy gateway uses unrestricted flooding among all domains to distribute its domain's IDPR routing information messages to route servers in the Internet. There are two kinds of IDPR routing information messages issued by each AD representative: **configuration** and **dynamic** messages. Each **configuration** message contains the transit policy information configured by the domain administrator, including for each transit policy, its identifier, its specification, and the set of virtual gateways configured as mutually reachable via intra-domain routes supporting the given transit policy. Each **dynamic** message contains information about current virtual gateway connectivity to adjacent domains and about which members of the sets of virtual gateways are at present mutually reachable via intra-domain routes supporting the configured transit policies.

The IDPR flooding protocol is similar to the flooding procedures described in [8]-[10]. Through flooding, the AD representative distributes its routing information messages to route servers in its own domain and in adjacent domains. After generating a routing information message, the AD representative distributes a copy to each of its peers, to a selected VG representative (see section 3.1.4) in all other virtual gateways connected to the domain, and to each route server to which it has been configured to deliver routing information. We recommend that for each route server not contained within a policy gateway, the domain administrator should configure at least two distinct policy gateways to deliver routing information to that route server. Thus, the route server will continue to receive routing information messages, even when one of its associated policy gateways becomes unreachable; the route server will, however, normally receive duplicate copies of a routing information message.

Each VG representative in turn distributes a copy of the routing information message to each member of its configured set of route servers and to each of its peers. We note that distribution of routing information messages among virtual gateways and among peers within a virtual gateway is identical to distribution of inter-VG messages in VGP, as described in section 3.1.3.

Within a virtual gateway, each policy gateway distributes a copy of the routing information message to each member of its configured set of route servers and to certain directly-connected adjacent policy gateways, selected as follows. Each policy gateway knows, through information provided by VGP, which peers have direct connections to which components of the adjacent domain. Only when it is the lowest-numbered operational peer with a direct connection to a given adjacent domain component does a policy gateway distribute a routing information message to a directly-connected adjacent policy gateway in that domain component. If the policy gateway has direct connections to more than one adjacent policy

gateway in that domain component, it selects the routing information message recipient according the order in which the adjacent policy gateways appear in its database, choosing the first encountered. This selection procedure ensures that a copy of the routing information message reaches each component of the adjacent domain, while limiting the number of copies distributed across the virtual gateway.

Once a routing information message reaches an adjacent policy gateway, that policy gateway distributes copies of the message throughout its domain. The adjacent policy gateway, acting as the first recipient of the routing information message in its domain, follows the same message distribution procedure as the AD representative in the source domain, as described above. The flooding procedure terminates when all reachable route servers in the Internet receive a copy of the routing information message.

Neighbor policy gateways may receive copies of the same routing information message from different neighboring domains. If two neighbor policy gateways receive the message copies simultaneously, they will distribute them to VG representatives in other virtual gateways within their domain, resulting in duplicate message distribution. However, each policy gateway stops the spread of duplicate routing information messages as soon as it detects them, as described in section 4.2.3 below. Moreover, we expect simultaneous message receptions to be the exception rather than the rule, given the hierarchical structure of the current Internet topology.

4.2.1 Message Generation

The AD representative generates and distributes a **configuration** message whenever there is a change in a transit policy or virtual gateway configured for its domain. This ensures that route servers maintain an up-to-date view of the domain's configured transit policies and adjacencies. The AD representative may also distribute a **configuration** message at a configurable period of `conf_per` (500) hours. A **configuration** message contains, for each configured transit policy, the identifier assigned by the domain administrator, the specification, and the set of associated virtual gateway groups. Each virtual gateway group comprises virtual gateways configured to be mutually reachable via intra-domain routes of the given transit policy. Accompanying each virtual gateway listed is an indication of whether the virtual gateway is configured to be a domain entry point, a domain exit point, or both according to the given transit policy. The **configuration** message also contains the set of local route servers that the domain administrator has configured to be available to IDPR clients in other domains.

The AD representative generates and distributes a **dynamic** message whenever there is a change in transit policy currently supported across the given domain or in current virtual gateway connectivity to an adjacent domain. This ensures that route servers maintain an up-to-date view of supported transit policies and existing domain adjacencies and how they differ from those configured for the domain.

Specifically, the AD representative generates a **dynamic** message whenever there is a change in the connectivity, through the given domain and with respect to a configured transit policy, between two neighboring domain components. The AD representative may also distribute a **dynamic** message at a configurable period of `dyn_per` (24) hours. A **dynamic** message contains, for each configured transit policy, its identifier, associated virtual gateway groups, and domain components reachable through virtual gateways in each group. Each **dynamic** message also contains the set of currently *unavailable*, either down or unreachable, virtual gateways in the domain.

We note that each virtual gateway group expressed in a **dynamic** message may be a proper subset of one of the corresponding virtual gateway groups expressed in a **configuration** message. For example, suppose that, for a given domain, the virtual gateway group $(VG_1, \ldots, VG_5)$ were configured for a transit policy such that each virtual gateway was both a domain entry and exit point. Thus, all virtual gateways in this group are configured to be mutually reachable via intra-domain routes of the given transit policy. Now suppose that VG_5 becomes unreachable because of a power failure and furthermore that the remaining virtual gateways form two distinct groups, (VG_1, VG_2) and (VG_3, VG_4), such that although virtual gateways in both groups are still mutually reachable via some intra-domain routes they are no longer mutually reachable via any intra-domain routes of a given transit policy. In this case, the virtual gateway groups for the given transit policy now become (VG_1, VG_2) and (VG_3, VG_4); VG_5 is listed as an unavailable virtual gateway.

A route server uses information about the set of unavailable virtual gateways to determine which of its routes are no longer viable, and it subsequently removes such routes from its route database. Although route servers could determine the set of unavailable virtual gateways using information about configured virtual gateways and currently reachable virtual gateways, the associated processing cost is high. In particular, a route server would have to examine all virtual gateway groups listed in a **dynamic** message to determine whether there are any unavailable virtual gateways in the given domain. To reduce the message processing at each route server, we have chosen to include the set of unavailable virtual gateways in each **dynamic** message.

In order to construct a **dynamic** message, the AD representative assembles information gathered from intra-domain routing and from VGP. Specifically, the AD representative uses the following information:

1. **VG connect** and **up/down** messages to determine the state, up or down, of each of its domain's virtual gateways and the adjacent domain components reachable through a given virtual gateway.
2. Intra-domain routing information to determine, for each of its domain's transit policies, whether a given virtual gateway in the domain is reachable with respect to that transit policy.
3. **VG policy** messages to determine the connectivity of neighboring domain components, across the given domain and with respect to a configured transit policy, such that these components are

adjacent to virtual gateways not currently reachable from the AD representative's virtual gateway according to the given transit policy.

4.2.2 Sequence Numbers

Each IDPR routing information message carries a sequence number which, when used in conjunction with the timestamp carried in the CMTP message header, determines the recency of the message. The AD representative assigns a sequence number to each routing information message it generates, depending upon its internal clock time:

1. The AD representative sets the sequence number to 0, if its internal clock time is greater than the timestamp in its previously generated routing information message.
2. The AD representative sets the sequence number to 1 greater than the sequence number in its previously generated routing information message, if its internal clock time equals the timestamp for its previously generated routing information message and if the previous sequence number is less than the maximum value. If the previous sequence number equals the maximum value, the AD representative waits until its internal clock time exceeds the timestamp in its previously generated routing information message and then sets the sequence number to 0.

In general, we do not expect generation of multiple distinct IDPR routing information messages carrying identical timestamps, and so the sequence number may seem superfluous. However, the sequence number may become necessary during synchronization of the AD representative's internal clock. In particular, the AD representative may need to freeze the clock value during synchronization, and thus distinct sequence numbers serve to distinguish routing information messages generated during the clock synchronization interval.

4.2.3 Message Acceptance

Prior to a policy gateway forwarding a routing information message or a route server incorporating routing information into its routing information database, the policy gateway or route server assesses routing information message acceptability. An IDPR routing information message is *acceptable* if:

1. It passes the CMTP validation checks.
2. Its timestamp is less than `conf_old` (530) hours behind the recipient's internal clock time for configuration messages and less than `dyn_old` (25) hours behind the recipient's internal clock time for dynamic messages.

3. Its timestamp and sequence number indicate that it is more recent than the currently-stored routing information from the given domain. If there is no routing information currently stored from the given domain, then the routing information message contains, by default, the more recent information.

Policy gateways acknowledge and forward acceptable IDPR routing information messages, according to the flooding protocol described in section 4.2 above. Moreover, each policy gateway retains the timestamp and sequence number for the most recently accepted routing information message from each domain and uses these values to determine acceptability of routing information messages received in the future. Route servers acknowledge the receipt of acceptable routing information messages and incorporate the contents of these messages into their routing information databases, contingent upon criteria discussed in section 4.2.4 below; however, they do not participate in the flooding protocol. We note that when a policy gateway or route server first returns to service, it immediately updates its routing information database with routing information obtained from another route server, using the route server query protocol described in section 5.

An AD representative takes special action upon receiving an acceptable routing information message, supposedly generated by itself but originally obtained from a policy gateway or route server other than itself. There are at least three possible reasons for the occurrence of this event:

1. The routing information message has been corrupted in a way that is not detectable by the integrity/authentication value.
2. The AD representative has experienced a memory error.
3. Some other entity is generating routing information messages on behalf of the AD representative.

In any case, the AD representative logs the event for network management. Moreover, the AD representative must reestablish its own routing information messages as the most recent for its domain. To do so, the AD representative waits until its internal clock time exceeds the value of the timestamp in the received routing information message and then generates a new routing information message using the currently-stored domain routing information supplied by VGP and by the intra-domain routing procedure. Note that the length of time the AD representative must wait to generate the new message is at most `cmtp_new` (5) minutes, the maximum CMTP-tolerated difference between the received message's timestamp and the AD representative's internal clock time.

IDPR routing information messages that pass the CMTP validity checks but appear less recent than stored routing information are unacceptable. Policy gateways do not forward unacceptable routing information messages, and route servers do not incorporate the contents of unacceptable routing information

messages into their routing information databases. Instead, the recipient of an unacceptable routing information message acknowledges the message in one of two ways:

1. If the routing information message timestamp and sequence number are equal to the timestamp and sequence number associated with the stored routing information for the given domain, the recipient assumes that the routing information message is a duplicate and acknowledges the message.

2. If the routing information message timestamp and sequence number indicate that the message is less recent than the stored routing information for the domain, the recipient acknowledges the message with an indication that it is out-of-date. Such a negative acknowledgement is a signal to the sender of the routing information message to request more up-to-date routing information from a route server, using the route server query protocol. Furthermore, if the recipient of the out-of-date routing information message is a route server, it regenerates a routing information message from its own routing information database and forwards the message to the sender. The sender may in turn propagate this more recent routing information message to other policy gateways and route servers.

4.2.4 Message Incorporation

A route server usually stores the entire contents of an acceptable IDPR routing information message in its routing information database, so that it has access to all advertised transit policies when generating a route and so that it can regenerate the routing information message at a later point in time if requested to do so by another route server or policy gateway. However, the route server may elect not to store all routing information message contents. In particular, the route server need not store any transit policy that excludes the route server's domain as a source or any routing information from a domain that the route server's domain's source policies exclude for transit. Selective storing of routing information message contents simplifies the route generation procedure since it reduces the search space of possible routes, and it limits the amount of route server memory devoted to routing information. However, selective storing of routing information also means that the route server cannot always regenerate the original routing information message, if requested to do so by another route server or policy gateway.

