Inter-Domain Policy Routing Working Group
Internet Draft

M. Steenstrup
BBN Communications
March 1992

IDPR as a Proposed Standard

Executive Summary

The IDPR working group of the IETF is submitting to the IESG, for consideration as a proposed standard, the set of protocols and procedures that compose inter-domain policy routing (IDPR). In accordance with the requirements stipulated by RFC 1264, we present justification for proposed standard status of 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.

Four documents describe IDPR in detail:

1. M. Lepp and M. Steenstrup. An architecture for inter-domain policy routing. *Internet Draft.* March 1992.

2. M. Steenstrup. Inter-domain policy routing protocol specification: version 1. *Internet Draft.* March 1992.

3. H. Bowns and M. Steenstrup. Inter-domain policy routing configuration and usage. *Internet Draft.* March 1992.

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

We request that neither the MIB nor the configuration guide be considered for proposed standard status at this time. Instead, we will submit these documents for proposed standard consideration, after we have gained more experience in using both the MIB and the configuration guide.

1 The Internet Environment

As data communications technologies evolve and user populations grow, the demand for internetworking increases. The Internet currently comprises over 4000 operational networks and over 10,000 registered networks. In fact, for the last several years, the number of constituent networks has approximately doubled annually. Although we do not expect the Internet to sustain this growth rate, we must prepare for the Internet of five to ten years in the future.

Internet connectivity has increased along with the number of component networks. Internetworks proliferate through interconnection of autonomous, heterogeneous networks administered by separate authorities. We use the term *administrative domain* (AD) to refer to any collection of contiguous networks, gateways, links, and hosts governed by a single administrative authority that selects the intra-domain routing procedures and addressing schemes, defines service requirements for locally-generated traffic, and specifies service restrictions for transit traffic.

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

We expect that, over the next five years, the Internet will grow to contain $O(10)$ backbone domains, most providing connectivity between many source and destination domains and offering a wide range of qualities of service, for a fee. Most domains will connect directly or indirectly to at least one Internet backbone domain, in order to communicate with other domains. In addition, some domains may install direct links to their most favored destinations. Domains at the lower levels of the hierarchy will provide some transit service, limited to traffic between selected sources and destinations. However, the majority of Internet domains will be stubs, that is, domains that do not provide any transit service for any other domains but that connect directly to one or more transit domains.

The bulk of Internet traffic will be generated by hosts in the stub domains, and thus, the applications running in these hosts will determine the traffic service requirements. We expect application diversity encompassing electronic mail, desktop videoconferencing, scientific visualization, and distributed simulation, for example. Many of these applications have strict requirements on delivery, delay, and bandwidth.

In such a large and heterogeneous Internet, the routing procedures must be capable of ensuring that traffic is forwarded along routes that offer the required services without violating domain usage restrictions. We believe that IDPR meets this goal; it has been designed to accommodate an Internet comprising $O(10^4)$ administrative domains with diverse service offerings and requirements.

2 An Overview of IDPR

IDPR generates, establishes, and maintains *policy routes* that satisfy the service requirements of the users and respect the service restrictions of the transit domains. Policy routes are constructed using information about the services offered by and the connectivity between administrative domains and information about the services requested by the users.

2.1 Policies

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 session 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 cost per byte, message, or session time.

2.2 Functions

The basic IDPR functions include:

1. Collecting and distributing routing information, i.e., domain transit policy and connectivity information. IDPR uses link state routing information distribution, so that each source domain may obtain routing information about all other domains.

2. Generating and selecting policy routes based on the routing information distributed and on source policy information. IDPR gives each source domain complete control over the routes it generates.

3. Setting up paths across the Internet, using the policy routes generated.

4. Forwarding messages across and between administrative domains along the established paths. IDPR uses source-specified message forwarding, giving each source domain complete control over the paths traversed by its hosts' traffic.

5. Maintaining databases of routing information, inter-domain policy routes, forwarding information, and configuration information.

2.3 Entities

Several different entities are responsible for performing the IDPR functions:

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

2. *Path agents*, resident within policy gateways, 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 willing to act on its behalf and a means by which the host's messages can reach that path agent.

3. Special-purpose servers maintain all other IDPR databases as follows:

   (a) Each *route server* is responsible for both its database of routing information, including domain connectivity and transit policy information, and its database of policy routes. Also, each route server generates policy routes on behalf of its domain, using entries from its routing information database and using source policy information supplied through configuration or obtained directly from 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 database and route database maintenance and the computationally intensive task of route generation.