An acceptable IDPR routing information message may contain transit policy information that is not well-defined according to the route server's perception. A **configuration** message may contain an unrecognized domain, virtual gateway, or other attribute, such as user class or offered service. In this case, unrecognized means that the value in the routing information message is not listed in the route server's configuration database, as described in section 1.8.2. A **dynamic** message may contain an unrecognized transit policy or virtual gateway. In this case, unrecognized means that the transit policy or virtual gateway was not listed in the most recent **configuration** message for the given domain.

Each route server can always parse an acceptable routing information messsage, even if some of the information is not well-defined, and thus can always use the information that it does recognize. Therefore, a route server can store the contents of acceptable routing information messages from domains in which it is interested, regardless of whether all contents appear to be well-defined at present. In this case, the route server attempts to obtain the additional information it needs to decipher unrecognized information. For a **configuration** message, the route server requests updated configuration information; for a **dynamic** message, the route server requests, from another route server, the most recent **configuration** message for the given domain.

When a domain is partitioned, each domain component has its own AD representative, which generates routing information messages on behalf of that component. Discovery of a domain partition prompts the AD representative for each domain component to generate and distribute a **dynamic** message. In this case, a route server receives and stores more than one routing information message at a time for the given domain, namely one for each domain component. When the partition heals, the AD representative for the entire domain generates and distributes a **dynamic** message. In each route server's routing information database, the new **dynamic** message does not automatically replace all of the currently-stored **dynamic** messages for the given domain. Instead, the new message only replaces that message whose AD representative matches the AD representative for the new message. The other **dynamic** messages remaining from the period during which the partition occurred will be removed from the routing information database when they attain their maximum age, as decribed in section 4.2.5 below. In a future version of IDPR, we may include mechanisms for removing partition-related **dynamic** messages immediately after the partition disappears.

4.2.5 Routing Information Database

We expect that most of the IDPR routing information stored in a routing information database will remain viable for long periods of time, perhaps until a domain reconfiguration occurs. However, to reduce the probability of retaining stale routing information, a route server imposes a maximum lifetime on each database entry, initialized when it incorporates an accepted entry into its routing information database. The maximum routing information database entry lifetime should be longer than the corresponding routing information message generation period, so that the database entry is likely to be refreshed before it expires.

Each **configuration** message stored in the routing information database remains viable for a maximum of `conf_old` (530) hours; each **dynamic** message stored in the routing information database remains viable for a maximum of `dyn_old` (25) hours. By viable, we mean that the message contents may be used in generating policy routes. Configuring periodic generation of routing information messages makes

it unlikely that any routing information message will remain in a routing information database for its full life span. However, a routing information message may attain its maximum age in a route server that is separated from the Internet for a long period of time.

When an IDPR routing information message attains its maximum age in a routing information database, the route server removes the message contents from its database, so that it will not generate new routes with the outdated routing information nor distribute old routing information in response to requests from other route servers or policy gateways. Nevertheless, the route server continues to dispense routes previously generated with the old routing information, as long as path setup (see section 7) for these routes succeeds.

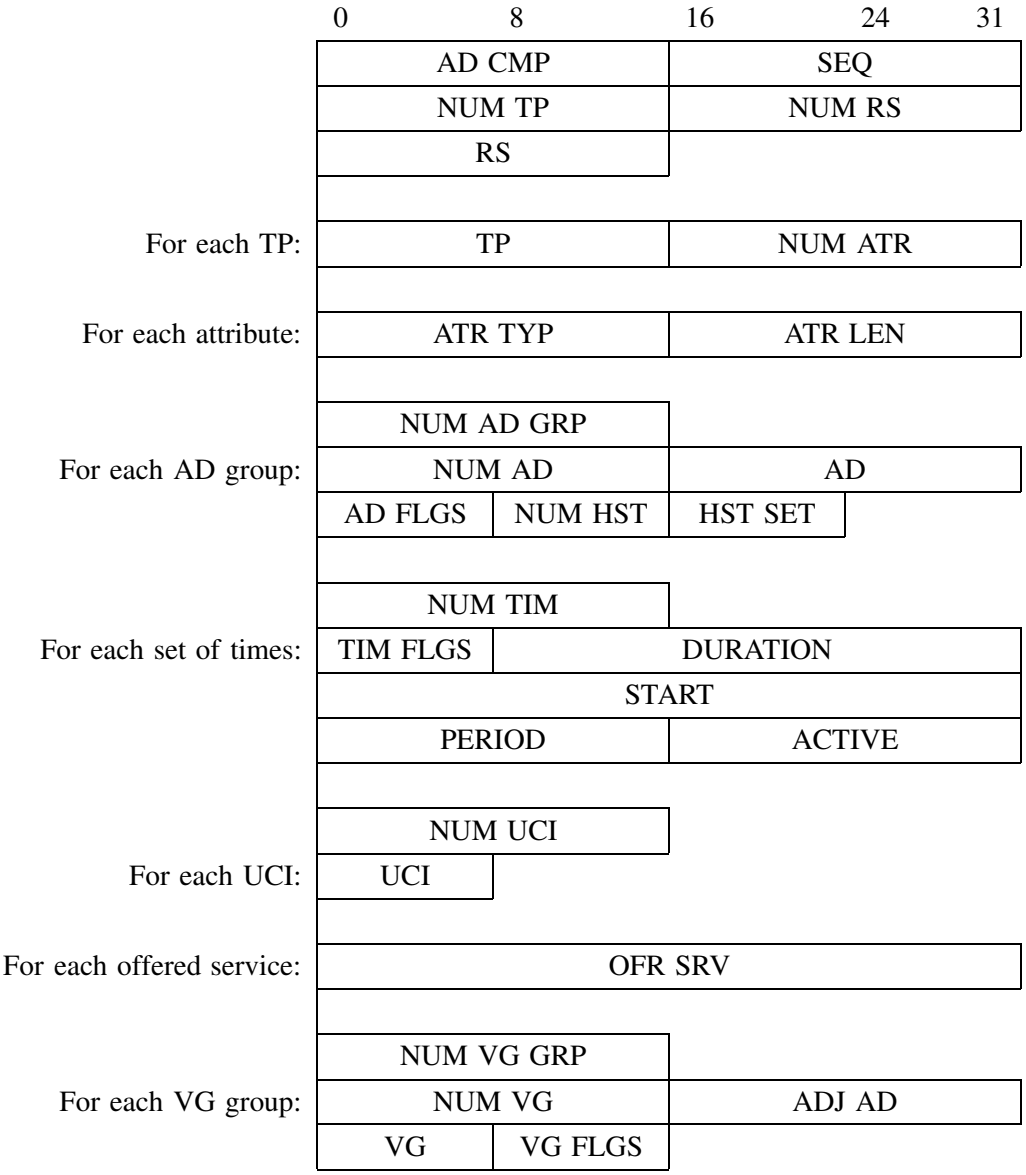
The route server treats routing information message expiration differently, depending on the type of routing information message. When a **configuration** message expires, the route server requests, from another route server, the most recent **configuration** message issued by the given domain. When a **dynamic** message expires, the route server does not initially attempt to obtain more recent routing information. Instead, if route generation is necessary, the route server uses the routing information contained in the corresponding **configuration** message for the given domain. Only if there is a path setup failure (see section 7.4) involving the given domain does the route server request, from another route server, the most recent **dynamic** message issued by the given domain.

4.3 Routing Information Message Formats

The flooding protocol number is equal to 1. We describe the contents of each type of routing information message below.

4.3.1 Configuration

The **configuration** message type is equal to 0.



AD CMP (16 bits) Numeric identifier for the domain component containing the AD representative policy gateway.

SEQ (16 bits) Routing information message sequence number.

NUM TP (16 bits) Number of transit policy specifications contained in the routing information message.

NUM RS (16 bits) Number of route servers advertised to serve clients outside of the domain.

RS (16 bits) Numeric identifier for a route server.

TP (16 bits) Numeric identifier for a transit policy specification.

NUM ATR (16 bits) Number of attributes associated with the transit policy.

ATR TYP (16 bits) Numeric identifier for a type of attribute. Valid attributes include the following types:

1. Set of virtual gateway groups (see section 1.4.2) associated with the transit policy (variable); must be included.
2. Set of source/destination domain groups (see section 1.4.2) associated with the transit policy (variable); may be omitted. Absence of this attribute implies that traffic from any source domain to any destination domain is acceptable.
3. Set of time specifications (see section 1.4.2) associated with the transit policy (variable); may be omitted. Absence of this attribute implies that the transit policy always applies.
4. Set of user classes (see section 1.4.2) associated with the transit policy (variable); may be omitted. Absence of this attribute implies that traffic of any user class is acceptable.
5. Average delay in milliseconds (16 bits); may be omitted.
6. Delay variation in milliseconds (16 bits); may be omitted.
7. Average available bandwidth in bits per second (48 bits); may be omitted.
8. Available bandwidth variation in bits per second (48 bits); may be omitted.
9. MTU in bytes (16 bits); may be omitted.
10. Charge per byte in thousandths of a cent (16 bits); may be omitted.
11. Charge per message in thousandths of a cent (16 bits); may be omitted.
12. Charge for session time in thousandths of a cent per second (16 bits); may be omitted.

Absence of any charge attributes implies that the domain provides free transit service.

ATR LEN (16 bits) Length of an attribute in bytes, beginning with the next field.

NUM AD GRP (16 bits) Number of source/destination domain groups associated with the transit policy.

NUM AD (16 bits) Number of domains or domain sets (see section 1.4.2) in a domain group.

AD (16 bits) Numeric identifier for a domain or domain set.

AD FLGS (8 bits) Set of five flags indicating how to interpret the AD field and contained in the right-most bits. Proceeding left to right, the first flag indicates whether the transit policy applies to all domains or to specific domains (1 all, 0 specific), and when set to 1, causes the second and third flags to be ignored. The second flag indicates whether the domain identifier signifies a single domain or a domain set (1 single, 0 set). The third flag indicates whether the transit policy applies to the given domain or domain set (1 applies, 0 does not apply) and is used for representing complements of sets of domains. The fourth flag indicates whether the domain is a source (1 source, 0 not source).

The fifth flag indicates whether the domain is a destination (1 destination, 0 not destination). At least one of the fourth and fifth flags must be set to 1.

NUM HST (8 bits) Number of host sets (see section 1.4.2) associated with a particular domain. The value 0 indicates that all hosts in the given domain are acceptable sources or destinations, as specified by the fourth and fifth AD flags.

HST (8 bits) Numeric identifier for a host set.

NUM TIM (16 bits) Number of time specifications associated with the transit policy. Each time specification is split into a set of continguous identical periods.

TIM FLGS (8 bits) Set of two flags indicating how to combine the time specifications and contained in the right-most bits. Proceeding left to right, the first flag indicates whether the transit policy applies during the periods specified in the time specification (1 applies, 0 does not apply) and is used for representing complements of transit policy applicability periods. The second flag indicates whether the time specification takes precedence over the previous time specifications listed (1 precedence, 0 no precedence). Precedence is equivalent to the boolean OR and AND operators in the following sense. At any given instant, a transit policy either applies or does not apply, according to a given time specification. We can assign a boolean value to the state of transit policy applicability according to a given time specification. If the second flag assumes the value 1 for a given time specification, that indicates the boolean operator OR should be applied to the value of transit policy applicability, according to the given time specification and to all previous time specifications. If the second flag assumes the value 0 for a given time specification, that indicate the boolean operator OR should be applied to the value of transit policy applicability, according to the given time specification and to all previous time specifications.

DURATION (24 bits) Length of time during which the time specification applies, in minutes. A value of 0 indicates the time specification applies forever.