   (b) Each *mapping server* is responsible for its database of mappings that resolve Internet names and addresses to administrative domains. The mapping server function can be easily integrated into an existing name service such as DNS.

   (c) Each *configuration server* is responsible for its database of configured information that applies to policy gateways, path agents, and route servers in the given administrative domain. Configuration information for a given domain includes source and transit policies and mappings

between local IDPR entities and their addresses. The configuration server function can be easily integrated into a domain's existing network management system.

2.4 Message Handling

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

3 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.

3.1 Integrity and Authenticity

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's source IDPR entity. The Internet coordinator specifies 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 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 control message integrity/authentication be based on a digital signature algorithm, such as MD5, which simultaneously verifies message integrity and source authenticity. The digital signature may be based on either public key or private key cryptography. However, we do not require that IDPR data messages carry a non-null integrity/authentication value. In fact, we recommend that a higher layer (end-to-end) procedure assume responsibility for checking the integrity and authenticity of data messages, because of the amount of computation involved.

3.2 Timestamps

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

must possess internal clocks that are synchronized to some degree, in order for the absolute value of a message timestamp to be meaningful. The synchronization granularity required by IDPR is on the order of minutes and can be achieved manually.

Each IDPR recipient of an IDPR control message must check that the message's timestamp is in the acceptable range. A message whose timestamp lies outside of the acceptable range may contain stale or corrupted information or may have been issued by a source whose clock has lost synchronization with the message recipient. Such messages must therefore be discarded, to prevent propagation of incorrect IDPR control information. We do not require IDPR entities to perform a timestamp acceptability test for IDPR data messages, but instead leave the choice to the individual domain administrators.

4 Size Considerations

IDPR provides policy routing among administrative domains and has been designed to accommodate an Internet containing tens of thousands of domains, supporting diverse source and transit policies.

In order to construct policy routes, route servers require routing information at the domain level only; no intra-domain details need be included in IDPR routing information. Thus, the size of the routing information database maintained by a route server depends only on the number of domains and transit policies and not on the number hosts, gateways, or networks in the Internet.

We expect that, within a domain, a pair of IDPR entities will normally be connected such that when the primary intra-domain route fails, the intra-domain routing procedure will be able to use an alternate route. In this case, a temporary intra-domain failure is invisible at the inter-domain level. Thus, we expect that most intra-domain routing changes will be unlikely to force inter-domain routing changes.

Policy gateways distribute routing information only when detectable inter-domain changes occur and may also elect to distribute routing information periodically as a backup. Thus, policy gateways do not often need to generate and distribute routing information messages, and the frequency of distribution of these messages depends only weakly on intra-domain routing changes.

IDPR entities rely on intra-domain routing procedures operating within domains to transport inter-domain messages across domains. Hence, IDPR messages must appear well-formed according to the intra-domain routing procedures and addressing schemes in each domain traversed; this requires appropriate header encapsulation of IDPR messages at domain boundaries. Only policy gateways and route servers must be capable of handling IDPR-specific messages; other gateways and hosts simply treat the encapsulated IDPR messages like any other. Thus, for the Internet to support IDPR, only a small proportion of Internet entities require special IDPR software.

With domain-level routes, many different traffic flows may use not only the same policy route but also the same path, as long their source domains, destination domains, and requested services are identical. Thus, the size of the forwarding information database maintained by a policy gateway depends on the number of domains and source policies and not on the number of hosts in the Internet. Moreover, memory associated with failed, expired, or disused paths can be reclaimed for new paths, and thus forwarding information for many paths can be accommodated.

5 Interactions with Other Inter-Domain Routing Procedures

We believe that many Internet domains will benefit from the introduction of IDPR. However, the decision to support IDPR in a given domain is an individual one, left to the domain administrator; not all domains must support IDPR.

Within a domain that supports IDPR, other inter-domain routing procedures, such as BGP and EGP, can comfortably coexist. Each inter-domain routing procedure is independent of the others. The domain administrator determines the relationship among the inter-domain routing procedures by deciding which of its traffic flows should use which inter-domain routing procedures and by configuring this information for use by the policy gateways.

Hosts in stub domains may have strict service requirements and hence will benefit from the policy routing provided by IDPR. However, the stub domain itself need not support IDPR in order for its traffic flows to use IDPR routes. Instead, a proxy domain may perform IDPR functions on behalf of the stub. The proxy domain must be reachable from the stub domain according to an inter-domain routing procedure independent of IDPR. Administrators of the stub and potential proxy domains mutually negotiate the relationship. Once an agreement is reached, the administrator of the stub domain should provide the proxy domain with its hosts' service requirements.