START (32 bits) Time at which the time specification first takes effect, in seconds elapsed since 1 January 1970 0:00 GMT.

PERIOD (16 bits) Length of each period within the time specification, in minutes.

ACTIVE (16 bits) Length of time the transit policy is applicable during each period, in minutes from the beginning of the period.

NUM UCI (16 bits) Number of user classes associated with the transit policy.

UCI (8 bits) Numeric identifier for a user class.

NUM VG GRP (16 bits) Number of virtual gateway groups associated with the transit policy.

NUM VG (16 bits) Number of virtual gateways in a virtual gateway group.

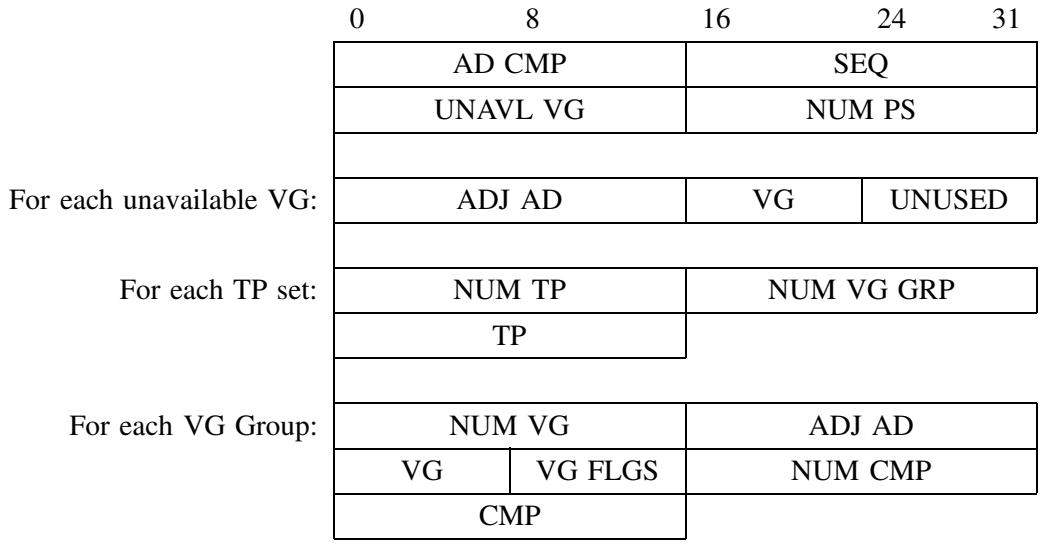
ADJ AD (16 bits) Numeric identifier for the adjacent domain to which a virtual gateway connects.

VG (8 bits) Numeric identifier for a virtual gateway.

VG FLGS (8 bits) Set of two flags indicating how to interpret the VG field and contained in the right-most bits. Proceeding left to right, the first flag indicates whether the virtual gateway is a domain entry point (1 entry, 0 not entry) for the transit policy. The second flag indicates whether the virtual gateway is a domain exit point (1 exit, 0 not exit) for the transit policy. At least one of the first and second flags must be set to 1.

4.3.2 Dynamic

The **dynamic** message type is equal to 1.



AD CMP (16 bits) Numeric identifier for the domain component containing the AD representative policy gateway.

SEQ (16 bits) Routing information message sequence number.

UNAVL VG (16 bits) Number of virtual gateways in the domain component that are currently unavailable via any intra-domain routes.

NUM PS (16 bits) Number of sets of transit policies listed. A single set of virtual gateway groups applies to all transit policies in a given set. Hence, transit policy sets provide a mechanism for reducing the size of **dynamic** messages.

ADJ AD (16 bits) Numeric identifier for the adjacent domain to which a virtual gateway connects.

VG (8 bits) Numeric identifier for a virtual gateway.

UNUSED (8 bits) Not currently used; must be set equal to 0.

NUM TP (16 bits) Number of transit policies in a set.

NUM VG GRP (16 bits) Number of virtual gateway groups currently associated with the transit policy set.

TP (16 bits) Numeric identifier for a transit policy.

NUM VG (16 bits) Number of virtual gateways in a virtual gateway group.

VG FLGS (8 bits) Set of two flags indicating how to interpret the VG field and contained in the right-most bits. Proceeding left to right, the first flag indicates whether the virtual gateway is a domain entry point (1 entry, 0 not entry) for the transit policies. The second flag indicates whether the virtual gateway is a domain exit point (1 exit, 0 not exit) for the transit policies. At least one of the first and second flags must be set to 1.

NUM CMP (16 bits) Number of adjacent domain components reachable via direct connections through the virtual gateway.

CMP (16 bits) Numeric identifier for a reachable adjacent domain component.

4.3.3 Negative Acknowledgements

When a policy gateway or route server receives an unacceptable IDPR routing information message that passes the CMTP validation checks, it includes, in its CMTP **ack**, an appropriate negative acknowledgement. This information is placed in the INFORM field of the CMTP **ack** (described in section 2.4); the numeric identifier for each type of routing information message negative acknowledgement is contained in the left-most 8 bits of the INFORM field. Negative acknowledgements associated with routing information messages include the following types:

1. Unrecognized IDPR routing information message type. Numeric identifier for the unrecognized message type (8 bits).

2. Out-of-date IDPR routing information message. This is a signal to the sender that it may not have the most recent routing information for the given domain.

5 Route Server Query Protocol

Each route server is responsible for maintaining both the routing information and route databases and for responding to database information requests from policy gateways and other route servers. These requests and their responses are the messages exchanged via the route server query protocol (RSQP).

Policy gateways and route servers normally invoke RSQP to replace absent, outdated, or corrupted information in their own routing information or route databases. In section 4, we discussed some of the situations in which RSQP must be invoked; in sections 6 and 7, we discuss other such situations.

5.1 Message Exchange

Policy gateways and route servers use CMTP for reliable transport of route server requests and responses. RSQP must communicate to CMTP the maximum number of transmissions per request/response message, `rsqp_ret`, and the interval between request/response message retransmissions, `rsqp_int` microseconds. A route server request/response message is *acceptable* if:

1. It passes the CMTP validation checks.
2. Its timestamp is less than `rsqp_old` (300) seconds behind the recipient's internal clock time.

With RSQP, a requesting entity expects to receive an acknowledgement from the queried route server indicating whether the route server can accommodate the request. The route server may fail to fill a given request, either because its corresponding database contains no entry or only a partial entry for the requested information, or because it is governed by special message distribution rules, imposed by the domain administrator, that preclude it from releasing the requested information. For all requests that it cannot fill, the route server responds with a negative acknowledgement message carried in a CMTP acknowledgement, indicating the set of unfulfilled requests (see section 5.3.4).

If the requesting entity either receives a negative acknowledgement or does not receive any acknowledgement after `rsqp_ret` attempts directed at the same route server, it queries a different route server, as long as the number of attempted requests to different route servers does not exceed `rsqp_try` (3). Specifically, the requesting entity proceeds in round-robin order through its list of addressable route servers. However, if the requesting entity is unsuccessful after `rsqp_try` attempts, it abandons the request altogether and logs the event for network management.

A policy gateway or a route server can request information from any route server that it can address. Addresses for local route servers within a domain are part of the configuration for each IDPR entity within

a domain; addresses for remote route servers in other domains are obtained through flooded **configuration** messages, as described in section 4.2.1. However, requesting entities always query local route servers before remote route servers, in order to contain the costs associated with the query and response. If the requesting entity and the queried route server are in the same domain, they can communicate over intra-domain routes, whereas if the requesting entity and the queried route server are in different domains, they must obtain a policy route and establish a path before they can communicate, as described in section 5.2 below.

5.1.1 Routing Information

Policy gateways and route servers request routing information from route servers, in order to update their routing information databases. To obtain routing information from a route server, the requesting entity issues a **routing information request** message containing the type of routing information requested – **configuration** messages, **dynamic** messages, or both – and the set of domains from which the routing information is requested.

Upon receiving a **routing information request** message, a route server first assesses message acceptability before proceeding to act on the contents. If the **routing information request** message is deemed acceptable, the route server determines how much the request it can fulfill and then instructs CMTP to generate an acknowledgement, indicating its ability to fulfill the request. The route server proceeds to fulfill as much of the request as possible by reconstructing individual routing information messages, one per requested message type and domain, from its routing information database. We note that only a regenerated routing information message whose entire contents match that of the original routing information message can pass the CMTP integrity/authentication checks.

5.1.2 Routes

Path agents request routes from route servers when they require policy routes for path setup. To obtain routes from a route server, the requesting path agent issues a **route request** message containing the destination domain and applicable service requirements, the maximum number of routes requested, a directive indicating whether to generate the routes or retrieve them from the route database, and a directive indicating whether to refresh the routing information database with the most recent **configuration** or **dynamic** message from a given domain, before generating the routes. To refresh its routing information database, a route server must obtain routing information from another route server. The path agent usually issues routing information database refresh directives in response to a failed path setup. We discuss the application of these directives in more detail in section 7.4.

Upon receiving a **route request** message, a route server first assesses message acceptability before proceeding to act on the contents. If the **route request** message is deemed acceptable, the route server determines whether it can fulfill the request and then instructs CMTP to generate an acknowledgement, indicating its ability to fulfill the request. The route server proceeds to fulfill the request with policy routes, either retrieved from its route database or generated from its routing information database if necessary, returned in a **route response** message.

5.2 Remote Route Server Communication

Communication with a remote route server requires a policy route and accompanying path setup (see section 7) between the requesting and queried entities, as these entities reside in different domains. After generating a request message, the requesting entity hands to CMTP its request message along with the remote route server's entity and domain identifiers. CMTP encloses the request in a **datagram** and hands the **datagram** and remote route server information to the path agent. Using the remote route server information, the path agent obtains, and if necessary sets up, a path to the remote route server. Once the path to the remote route server has been successfully established, the path agent encapsulates the **datagram** within an IDPR data message and forwards the data message along the designated path.

When the path agent in the remote route server receives the IDPR data message, it extracts the **datagram** and hands it to CMTP. In addition, the path agent, using the requesting entity and domain identifiers contained in the path identifier, obtains, and if necessary sets up, a path back to the requesting entity.

If the **datagram** fails any of the CMTP validation checks, CMTP returns a **nak** to the requesting entity. If the **datagram** passes all of the CMTP validation checks, the remote route server assesses the acceptability of the request message. Provided the request message is acceptable, the remote route server determines whether it can fulfill the request and directs CMTP to return an **ack** to the requesting entity. The **ack** may contain a negative acknowledgement if the entire request cannot be fulfilled.

The remote route server generates responses for all requests that it can fulfill and returns the responses to the requesting entity. Specifically, the remote route server hands to CMTP its response and the requesting entity information. CMTP in turn encloses the response in a **datagram**.

When returning an **ack**, a **nak**, or a response to the requesting entity, the remote route server hands the corresponding CMTP message and requesting entity information to the path agent. Using the requesting entity information, the path agent retrieves the path to the requesting entity, encapsulates the CMTP message within an IDPR data message, and forwards the data message along the designated path.

The requesting entity, upon receiving an **ack**, **nak**, or response to its request, performs the CMTP

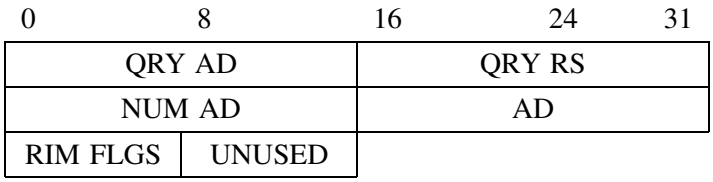
validation checks for that message. In the case of a response messsage, the requesting entity assesses message acceptability before incorporating the contents into the appropriate database.

5.3 Route Server Message Formats

The route server query protocol number is equal to 2. We describe the contents of each type of RSQP message below.

5.3.1 Routing Information Request

The **routing information request** message type is equal to 0.



QRY AD (16 bits) Numeric identifier for the domain containing the queried route server.

QRY RS (16 bits) Numeric identifier for the queried route server.

NUM AD (16 bits) Number of domains about which information is requested. The value 0 indicates a request for routing information from all domains.

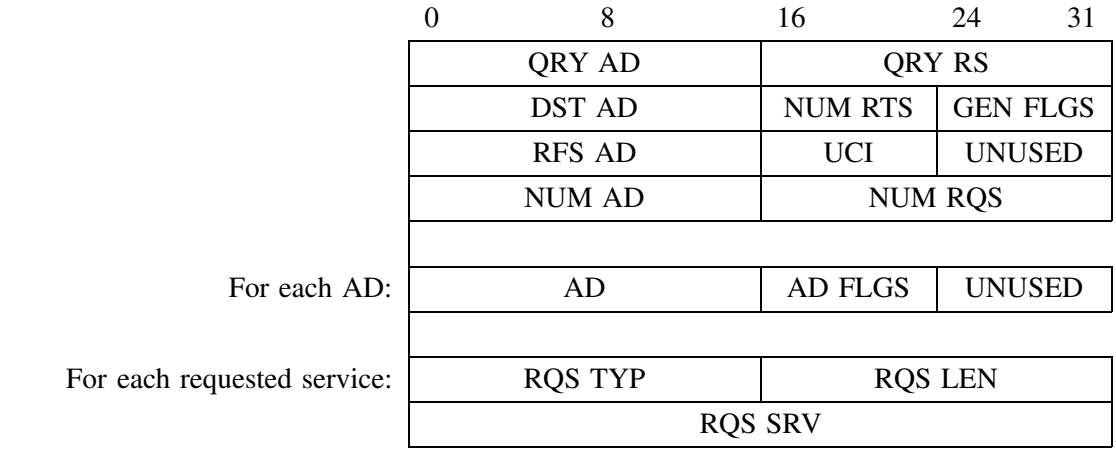
AD (16 bits) Numeric identifier for a domain. This field is absent when NUM AD equals 0.

RIM FLGS (8 bits) Set of two flags indicating the type of routing information messages requested and contained in the right-most bits. Proceeding left to right, the first flag indicates whether the request is for a **configuration** message (1 configuration, 0 no configuration). The second flag indicates whether the request is for a **dynamic** message (1 dynamic, 0 no dynamic). At least one of the first and second flags must be set to 1.

UNUSED (8 bits) Not currently used; must be set equal to 0.

5.3.2 Route Request

The **route request** message type is equal to 1.



QRY AD (16 bits) Numeric identifier for the domain containing the queried route server.

QRY RS (16 bits) Numeric identifier for the queried route server.

DST AD (16 bits) Numeric identifier for the route's destination domain.

NUM RTS (8 bits) Number of policy routes requested.

GEN FLGS (8 bits) Set of three flags indicating how to obtain the requested routes and contained in the right-most bits. Proceeding left to right, the first flag indicates whether the route server should retrieve existing routes from its route database or generate new routes (1 retrieve, 0 generate). The second flag indicates whether the route server should refresh its routing information database before generating the requested routes (1 refresh, 0 no refresh) and when set to 1, causes the third flag and the RFS AD field to become significant. The third flag indicates whether the routing information database refresh should include **configuration** messages or **dynamic** messages (1 configuration, 0 dynamic).

RFS AD (16 bits) Numeric identifier for the domain for which routing information should be refreshed. This field is meaningful only if the second flag in the GEN FLGS field is set to 1.

UCI (8 bits) Numeric identifier of the source user class. The value 0 indicates that there is no particular source user class.

UNUSED (8 bits) Not currently used; must be set equal to 0.

NUM AD (16 bits) Number of transit domains that are to be favored, avoided, or excluded during route selection.

NUM RQS (16 bits) Number of requested services. The value 0 indicates that there is no special service requested.

AD (16 bits) Numeric identifier for the transit domain to be favored, avoided, or excluded.

AD FLGS (8 bits) Three flags indicating how to interpret the AD field and contained in the right-most bits. Proceeding left to right, the first flag indicates whether the domain should be favored (1 favored, 0 not favored). The second flag indicates whether the domain should be avoided (1 avoided, 0 not avoided). The third flag indicates whether the domain should be excluded (1 excluded, 0 not excluded). No more than one of the first, second, and third flags must set to 1.

RQS TYP (16 bits) Numeric identifier for a type of requested service. Valid requested services include the following types:

1. Delay in milliseconds (16 bits); may be omitted.
2. Minimum delay route; may be omitted.
3. Delay variation in milliseconds (16 bits); may be omitted.
4. Minimum delay variation route; may be omitted.
5. Bandwidth in bits per second (48 bits); may be omitted.
6. Maximum bandwidth route; may be omitted.
7. Session monetary cost in cents (32 bits); may be omitted.
8. Minimum session monetary cost route; may be omitted.
9. Path lifetime in minutes (16 bits); may be omitted but must be present if types 7 or 8 are present.
10. Path lifetime in messages (16 bits); may be omitted but must be present if types 7 or 8 are present.
11. Path lifetime in bytes (48 bits); may be omitted but must be present if types 7 or 8 are present.
12. MD4 data message authentication; relevant only to **setup** messages (see section 7.4).
13. MD5 data message authentication; relevant only to **setup** messages.
14. Billing address (variable); relevant only to **setup** messages.
15. Charge number (variable); relevant only to **setup** messages.

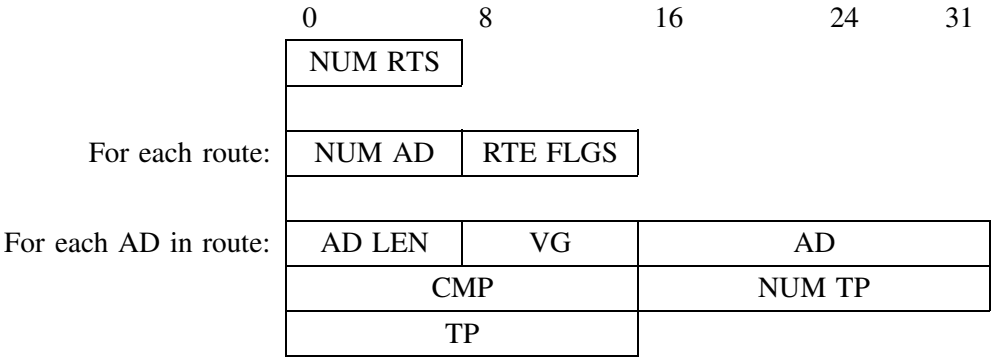
Route servers use path lifetime information together with domain charging method to compute expected session monetary cost over a given domain.

RQS LEN (16 bits) Length of the requested service in bytes, beginning with the next field.

RQS SRV (variable) Description of the requested service.

5.3.3 Route Response

The **route response** message type is equal to 2.



NUM RTS (16 bits) Maximum number of policy routes requested.

RTE FLGS (8 bits) Set of two flags indicating the directions in which a route can be used and contained in the right-most bits. Refer to sections 6.1.1, 7.2, and 7.4 for detailed discussions of path directionality. Proceeding left to right, the first flag indicates whether the route can be used from source to destination (1 from source, 0 not from source). The second flag indicates whether the route can be used from destination to source (1 from destination, 0 not from destination). At least one of the first and second flags must be set to 1, if NUM RTS is greater than 0.

NUM AD (8 bits) Number of domains in the policy route, not including the source domain.

AD LEN (8 bits) Length of the information associated with a particular domain in bytes, beginning with the next field.

VG (8 bits) Numeric identifier for an entry virtual gateway.

AD (16 bits) Numeric identifier for an adjacent administrative domain.

CMP (16 bits) Numeric identifier for an adjacent domain component. Used by policy gateways to select a route across a virtual gateway connecting to a partitioned domain.

NUM TP (16 bits) Number of transit policies that apply to the section of the route traversing the domain.

TP (16 bits) Numeric identifier for a transit policy.

5.3.4 Negative Acknowledgements

When a policy gateway receives an unacceptable RSQP message that passes the CMTP validation checks, it includes, in its CMTP **ack**, an appropriate negative acknowledgement. This information is placed in

the INFORM field of the CMTP **ack** (described in section 2.4); the numeric identifier for each type of RSQP negative acknowledgement is contained in the left-most 8 bits of the INFORM field. Negative acknowledgements associated with RSQP include the following types:

1. Unrecognized RSQP message type. Numeric identifier for the unrecognized message type (8 bits).
2. Out-of-date RSQP message.
3. Unable to fill requests for routing information from the following domains. Number of domains for which requests cannot be filled (16 bits); a value of 0 indicates that the route server cannot fill any of the requests. Numeric identifier for each domain for which a request cannot be filled (16 bits).
4. Unable to fill requests for routes to the following destination domain. Numeric identifier for the destination domain (16 bits).

6 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 optimal requested service specified in the **route request** message. The route server should resolve ties between otherwise equivalent routes by evaluating these routes according to the other optimal requested services contained in the **route request** message, 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 specified in the **route request** message but is not necessarily optimal according to any other optimal requested service specified in the **route request** message.

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.

8. A route server should generate at least one route to each component of a partitioned destination domain, because it does not know in which domain component the destination host resides. Hence, a route server can maximize the chances of providing a feasible route to a destination within a partitioned domain.

6.1 Searching

We do not require that all route servers execute the identical procedures for generating routes. 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.

We offer an IDPR route generation procedure as a model. This procedure can be used either to generate a single policy route from the source domain to a specified destination domain or to generate a set of policy routes from the source domain to all destination domains. With slight modification, this procedure can be made to search in either BF or SPF order.

For high-bandwidth traffic flows, BF search is the recommended search technique, because it produces minimum-hop routes. For low-bandwidth traffic flows, the route server may use either BF search or SPF search. We recommend using SPF search only for optimal requested services and never in response to a request for a maximum bandwidth route.

6.1.1 Implementation

Data Structures: The routing information database contains the graph of the Internet, in which virtual gateways are the nodes and intra-domain routes between virtual gateways are the links. During route generation, each route is represented as a sequence of domains and relevant transit policies, together with a list of route characteristics, stored in a temporary array and indexed by destination domain.

1. Execute the **Policy Consistency** routine, first with the source domain as the given domain and second with the destination domain as the given domain. If any policy inconsistency precludes the requested traffic flow, go to **Exit**.
2. For each domain, initialize a null route, set the route bandwidth to 0, and set the following route characteristic values to ∞: route delay, route delay variation, session monetary cost, and route length in hops.
3. With each operational virtual gateway in the source domain, associate the route characteristics of the source domain.
4. Initialize a next-node data structure which will contain, for each route in progress, the virtual gateway at the current endpoint of the route together with the associated route characteristics. The next-node data structure determines the order in which routes get expanded.

   BF: A fifo queue.

   SPF: A heap, ordered according to the first optimal requested service listed in the **route request** message.

Remove Next Node: These steps are performed for each virtual gateway in the next-node data structure.

1. If there are no more virtual gateways in the next-node data structure, go to **Exit**.
2. Extract a virtual gateway and its associated route characteristics from the next-node data structure, obtain the adjacent domain, and:

   SPF: Remake the heap.
3. If there is a specific destination domain and if for the primary optimal service:

   BF: Route length in hops.