IDPR policy routes must traverse a contiguous set of IDPR domains. Hence, the degree of IDPR deployment in transit domains will determine the availability of IDPR policy routes for Internet users. For a given traffic flow, if there exists no contiguous set of IDPR domains between the source and destination, the traffic flow relies on an alternate inter-domain routing procedure to provide a route. However, if there does exist a contiguous set of IDPR domains between the source and destination, the traffic flow may take advantage of policy routes provided by IDPR.

6 Implementation Experience

To date, there exist two implementations of IDPR: one an independent prototype and the other an integral part of the *gated* UNIX process. We describe each of these implementations and our experience with them in the following sections.

6.1 The Prototype

During the summer of 1990, the IDPR development group consisting of participants from USC, SAIC, and BBN began work on a UNIX-based software prototype of IDPR, designed for implementation in

Sun workstations. This prototype consisted of multiple user-level processes to provide the basic IDPR functions together with kernel modifications to speed up IDPR data message forwarding.

Most, but not all, of the IDPR functionality was captured in the prototype. In the interests of producing working software as quickly as possible, we intentionally left out of the IDPR prototype support for source policies and for multiple policy gateways connecting two domains. This simplified configuration and route generation without compromising the basic functionality of IDPR.

The IDPR prototype software was extensively instrumented to provide detailed information for monitoring its behavior. The instrumentation allowed us to detect events including but not limited to:

1. Change in policy gateway connectivity to adjacent domains.
2. Change in transit policies configured for a domain.
3. Transmission and reception of link state routing information.
4. Generation of policy routes, providing a description of the actual route.
5. Transmission and reception of path control information.
6. Change of path state, such as path setup or teardown.

With the extensive behavioral information available, we were able to track most events occurring in our test networks and hence determine whether the prototype software provided the expected functionality.

6.1.1 Test Networks

In February 1991, the IDPR development group began experimenting with the completed IDPR prototype software. Each IDPR development site had its own testing environment, consisting of a set of interconnected Sun workstations, each workstation performing the functions of a policy gateway and route server:

- USC used a laboratory test network consisting of SPARC1+ workstations, each pair of workstations connected by an Ethernet segment. The topology of the test network could be arbitrarily configured.
- SAIC used Sun3 workstations in networks at Sparta and at MITRE. These two sites were connected through Alternet using a 9.6kb SLIP link and through an X.25 path across the DCA EDN testbed.
- BBN used SPARC1+ workstations at BBN and ISI connected over both DARTnet and TWBnet.

6.1.2 Experiments

The principal goal of our experiments with the IDPR prototype software was to provide a proof of concept. In particular, we set out to verify that the IDPR prototype software was able to:

1. Monitor connectivity across and between domains.
2. Update routing information when inter-domain connectivity changed or when new transit policies were configured.
3. Distribute routing information to all domains.
4. Generate acceptable policy routes based on current link state routing information.
5. Set up and maintain paths for these policy routes.
6. Tear down paths that contained failed components, supported stale policies, or attained their maximum age.

Furthermore, we wanted to verify that the IDPR prototype software quickly detected and adapted to those events that directly affected policy routes.

The internetwork topology on which we based most of our experiments consisted of four distinct administrative domains connected in a ring. Two of the four domains served as host traffic source and destination, $AD\ S$ and $AD\ D$ respectively, while the two intervening domains provided transit service for the host traffic, $AD\ T_1$ and $AD\ T_2$. $AD\ S$ and $AD\ D$ each contained a single policy gateway that connected to two other policy gateways, one in each transit domain. $AD\ T_1$ and $AD\ T_2$ each contained at most two policy gateways, each policy gateway connected to the other and to a policy gateway in the source or destination domain. This internetwork topology provided two distinct inter-domain routes between $AD\ S$ and $AD\ D$, allowing us to experiment with various component failure and transit policy reconfiguration scenarios in the transit domains.

For the first set of experiments, we configured transit policies for $AD\ T_1$ and $AD\ T_2$ that were devoid of access restrictions. We then initialized each policy gateway in our internetwork, loading in the domain-specific configurations and starting up the IDPR processes. In our experiments, we did not use mapping servers; instead, we configured address/domain mapping tables in each policy gateway.

After policy gateway initialization, we observed that each policy gateway immediately determined the connectivity to policy gateways in its own domain and in the adjacent domains. The representative policy gateway in each domain then generated a routing information message that was received by all other policy gateways in the internetwork.