   SPF: First optimal requested service listed in the **route request** message.

   the extracted virtual gateway's associated route characteristic is no better than that of the destination domain, go to **Remove Next Node**.
4. Execute the **Policy Consistency** routine with the adjacent domain as the given domain. If any policy inconsistency precludes the requested traffic flow, go to **Remove Next Node**.

5. Check that the source domain's transit policies do not preclude traffic generated by the source host with the specified user class and requested services, from flowing to the adjacent domain as destination. This check is necessary because the route server caches all feasible routes, to intermediate domains, generated during the computation of the requested route. If there are no policy inconsistencies, associate the route and its characteristics with the adjacent domain.
6. If there is a specific destination domain and if the adjacent domain is that destination destination domain, go to **Remove Next Node**.
7. Record the set of all exit virtual gateways in the adjacent domain for which the adjacent domain's transit policies permit the requested traffic flow and which are currently reachable from the entry virtual gateway.

Next Node: These steps are performed for all exit virtual gateways in the above set.

1. If there are no exit virtual gateways in the set, go to **Remove Next Node**.
2. Compute the characteristics for the route to the exit virtual gateway, and check that all of the route characteristic values are within the requested service limits. If any of the route characteristic values are outside of these limits, go to **Next Node**.
3. Compare these route characteristic values with those already associated with the exit virtual gateway (there may be none, if this is the first time the exit virtual gateway has been visited in the search), according to the primary optimal service.
4. Select the route with the optimal value of the primary optimal service, resolve ties by considering optimality according to the other optimal requested services in rank order, and associate the selected route and its characteristics with the exit virtual gateway.
5. Add the virtual gateway to the next-node structure:

   BF: Add to the end of the fifo queue.

   SPF: Add to the heap.

   and go to **Next Node**.

Exit: Return a response to the route request, consisting of either a set of candidate policy routes or an indication that the route request cannot be fulfilled.

Policy Consistency: Check policy consistency for the given domain.

1. Check that the given domain is not specified as an excluded domain in the route request.
2. Check that the given domain's transit policies do not preclude traffic generated by the source host with the specified user class and requested services, from flowing to the destination host and domain.

A path agent may wish to set up a bidirectional path using a route supplied by a route server. (Refer to sections 7.2 and 7.4 for detailed discussions of path directionality.) However, a route server can only guarantee that the routes it supplies are feasible if used in the direction from source to destination. The reason is that the route server, which resides in the source domain, does not have access to, and thus cannot account for, the source policies of the destination domain. Nevertheless, the route server can provide the path agent with an indication of its assessment of route feasibility in the direction from destination to source.

A necessary but insufficient condition for a route to be feasible in the direction from destination to source is as follows. The route must be consistent, in the direction from destination to source, with the transit policies of the domains that compose the route. The transit policy consistency checks performed by the route server during route generation account for the direction from source to destination but not for the direction from destination to source. Only after a route server generates a feasible route from source to destination does it perform the transit policy consistency checks for the route in the direction from destination to source. Following these checks, the route server includes in its **route response** message to the path agent an indication of its assessment of route feasibility in each direction.

6.2 Route Database

A policy route, as originally specified by a route server, is an ordered list of virtual gateways, domains, and transit policies: $VG_1 - AD_1 - TP_1 - \cdots VG_n - AD_n - TP_n$, where VG_i is the virtual gateway that serves as exit from AD_{i-1} and entry to AD_i, and TP_i is the set of transit policies associated with AD_i and relevant to the particular route. Route servers and paths agents store policy routes in route databases maintained as caches whose entries must be periodically flushed to avoid retention of stale policy routes. A route server's route database is the set of all routes it has generated on behalf of its domain as source. A path agent's route database is the set of all routes it has requested and received from route servers on behalf of hosts within its domain.

When attempting to locate a feasible route for a traffic flow, a path agent first consults its own route database before querying a route server, provided that the source policy associated with the source host does not include any requested qualities of service. In this case, if its route database contains one or more routes between the given source and destination domains, the path agent checks each such route against the set of excluded domains listed in the source policy. The path agent either selects the first route encountered that does not include the excluded domains, or, if no such route exists in its route database, requests a route from a route server.

The path agent must query a route server for routes when the source policy includes requested qualities of service. The reason is that the path agent retains no transit policy information, and in particular, no

offered service information about other domains. Hence, the path agent cannot determine whether an entry in its route database satisfies the requested services.

When responding to a path agent's request for a policy route, a route server first consults its route database, unless the **route request** message contains an explicit directive to generate a new route. If its route database contains one or more routes between the given source and destination domains, the route server checks each such route against the services requested by the path agent and the services offered by the domains composing the route. To obtain the offered services information, the route server consults its routing information database. The route server either selects the first route encountered that is consistent with both the requested and offered services, or, if no such route exists in its route database, attempts to generate a new route.

6.2.1 Cache Maintenance

Each route stored in a route database has a finite cache lifetime equal to `rdb_rs` minutes for a route server and `rdb_pa` minutes for a path agent. Route servers and path agents reclaim cache space by flushing expired entries. Moreover, paths agents reclaim cache space for routes whose paths have failed to be successfully set up or have been torn down (see section 7.4).

Nevertheless, cache space may become scarce, even with reclamation of entries. If the cache fills, the route server or path agent logs the event for network management. To obtain a cache entry when the cache is full, the route server or path agent deletes from the cache the oldest entry.

7 Path Control Protocol and Data Message Forwarding Procedure

Two entities in different domains can exchange IDPR data messages, only if there exists an IDPR path set up between the two domains. Path setup requires cooperation among path agents and intermediate policy gateways. Path agents locate policy routes, initiate the path control protocol (PCP), and manage existing paths between administrative domains. Intermediate policy gateways verify that a given policy route is consistent with their domains' transit policies, establish the forwarding information, and forward messages along existing paths.

Each policy gateway and each route server contains a path agent. The path agent that initiates path setup in the source domain is the *originator*, and the path agent that handles the originator's path setup message in the destination domain is the *target*. Every path has two possible directions of traffic flow: from originator to target and from target to originator. Path control messages are free to travel in either direction, but data messages may be restricted to only one direction.

Once a path for a policy route is set up, its physical realization is a set of consecutive policy gateways, with policy gateways or route servers forming the endpoints. Two successive entities in this set belong to either the same domain or the same virtual gateway. A policy gateway or route server may, at any time, recover the resources dedicated to a path that goes through it by tearing down that path. For example, a policy gateway may decide to tear down a path that has not been used for some period of time.

PCP may build multiple paths between source and destination domains, but it is not responsible for managing such paths as a group or for eliminating redundant paths.

7.1 An Example of Path Setup

We illustrate how path setup works by stepping through an example. 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 only. 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 forwading information. Eventually, the message will arrive at a policy gateway in $AD\ X$, as policy gateways are the only egress points to other administrative domains, in domains that support IDPR.

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, from the information contained in the message header. In the future, for IP messages, the relevant header information may also include special service-specific IP options or even information from higher layer protocols.

Forwarding database entries exist for all of the following:

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

2. Inter-domain traffic flows that do not require IDPR policy routes. Non-IDPR 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 recommend a radix search to locate such an entry. When the search terminates, it produces either an entry, or, in the case of a new IDPR traffic flow, a directive to generate an entry. If the search terminates in an existing forwarding information database entry, the path agent forwards the message according to that entry.

Suppose that the search terminates indicating that the traffic flow from H_X to H_Y requires an IDPR policy route and that no entry in the forwarding information database yet exists for that 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 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 usually consults its local cache before contacting a route server, as described in section 6.2.

If no suitable cache entry exists, the path agent queries the route server, providing it with the source and destination domains together with source policy information carried in the host message and 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 $AD\ Y$, consistent with the requested services for H_X.

The route server always returns a response to the path agent, regardless of whether it is successful in locating a suitable policy route. 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, IDPR 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, IDPR does not specify how the path agent should use multiple routes.

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. IDPR 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, IDPR does not preclude a path agent from setting up distinct paths along the same policy route to preserve the distinction between the host traffic flows.

The path agent associates an identifier with the path, which is included in each message that travels down the path and is 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 entry in the forwarding information database 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.

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, PCP does not preclude a path agent from forwarding messages

along a path prior to confirmation of successful path establishment. Paths remain in place until they are torn down because of failure, expiration, or when resources are scarce, preemption in favor of other paths.

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 an independent path from $AD\ Y$ to $AD\ X$.

7.2 Path Identifiers

Each path has an associated path identifier, unique throughout the Internet. Every IDPR data message travelling along that path includes the path identifier, used for message forwarding. The path identifier is the concatenation of three items: the identifier of the originator's domain; the identifier of the originator's policy gateway or route server; and a 32-bit local path identifier specified by the originator. The path identifier and the CMTP transaction identifier have analogous syntax and play analogous roles in their respective protocols.

When issuing a new path identifier, the originator always assigns a local path identifier that is different from that of any other active or recently torn-down path originally set up by that path agent. This helps to distinguish new paths from replays. Hence, the originator must keep a record of each extinct path for long enough that all policy gateways on the path will have eliminated any reference to it from their memories. The right-most 30 bits of the local identifier are the same for each path direction, as they are assigned by the originator. The left-most 2 bits of the local identifier indicate the path direction.

At path setup time, the originator specifies which of the path directions to enable contingent upon the information received from the route server in the **route response** message. By enable, we mean that each path agent and each intermediate policy gateway establishes an association between the path identifier and the previous and next policy gateways on the path, which it uses for forwarding data messages along that path. IDPR data messages may travel in the enabled path directions only, but path control messages are always free to travel in either path direction. The originator may enable neither path direction, if the entire data transaction can be carried in the path setup message itself. In this case, the path agents and the intermediate policy gateways do not establish forwarding associations for the path, but they do verify consistency of the policy information contained in the path setup message, with their own transit policies, before forwarding the setup message on to the next policy gateway.

The path direction portion of the local path identifier has different interpretations, depending upon message type. In an IDPR path setup message, the path direction indicates the directions in which the path should be enabled: the value 01 denotes originator to target; the value 10 denotes target to originator; the value 11 denotes both directions; and the value 00 denotes neither direction. Each policy gateway along the path interprets the path direction in the setup message and sets up the forwarding information as directed. In an IDPR data message, the path direction indicates the current direction of traffic flow: either 01 for originator to target or 10 for target to originator. Thus, if for example, an originator sets up a path enabling only the direction from target to originator, the target sends data messages containing the path identifier selected by the originator together with the path direction set equal to 10.

Instead of using path identifiers that are unique throughout the Internet, we could have used path identifiers that are unique only between a pair of consecutive policy gateways and that change from one policy gateway pair to the next. The advantage of locally unique path identifiers is that they can be much shorter than global identifiers and hence consume less bandwidth on links. However, the disadvantage is that the path identifier carried in each IDPR data message must be modified at each policy gateway, and hence if the integrity/authentication information covers the path identifier, it must be recomputed at each policy gateway. For security reasons, we have chosen to include the path identifier in the set of information covered by the integrity/authentication value, and moreover, we advocate public-key based signatures for authentication. Thus, it is not possible for intermediate policy gateways to modify the path identifier and then recompute the correct integrity/authentication value. Therefore, we have decided in favor of path identifiers that do not change from hop to hop and hence must be globally unique. To speed forwarding of IDPR data messages with long path identifiers, policy gateways hash the path identifiers in order to index IDPR forwarding information.

7.3 Path Control Messages

Messages exchanged by the path control protocol are classified into *requests*: **setup**, **teardown**, **repair**; and *responses*: **accept**, **refuse**, **error**. These messages have significance for intermediate policy gateways as well as for path agents.

setup Establishes a path by linking together pairs of policy gateways. The **setup** message is generated by the originator and propagates to the target. In response to a **setup** message, the originator expects to receive an **accept**, **refuse**, or **error** message. The **setup** message carries all information necessary to set up the path including path identifier, requested services, transit policy information relating to each domain traversed, and optionally, expedited data.

accept Signals successful path establishment. The **accept** message is generated by the target, in response to a **setup** message, and propagates back to the originator. Reception of an **accept** message by the

originator indicates that the originator can now safely proceed to send data along the path. The **accept** message contains the path identifier and an optional reason for conditional acceptance.

refuse Signals that the path could not be successfully established, either because resources were not available or because there was an inconsistency between the services requested and the services offered. The **refuse** message is generated by the target or by any intermediate policy gateway, in response to a **setup** message, and propagates back to the originator. All recipients of a **refuse** message usually recover the resources dedicated to the given path. The **refuse** message contains the path identifier and the reason for path refusal.

teardown Tears down a path, typically when a non-recoverable failure is detected. The **teardown** message may be generated by any path agent or policy gateway in the path and usually propagates in both path directions. All recipients of a **teardown** message recover the resources dedicated to the given path. The **teardown** message contains the path identifier and the reason for path teardown.

repair Establishes a repaired path by linking together pairs of policy gateways. The **repair** message is generated by a policy gateway after detecting that the next policy gateway on one of its existing paths is unreachable. A policy gateway that generates a **repair** message propagates the message forward at most two policy gateways. In response to a **repair** message, the policy gateway expects to receive an **accept**, **refuse**, **teardown**, or **error** message. The **repair** message carries the original **setup** message.

error Transports information about a path error back to the originator, when a PCP message contains unrecognized information. The **error** message may be generated by the target or by any intermediate policy gateway and propagates back to the originator. Most but not all **error** messages are generated in response to errors encountered during path setup. The **error** message includes the path identifier and an explanation of the error detected.

Policy gateways use CMTP for reliable transport of PCP messages, between path agents and policy gateways and between consecutive policy gateways on a path. PCP must communicate to CMTP the maximum number of transmissions per path control message, `pcp_ret`, and the interval between path contol message retransmissions, `pcp_int` microseconds. All path control messages, except **error** messages, may be transmitted up to `pcp_ret` times; **error** messages are never retransmitted. A path control message is *acceptable* if:

1. It passes the CMTP validation checks.
2. Its timestamp is less than `pcp_old` (300) seconds behind the recipient's internal clock time.
3. It carries a recognized path identifier, provided it is not a **setup** message.

The path control message age limit reduces the likelihood of denial of service attacks based on message replay. An intermediate policy gateway forwards acceptable PCP messages. As we describe in section 7.4 below, **setup** messages must undergo additional tests at each intermediate policy gateway prior to forwarding. Moreover, receipt of an acceptable **accept**, **refuse**, **teardown**, or **error** message at either path agent or at any intermediate policy gateway indirectly cancels any active local CMTP retransmissions of the original **setup** message. When a path agent or intermediate policy gateway receives an unacceptable path control message, it discards the message and logs the event for network management.

7.4 Setting Up and Tearing Down a Path

Path setup begins when the originator generates a **setup** message containing:

1. The path identifier, including path directions to enable.
2. An indication of whether the message includes expedited data.
3. The source user class.
4. The requested services for the path (see section 5.3.2).
5. For each domain on the path, the domain component, applicable transit policies, and entry and exit virtual gateways.

The only mandatory requested services are the maximum path lifetime, `pth_lif`, and the data message integrity/authentication type. If these are not specified in the path setup message, each recipient policy gateway assigns them default values, (60) minutes for `pth_lif` and no authentication for integrity/authentication type. Each path agent and intermediate policy gateway tears down a path when the path lifetime is exceeded. Hence, no single source can indefinitely monopolize policy gateway resources or still functioning parts of partially broken paths.

After generating the **setup** message and establishing the proper local forwarding information, the originator selects the next policy gateway on the path and forwards the **setup** message to the selected policy gateway. The next policy gateway selection procedure described below applies when the originator or when an intermediate policy gateway is making the selection. We have elected to describe the procedure from the perspective of a selecting intermediate policy gateway.

The policy gateway selects the next policy gateway on a path, in round-robin order from its list of policy gateways contained in the next hop virtual gateway. In selecting the next policy gateway, the policy gateway uses information contained in the **setup** message and information provided by VGP and by the intra-domain routing procedure.

If the selecting policy gateway is a domain entry point, the next policy gateway must be:

1. A member of the next virtual gateway listed in the **setup** message.
2. Reachable according to intra-domain routes supporting the transit policies listed in the **setup** message.
3. Able to reach, according to VGP, the next domain component listed in the **setup** message.

If the selecting policy gateway is a domain exit point, the next policy gateway must be:

1. A member of the current virtual gateway listed in the **setup** message (which is also the selecting policy gateway's virtual gateway).
2. Reachable according to VGP.
3. A member of the next domain component listed in the **setup** message.

In addition, the selecting policy gateway may use the requested services listed in the **setup** message to resolve ties between otherwise equivalent next policy gateways in the same domain. In particular, the selecting policy gateway may use any quality of service information supplied by intra-domain routing, to select the next policy gateway whose connecting intra-domain route is optimal according to the requested services.

Once the originator or intermediate policy gateway selects a next policy gateway, it forwards the **setup** message to the selected policy gateway. Each recipient (policy gateway or target) of an acceptable **setup** message performs several checks on the contents of the message, in order to determine whether to establish or reject the path. We describe these checks in detail below from the perspective of a policy gateway as **setup** message recipient.

7.4.1 Validating Path Identifiers

The recipient of a **setup** message first checks the path identifier, to make sure that it does not correspond to that of an already existing or recently extinct path. To detect replays, malicious or otherwise, path agents and policy gateways maintain a record of each path that they establish, for $max\{$`pth_lif`, `pcp_old`$\}$ seconds. If the path identifier and timestamp carried in the **setup** message match a stored path identifier and timestamp, the policy gateway considers the message to be a retransmission and does not forward the message. If the path identifier carried in the **setup** message matches a stored path identifier but the two timestamps do not agree, the policy gateway abandons path setup, logs the event for network management, and returns an **error** message to the originator via the previous policy gateway.

7.4.2 Path Consistency with Configured Transit Policies

Provided the path identifier in the **setup** message appears to be new, the policy gateway proceeds to determine whether the information contained within the **setup** message is consistent with the transit policies configured for its domain. The policy gateway must locate the source and destination domains, the source user class, and its domain-specific information, within the **setup** message, in order to evaluate path consistency. If the policy gateway fails to recognize the source user class (or one or more of the requested services), it logs the event for network management but continues with path setup. If the policy gateway fails to locate its domain within the **setup** message, it abandons path setup, logs the event for network management, and returns an **error** message to the originator via the previous policy gateway. The originator responds by tearing down the path and subsequently removing the route from its cache.

Once the policy gateway locates its domain-specific portion of the **setup** message, it may encounter the following problems with the contents:

1. The domain-specific portion lists a transit policy not configured for the domain.
2. The domain-specific portion lists a virtual gateway not configured for the domain.

In each case, the policy gateway abandons path setup, logs the event for network management, and returns an **error** message to the originator via the previous policy gateway. These types of **error** messages indicate to the originator that the route may have been generated using information from an out-of-date **configuration** message.

The originator responds to the receipt of such an **error** message as follows. First, it tears down the path and removes the route from its cache. Then, it issues to a route server a **route request** message containing a directive to refresh the routing information database with the most recent **configuration** message from the domain that issued the **error** message, before generating a new route.

Once it verifies that its domain-specific information in the **setup** message is recognizable, the policy gateway then checks that the information contained within the **setup** message is consistent with the transit policies configured for its domain. A policy gateway at the entry to a domain checks path consistency in the direction from originator to target, if the enabled path directions include originator to target. A policy gateway at the exit to a domain checks path consistency in the direction from target to originator, if the enabled path directions include target to originator.

When evaluating the consistency of the path with the configured transit policies, the policy gateway may encounter any of the following problems with **setup** message contents:

1. A listed transit policy does not apply between the listed virtual gateways in the given direction.

2. A listed transit policy denies access to traffic between the listed source and destination domains.

3. A listed transit policy denies access to traffic of the listed user class.

4. A listed transit policy denies access to traffic at the current time.

In each case, the policy gateway abandons path setup, logs the event for network management, and returns a **refuse** message to the originator via the previous policy gateway. These types of **refuse** messages indicate to the originator that the route may have been generated using information from an out-of-date **configuration** message. The **refuse** message also serves to teardown the path.

The originator responds to such a **refuse** message, first by removing the route from its cache. Then, it issues to a route server a **route request** message containing a directive to refresh the routing information database with the most recent **configuration** message from the domain that issued the **refuse** message, before generating a new route.

7.4.3 Path Consistency with Virtual Gateway Reachability

Provided the information contained in the **setup** message is consistent with the transit policies configured for its domain, the policy gateway proceeds to determine whether the path is consistent with the reachability of the virtual gateway containing the potential next hop. To determine virtual gateway reachability, the policy gateway uses information provided by VGP and by the intra-domain routing procedure.

When evaluating the consistency of the path with virtual gateway reachability, the policy gateway may encounter any of the following problems:

1. The virtual gateway containing the potential next hop is down.

2. The virtual gateway containing the potential next hop is not reachable via any intra-domain routes supporting the transit policies listed in the **setup** message.

3. The next domain component listed in the **setup** message is not reachable.

Each of these determinations is made from the perspective of a single policy gateway and may not reflect actual reachability. In each case, the policy gateway encountering such a problem returns a **refuse** message to the previous policy gateway which then selects a different next policy gateway as described in section 7.4 above. If the policy gateway receives the same response from all next policy gateways selected, it abandons path setup, logs the event for network management, and returns the **refuse** message to the originator via the previous policy gateway. These types of **refuse** messages indicate to the originator

that the route may have been generated using information from an out-of-date **dynamic** message. The **refuse** message also serves to teardown the path.

The originator first responds to such a **refuse** message by removing the route from its cache. Then, it issues to a route server a **route request** message containing a directive to refresh the routing information database with the most recent **dynamic** message from the domain that issued the **refuse** message, before generating a new route.

7.4.4 Obtaining Resources

Once the policy gateway determines that the **setup** message contents are consistent with the transit policies and virtual gateway reachability of the recipient's domain, it attempts to gain resources for the new path. For this version of IDPR, path resources consist of memory in the local forwarding information database. However, in the future, path resources may also include reserved link bandwidth.

If the policy gateway does not have resources to establish the new path, it uses the following algorithm to determine whether to generate a **refuse** message for the new path or a **teardown** message for an existing path in favor of the new path. There are two cases:

1. No paths have been idle for more than `pcp_idle` (300) seconds. In this case, the policy gateway returns a **refuse** message to the previous policy gateway. This policy gateway then tries to select a different next policy gateway, as described in section 7.4 above, provided the policy gateway that issued the **refuse** message was not the target. If the **refuse** message was issued by the target or if there is no available next policy gateway, the policy gateway returns the **refuse** message to the originator via the previous policy gateway and logs the event for network management. The **refuse** message serves to tear down the path.

2. At least one path has been idle for more than `pcp_idle` seconds. In this case, the policy gateway tears down an older path in order to accommodate the newer path and logs the event for network management. Specifically, the entity tears down the least recently used path of those that have been idle for longer than `pcp_idle` seconds, resolving ties by choosing the oldest such path.

If the policy gateway has sufficient resources to establish the path, it attempts to update its local forwarding information database with information about the path identifier, previous and next policy gateways on the path, and directions in which the path should be enabled for data traffic transport.