To test the route generation and path setup functionality of the IDPR prototype software, we began a *telnet* session between a host in $AD\ S$ and a host in $AD\ D$. We observed that the *telnet* traffic prompted the path agent resident in the policy gateway in $AD\ S$ to request a policy route from its route server. The route server then generated a policy route and returned it to the path agent. Using the policy route supplied by the route server, the path agent initiated path setup, and the *telnet* session was established immediately.

Having confirmed that the prototype software satisfactorily performed the basic IDPR functions, we proceeded to test the software under changing network conditions. The first of these tests showed that the IDPR prototype software was able to deal successfully with a component failure along a path. To simulate a path component failure, we terminated the IDPR processes on a policy gateway in the transit domain, $AD\ T_1$, traversed by the current path. The policy gateways on either side of the failed policy gateway immediately detected the failure. Next, these two policy gateways, representing two different domains, each issued a routing information message indicating the connectivity change and each initiated path teardown for its remaining path section.

Once the path was torn down, the path agent agent in $AD\ S$ requested a new route from its route server, to carry the existing *telnet* traffic. The route server, having received the new routing information messages, proceeded to generate a policy route through the other transit domain, $AD\ T_2$. Then, the path agent in $AD\ S$ set up a path for the new route supplied by the route server. Throughout the component failure and traffic rerouting, the *telnet* session remained intact.

At this point, we restored the failed policy gateway in $AD\ T_1$ to the functional state, by restarting its IDPR processes. The restored policy gateway connectivity prompted the generation and distribution of routing information messages indicating the change in domain connectivity.

Having returned the internetwork topology to its initial configuration, we proceeded to test that the IDPR prototype software was able to deal successfully with transit policy reconfiguration. The current policy route carrying the *telnet* traffic traversed $AD\ T_2$. We then reconfigured the transit policy for $AD\ T_2$ to preclude access of traffic travelling from $AD\ S$ to $AD\ D$. The transit policy reconfiguration prompted both the distribution of routing information advertising the new transit policy for $AD\ T_2$ and the initiation of path teardown.

Once the path was torn down, the path agent in $AD\ S$ requested a new route from its route server, to carry the existing *telnet* traffic. The route server, having received the new routing information message, proceeded to generate a policy route through the original transit domain, $AD\ T_1$. Then, the path agent in $AD\ S$ set up a path for the new route supplied by the route server. Throughout the policy reconfiguration and rerouting, the *telnet* session remained intact.

This set of experiments, although simple, tested all of the major functionality of the IDPR prototype software and demonstrated that the prototype software could quickly and accurately adapt to changes in the internetwork.

6.1.3 Performance Analysis

We (USC and SAIC members of the IDPR development group) evaluated the performance of the path setup and message forwarding portions of the IDPR prototype software. For path setup, we measured the amount of processing required at the source path agent and at intermediate policy gateways during path setup. For message forwarding, we compared the processing required at each policy gateway when using IDPR forwarding with IP encapsulation and when using only IP forwarding. We also compared the processing required when no integrity/authentication value was calculated for the message and when the MD4 digital signature algorithm was employed.

Our performance measurements were encouraging, but we have not listed them here. We emphasize that although we tried to produce efficient software for the IDPR prototype, we were not able to devote much effort to optimizing this software. Hence, the performance measurements for the IDPR prototype software should not be blindly extrapolated to other implementations of IDPR. To obtain a copy of the performance measurements for path setup and message forwarding in the IDPR prototype software, contact Robert Woodburn (woody@sparta.com) and Deborah Estrin (estrin@usc.edu).

6.2 The *gated* Version

The SAIC and BBN members of the IDPR development group, who previously worked on the IDPR prototype, are now nearing completion of the task of integrating IDPR into the *gated* UNIX process. The *gated* version of IDPR contains the full functionality of IDPR together with a simple yet versatile user interface for IDPR configuration. As a single process, the *gated* version of IDPR should perform more efficiently than the multiple-process prototype version. The central respository for the *gated* IDPR software is cseic.saic.com; to obtain a copy of the current software, contact Robert Woodburn (woody@sparta.com).

Once completed, the *gated* version of IDPR will be freely available to the Internet community. Hence, anyone with a UNIX-based machine can experiment with IDPR, without investing any money or implementation effort. By making IDPR widely accessible, we can begin to gain Internet experience by introducing IDPR into operational networks with real usage constraints transporting host traffic with real service requirements.