7.4.5 Target Response

When an acceptable **setup** message successfully reaches an entry policy gateway in the destination domain, this policy gateway performs the all of the checks described in the above sections. Provided no problems are encountered, the policy gateway's path agent becomes the target, unless there is an explicit target specified in the **setup** message, as with RSQP messages exchanged between remote route servers (see section 5.2). If the policy gateway is not the target, it attempts to forward the **setup** message to the target along an intra-domain route. However, if the target is not reachable via intra-domain routing, the policy gateway abandons path setup, logs the event for network management, and returns a **refuse** message to the originator via the previous policy gateway. The **refuse** message serves to tear down the path.

Once the **setup** message reaches the target, the target determines whether it has sufficient path resources. Provided the target does have sufficient resources to establish the path, it generates an **accept** message. The target then determines whether the destination host is reachable via intra-domain routing and includes this information in the **accept** message, before returning the **accept** message to the originator via the previous policy gateway. Destination host reachability information aids the originator in determining if the path can be used to reach the destination host.

The target may choose to use the reverse path to transport data traffic to the source domain, if the enabled path directions include 10 or 11. However, the target must first verify the consistency of the reverse path with its domain's configured source and transit policies.

7.4.6 Originator Response

The originator expects to receive an **accept**, **refuse**, or **error** message in response to a **setup** message. There are three cases:

1. The originator receives an **accept** message, confirming successful path establishment. To expedite data delivery, the originator may forward data messages along the path prior to receiving an **accept** message, with the understanding that there is no guarantee that the path actually exists.

2. The originator receives a **refuse** message or an **error** message, implying that the path could not be successfully established. In response, the originator attempts to set up a different path to the same destination, as long as the number of selected different paths does not exceed `setup_try` (3). If the originator is unsuccessful after `setup_try` attempts, it abandons path setup and logs the event for network management.

3. The originator fails to receive any response to the **setup** message within `setup_int` microseconds after transmission. In this case, the originator attempts path setup using the same policy route and a new path identifier, as long as the number of path setup attempts using the same route does not exceed `setup_ret` (2). If the originator fails to receive a response to a **setup** message after `setup_ret` attempts, it logs the event for network management and then proceeds as though it received a negative response, namely a **refuse** or an **error**, to the **setup** message. Specifically, it attempts to set up a different path to the same destination, or it abandons path setup altogether, depending on the value of `setup_try`.

7.4.7 Path Life

Once set up, a path does not live forever. A path agent or policy gateway may tear down an existing path, provided any of the following conditions are true:

1. The maximum path lifetime (in minutes, bytes, or messages) has been exceeded. An originator path agent generates a **teardown** message for propagation toward the target. A target path agent generates a **teardown** message for propagation toward the originator. An intermediate policy gateway generates two **teardown** messages, one for propagation toward the originator and one for propagation toward the target. In all cases, the IDPR entity detecting path expiration logs the event for network management.

2. The previous or next policy gateway becomes unreachable, across a virtual gateway or across a domain according to a given transit policy, and the path is not repairable. If the previous policy gateway is unreachable, a policy gateway generates a **teardown** message for propagation to the target. If the next policy gateway is unreachable, a policy gateway generates a **teardown** message for propagation to the originator. In either case, the policy gateway detecting the reachability problem logs the event for network management.

3. All of the policy gateway's path resources are in use, a new path requires resources, and the given existing path is expendable, according to the least recently used criterion discussed in section 7.4.4 above. A target path agent generates a **teardown** message for propagation toward the originator. An intermediate policy gateway generates two **teardown** messages, one for propagation toward the originator and one for propagation toward the target. In either case, the IDPR entity initiating path preemption logs the event for network management.

Path teardown at a path agent or policy gateway, whether initiated by one of the above events or by receipt of a **teardown** message (or a **refuse** message during path setup, as discussed in the previous

sections), causes the path agent or policy gateway to release all resources devoted to both directions of the path.

7.5 Path Failure and Recovery

When a policy gateway fails, it may not be able to save information pertaining to its established paths. Thus, when the policy gateway returns to service, it has no recollection of the paths set up through it and can no longer forward data messages along these paths. We expect that when a policy gateway fails, it will usually be out of service for long enough that the up/down protocol and the intra-domain routing procedure can detect that the particular policy gateway is no longer reachable. In this case, adjacent or neighbor policy gateways that have set up paths through the failed policy gateway and that have detected the failure, attempt local route repair (see section 7.5.2 below), and if unsuccessful, issue **teardown** messages for all affected paths.

7.5.1 Handling Implicit Path Failures

Nevertheless, policy gateways along a path must be able to handle the case in which a policy gateway fails and subsequently returns to service without either the up/down protocol or the intra-domain routing procedure detecting the failure, although we do not expect this event to occur often. If the policy gateway previously contained forwarding information for several established paths, it may now receive many IDPR data messages containing unrecognized path identifiers. This policy gateway must alert the data sources that their paths through the given policy gateway are no longer viable.

Policy gateways that receive IDPR data messages with unrecognized path identifiers take one of the following two actions, depending upon their past failure record:

1. The policy gateway has not failed in the past `pg_up` (24) hour period. In this case, there are at least four possible reasons for the unrecognized path identifier in the data message:
   (a) The data message path identifier has been corrupted in a way that is not detectable by the integrity/authentication value, if one is present.
   (b) The policy gateway has experienced a memory error.
   (c) The policy gateway failed sometime during the life of the path and the source sent no data on the path for a period of `pg_up` hours following the failure. Although paths may persist for more than `pg_up` hours, we expect that they will also be used more frequently than once every `pg_up` hours.

(d) The path was not successfully established, and the originator sent data messages down the path prior to receiving a response to its **setup** message.

In all cases, the policy gateway discards the data message and logs the event for network management.

2. The policy gateway has failed at least once in the past `pg_up` hour period. Thus, the policy gateway assumes that the unrecognized path identifier in the data message can be attributed to its failure. In response to the data message, the policy gateway generates an **error** message containing the unrecognized path identifier. The policy gateway then sends the **error** message back to the entity from which it received the data message, which should be equivalent to the previous policy gateway on the path.

When the previous policy gateway receives the **error** message, it decides whether the message is acceptable. If the policy gateway does not recognize the path identifier contained in the **error** message, it does not find the **error** message acceptable and subsequently discards the message. However, if the policy gateway does find the **error** message acceptable, it then determines whether it has already received an **accept** message for the given path. If the policy gateway has not received an **accept** message for that path, it discards the **error** message and takes no further action.

If the policy gateway has received an **accept** message for that path, it then attempts path repair, as described in section 7.5.2 below. Only if path repair is unsuccessful does the previous policy gateway generate a **teardown** message for the path and return it to the originator. The **teardown** message includes the domain and virtual gateway containing the policy gateway that failed, which aids the originator in selecting a new path that does not include the domain containing the failed policy gateway. This mechanism ensures that path agents quickly discover and recover from disrupted paths, while guarding against unwarranted path teardown.

7.5.2 Local Path Repair

Failure of one of more entities on a given path may render the path 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 the same domain or in the same virtual gateway. We say that a path is *locally repairable* if there exists an alternate route between two policy gateways, separated by at most one policy gateway, on the path. This definition covers path repair in the presence of failed routes between consecutive policy gateways as well as failed policy gateways themselves.

An IDPR entity attempts local repair of an established path, in the direction from originator to target, immediately after detecting that the next policy gateway on the path is no longer reachable. To prevent multiple path repairs in response to the same failure, we have stipulated that path repair can only be initiated in the direction from originator to target. The entity initiating local path repair attempts to find an alternate path to the IDPR entity immediately following the unreachable policy gateway on the path, hence the adjective "local".

Local path repair minimizes the disruption of data traffic flow caused by certain types of failures along an established path. Specifically, local path repair can accommodate an individual failed policy gateway or failed direct connection between two adjacent policy gateways. However, it can only be attempted through virtual gateways containing multiple peer policy gateways. Local path repair is not designed to repair paths traversing failed virtual gateways or domain partitions. Whenever local path repair is impossible, the failing path must be torn down.

7.5.3 Repairing a Path

When an entity detects through an **error** message that the next policy gateway has no knowledge of a given path, it generates a **repair** message and forwards it to the next policy gateway. This **repair** message will reestablish the path through the next policy gateway.

When an entity detects that the next policy gateway on a path is no longer reachable, it takes one of the following actions, depending upon whether the entity is a member of the next policy gateway's virtual gateway. If the entity is not a member of the next policy gateway's virtual gateway, then one of the following two conditions must be true:

1. The next policy gateway has a peer that is reachable via an intra-domain route consistent with the requested services. In this case, the entity generates a **repair** message containing the original **setup** message and forwards it to the next policy gateway's peer.

2. The next policy gateway has no peers that are reachable via intra-domain routes consistent with the requested services. In this case, the entity tears down the path back to the originator.

If the entity is a member of the next policy gateway's virtual gateway, then one of the following four conditions must be true:

1. The next policy gateway has a peer that belongs to the same domain component and is directly-connected to and reachable from the entity. In this case, the entity generates a **repair** message and forwards it to the next policy gateway's peer.

2. The next policy gateway has a peer that belongs to the same domain component, is not directly-connected to the entity, but is directly-connected to and reachable from one of the entity's peers, which in turn is reachable from the entity via an intra-domain route consistent with the requested services. In this case, the entity generates a **repair** message and forwards it to its peer.

3. The next policy gateway has no operational peers within its domain component, but is directly-connected to and reachable from one of the entity's peers, which in turn is reachable from the entity via an intra-domain route consistent with the requested services. In this case, the entity generates a **repair** message and forwards it to its peer.

4. The next policy gateway has no operational peers within its domain component, and the entity has no operational peers which are both reachable via intra-domain routes consistent with the requested services and directly-connected to and reachable from the next policy gateway. In this case, the entity tears down the path back to the originator.

A recipient of a **repair** message takes the following steps, depending upon its relationship to the entity that issued the **repair** message. If the recipient and the issuing entity are in the same domain or in the same virtual gateway, the recipient extracts the **setup** message contained within the **repair** message and treats the message as it would any other **setup** message. Specifically, the recipient checks consistency of the path with its domain's transit policies and virtual gateway reachability. If there are unrecognized portions of the **setup** message, the recipient generates an **error** message, and if there are path inconsistencies, the recipient generates a **refuse** message. In either case, the recipient returns the message to the entity that issued the **repair** message. Otherwise, if the recipient accepts the **repair** message, it updates its local forwarding information database accordingly and forwards the **repair** message to a potential next hop, according to the information contained in the enclosed **setup** message.

If the recipient and the issuing entity are in different domains and in different virtual gateways, the recipient extracts the **setup** message from the **repair** message and determines whether the associated path matches any of its established paths. If the path does not match an established path, the recipient

generates a **refuse** message and returns it to the previous policy gateway. In response to this **refuse** message, the previous policy gateway tries a different next policy gateway.

The path is irreparable if all potential next policy gateways have been exhausted and a path match has yet to be discovered. In this case, the previous policy gateway issues a **teardown** message to return to the originator.

The path is repairable, if a path match is discovered. In this case, the recipient updates the path entry in the local forwarding information database and issues an **accept** message to return to the entity that generated the **repair** message.

An IDPR entity expects to receive an **accept**, **teardown**, **refuse**, or **error** message in response to a **repair** message and reacts to these responses differently. The entity always returns a **teardown** message to the originator via the previous policy gateway. It does not return an **accept** message, but receipt of such a message indicates that the path has been successfully repaired. Upon receipt of a **refuse** or an **error** message or when no response to the **repair** message arrives within `setup_int` microseconds, the entity infers that the path is irreparable and subsequently tears down the path and logs the event for network management.

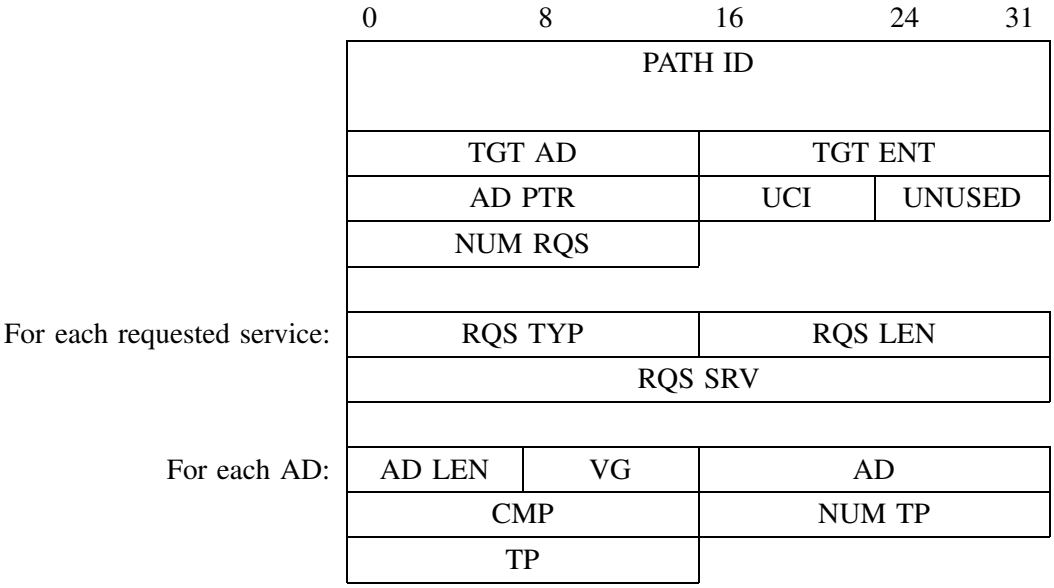
When an entity detects that the previous policy gateway on a path becomes unreachable, it expects to receive a **repair** message within `setup_wait` microseconds. If the entity does not receive a **repair** message for the path within that time, it infers that the path is irreparable and subsequently tears down the path and logs the event for network management.

7.6 Path Control Message Formats

The path control protocol number is equal to 3. We describe the contents of each type of PCP message below.

7.6.1 Setup

The **setup** message type is equal to 0.



PATH ID (64 bits) Path identifier consisting of the numeric identifier of the originator's domain (16 bits), the numeric identifier of the originator policy gateway or route server (16 bits), the path direction (2 bits), and the local path identifier (30 bits).

TGT AD (16 bits) Numeric identifier for the target domain.

TGT ENT (16 bits) Numeric identifier for the target entity. A value of 0 indicates that there is no specific target entity.

AD PTR (16 bits) Byte offset from the beginning of the message indicating the location of the beginning of the domain-specific information, contained in the right-most 15 bits. The left-most bit indicates whether the message includes expedited data (1 expedited data, 0 no expedited data).

UCI (8 bits) Numeric identifier for the source user class. The value 0 indicates that there is no particular source user class.

UNUSED (8 bits) Not currently used; must be set equal to 0.

NUM RQS (16 bits) Number of requested services.

RQS TYP (16 bits) Numeric identifier for a type of requested service. Valid requested services are described in section 5.3.2.

RQS LEN (16 bits) Length of the requested service in bytes, beginning with the next field.

RQS SRV (variable) Description of the requested service.

AD LEN (8 bits) Length of the information associated with a particular domain in bytes, beginning with the next field.

VG (8 bits) Numeric identifier for an entry virtual gateway.

AD (16 bits) Numeric identifier for a domain.

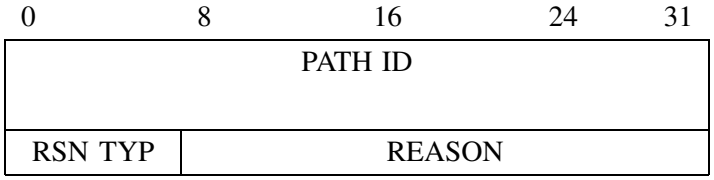
CMP (16 bits) Numeric identifier for a domain component. Used to aid a policy gateway in routing across a virtual gateway connected to a partitioned domain.

NUM TP (16 bits) Number of transit policies that apply to the section of the path traversing the given domain.

TP (16 bits) Numeric identifier for a transit policy.

7.6.2 Accept

The **accept** message type is equal to 1.



PATH ID (64 bits) Path identifier contained in the original **setup** message.

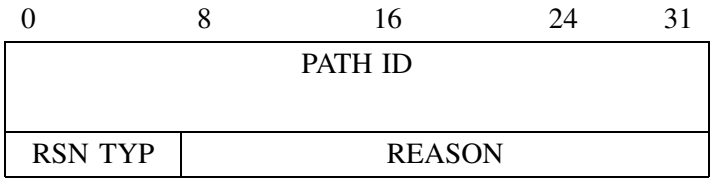
RSN TYP (8 bits) Numeric identifier for the reason for conditional path acceptance.

REASON (variable) Description of the reason for conditional path acceptance. Valid reasons include the following types:

1. Destination host is not currently reachable via intra-domain routing.

7.6.3 Refuse

The **refuse** message type is equal to 2.



PATH ID (64 bits) Path identifier contained in the original **setup** message.

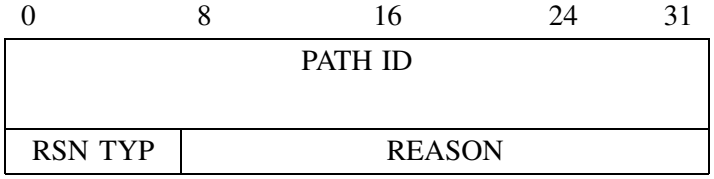
RSN TYP (8 bits) Numeric identifier for the reason for path refusal.

REASON (variable) Description of the reason for path refusal. Valid reasons include the following types:

1. Transit policy does not apply between the virtual gateways in a given direction. Numeric identifier for the transit policy (16 bits).
2. Transit policy denies access to traffic between the source and destination domains. Numeric identifier for the transit policy (16 bits).
3. Transit policy denies access to traffic of the given user class. Numeric identifier for the transit policy (16 bits).
4. Transit policy denies access to traffic at the current time. Numeric identifier for the transit policy (16 bits).
5. Virtual gateway is down. Numeric identifier for the virtual gateway (8 bits) and associated adjacent domain (16 bits).
6. Virtual gateway is not reachable according to the given transit policy. Numeric identifier for the virtual gateway (8 bits), associated adjacent domain (16 bits), and transit policy (16 bits).
7. Domain component is not reachable. Numeric identifier for the domain (16 bits) and the component (16 bits).
8. Insufficient resources to establish the path.
9. Target is not reachable via intra-domain routing.
10. No existing path with the given path identifier, in response to a **repair** message only.

7.6.4 Teardown

The **teardown** message type is equal to 3.



PATH ID (64 bits) Path identifier contained in the original **setup** message.

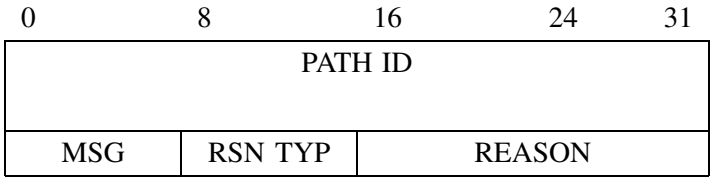
RSN TYP (8 bits) Numeric identifier for the reason for path teardown.

REASON (variable) Description of the reason for path teardown. Valid reasons include the following types:

1. Virtual gateway is down. Numeric identifier for the virtual gateway (8 bits) and associated adjacent domain (16 bits).
2. Virtual gateway is not reachable according to the given transit policy. Numeric identifier for the virtual gateway (8 bits), associated adjacent domain (16 bits), and transit policy (16 bits).
3. Domain component is not reachable. Numeric identifier for the domain (16 bits) and the component (16 bits).
4. Maximum path lifetime exceeded.
5. Preempted path.
6. Unable to repair path.

7.6.5 Error

The **error** message type is equal to 4.



PATH ID (64 bits) Path identifier contained in the path control message in error.

MSG (8 bits) Numeric identifier for the type of path control message in error. This field is ignored for error type 8.

RSN TYP (8 bits) Numeric identifier for the reason for the PCP message error.

REASON (variable) Description of the reason for the PCP message error. Valid reasons include the following types:

1. Path identifier is already currently active.
2. Domain does not appear in the **setup** message.
3. Transit policy not configured for the domain. Numeric identifer for the transit policy (16 bits).
4. Virtual gateway not configured for the domain. Numeric identifier for the virtual gateway (8 bits) and associated adjacent domain (16 bits).
5. Unrecognized path identifier in IDPR data message.

7.6.6 Repair

The **repair** message type is equal to 5. A **repair** message contains the original **setup** message only.

7.6.7 Negative Acknowledgements

When a policy gateway receives an unacceptable PCP message that passes the CMTP validation checks, it includes, in its CMTP **ack**, an appropriate negative acknowledgement. This information is placed in the INFORM field of the CMTP **ack** (described in section 2.4); the numeric identifier for each type of PCP negative acknowledgement is contained in the left-most 8 bits of the INFORM field. Negative acknowledgements associated with PCP include the following types:

1. Unrecognized PCP message type. Numeric identifier for the unrecognized message type (8 bits).
2. Out-of-date PCP message.
3. Unrecognized path identifier (for all PCP messages except **setup**). Numeric identifier for the unrecognized path (64 bits).

References

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

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

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

[4] L. Breslau and D. Estrin. Design of inter-administrative domain routing protocols. *Proceedings of the ACM SIGCOMM '90 Symposium*, September 1990.

[5] M. Lepp and M. Steenstrup. An architecture for inter-domain policy routing. *Internet Draft.* May 1992.

[6] H. Bowns and M. Steenstrup. Inter-domain policy routing configuration and usage. *Internet Draft.* July 1992.

[7] R. Woodburn. Definitions of managed objects for inter-domain policy routing (version 1). *Internet Draft*. March 1992.

[8] J. McQuillan, I. Richer, E. Rosen, and D. Bertsekas. ARPANET routing algorithm improvements: second semiannual technical report. *BBN Report No. 3940.* October 1978.

[9] J. Moy. The OSPF Specification. *RFC 1131.* October 1989.

[10] D. Oran (editor). *Intermediate system to Intermediate system routeing exchange protocol for use in Conjunction with the Protocol for providing the Connectionless-mode Network Service (ISO 8473).* ISO/IEC JTC1/SC6/WG2. October 1989.

[11] D. Estrin and G. Tsudik. Secure control of transit internetwork traffic. *TR-89-15.* Computer Science Department. University of Southern California.

[12] J. Linn. Privacy enhancement for Internet electronic mail: part I – message encipherment and authentication procedures. *RFC 1113.* August 1989.

[13] S. Kent and J. Linn. Privacy enhancement for Internet electronic mail: part II – certificate-based key management. *RFC 1114.* August 1989.

[14] J. Linn. Privacy enhancement for Internet electronic mail: part III – algorithms, modes, and identifiers. *RFC 1115.* August 1989.

[15] R. Rivest. The MD4 Message-Digst Algorithm. *RFC 1320.* April 1992.

[16] R. Rivest. The MD5 Message-Digst Algorithm. *RFC 1321.* April 1992.

Expires 30 November 1992