

CONNECTIVITY DATABASE OVERHEAD FOR INTER-DOMAIN POLICY ROUTING

D. Estrin and K. Obraczka¹

Computer Science Department
University of Southern California
Los Angeles, California 90089-0782
estrin@usc.edu kobracczk@usc.edu

Abstract

Policy Routing protocols incorporate policy related constraints into the route computation and packet forwarding functions for inter-Administrative Domain (AD) communication. However, this new functionality exacerbates the already critical problem of routing information distribution and storage overhead in very large internets (e.g., 100,000 ADs). This paper investigates the scalability of the Inter-Domain Policy Routing (IDPR) architecture. In particular, we present an informal analysis of Connectivity Database and update overhead.

A model of the IDPR architecture is defined and exercised by varying several parameters. The results obtained illustrate the scaling properties of the IDPR architecture and their dependencies upon internet configuration, connectivity among ADs, and number of policies.

We find that, under certain reasonable assumptions, global information for an internet of 5,000 transit ADs (and 50,000 stub ADs) will occupy on the order of 2.5 Megabytes of storage in the route server. Since route servers need not be co-located with gateways, this requirement is not necessarily prohibitive. However, the connectivity database size predicted by the model is quite large in terms of computing routes over this database, and distributing updates to maintain the database. Future work must address these other critical dimensions of scaling as well.

1 Introduction

A new generation of routing protocols are under development to address problems of routing in very large scale internetworks. These protocols provide new functionality (in particular, resource control) that is needed when connections cross administrative domain (AD) boundaries.² However, this new functionality, referred to as **policy routing**, compounds the problems of protocol overhead and

¹Names listed in alphabetical order.

²An AD is a set of resources—hosts, networks, and gateways—that is governed by a single administrative authority. See [7, 6].

efficiency that are of critical concern when the number of internetwork routing elements is very large.

In this paper we investigate scaling for the Inter-Domain Policy Routing architecture.[9, 14] This architecture proposes to support a very rich notion of policy control for endpoint and transit networks. Other inter-AD routing protocols support more limited types of policy with less overhead.[11, 1, 5] Our purpose is to consider how large of a global internet can be accommodated by the relatively expressive IDPR architecture. The informal analysis presented in this paper clearly motivates the need for efficient route computation and update distribution techniques.

In the remainder of this section we give a brief overview of the IDPR architecture, and outline the dimensions of the architecture that are impacted by internet scale. After identifying specific questions associated with routing database and update overhead, we define a simple model of the architecture in 2 and exercise it to generate the results reported in Section 3. The significance of our investigation is discussed in Section 4.

1.1 Inter-Domain Policy Routing Overview

IDPR uses source routing, explicit advertisement of policy along with topology information, and a link-state style routing protocol. The source routes specify the ADs that are passed through along a source-destination path; the particular links and routers transited within ADs is transparent to the source. Source routes are computed based on topology and policy information advertised by each par-

3D.2.1.

icipating AD. Policy information specifies which traffic the particular AD will carry, e.g., an AD can limit the endpoints and the types of service made available. The routing updates (topology and policy information) are advertised to all AD-level routing elements so that source routes can be computed appropriately. However, the databases need not be consistent in every AD at all times because the source routing function is used to avoid loops.

When a packet is outgoing from one AD to another, and assuming an AD-level source route has been computed, a policy route is *setup* from the source AD to the destination AD. Successive packets between those two ADs, that meet the policy and type of service requirements of the setup packet, need not carry the full source route. Instead the successive packets are forwarded based upon a globally unique path ID and corresponding state information maintained in the AD boundary routers. In effect, the architecture allows routes to be installed *on demand* in policy gateways through the setup function. For further discussion see [3, 9, 14, 2].

The following definitions and terminology are used throughout the remainder of the paper.

- **Administrative Domain (AD):** a collection of links, routers, and end-systems governed by a single administrative authority.[6].
- **Stub ADs:** contain the communicating end-systems that are the endpoints of communication; stub ADs do not provide transit services and need not advertise policies.
- **Transit ADs:** provide transit services and must advertise their policies.
- **Hybrid ADs** offer transit services (typically to fewer neighbor ADs), as well as end-system access.
- **Policy gateway (PG):** router at the border of an AD that implements the IDPR protocol.
- **Virtual gateway (VG):** a set of (at least two) physical policy gateways that form a connection between two neighboring ADs. Every VG has two "sides", one in each of the ADs that it connects; in its simplest form

a VG consists of two PGs, one in each of the connecting ADs. The abstraction is introduced to reduce the amount of detailed (i.e., PG specific) status information that is distributed and maintained in the event that there are more than two PGs in a VG.

- **Route Server (RS):** entity within each AD that collects topology and policy information from other ADs and computes Policy Routes (see below) based on this information.
- **Policy Term (PT):** conditions specified by an AD for use of its transit resources. Policies may be in terms of endpoint ADs that are permitted to use the resource, type of service made available, or previous and next hop AD via which resources may be accessed, etc. For further discussion of policy types see [4]. Policy terms are carried in routing updates (see Update Protocol below).
- **Policy route (PR):** the AD-level source route from source to destination AD. In particular, a PR is a sequence of VGs with the associated PTs that permit use of the respective AD resource.
- **Connectivity database:** the database of policy and topology information used by the Route Server as input to the route computation. The database includes configured topology and policy information, as well as most-recently reported status (i.e., up or down) of particular VGs.
- **Route database:** the database generated by route computation and kept by the route server. Policy gateways query the route server for routes from this database when outgoing packets arrive for which no PR exists.
- **Forwarding information base:** the database of active PRs maintained by PGs in order to forward packets along active PRs.
- **Setup and data protocols:** the protocols used to set up PRs and forward data packets along established PRs.
- **Update protocol:** distributes configured and status information about topology and policy. A link state protocol in which every

3D.2.2.

AD advertises information about its neighbor connections to all ADs in the internet.³

- **Virtual gateway protocol:** carries information among the PGs in an AD to determine the status of VGs that should be advertised in the update protocol.

1.2 Scaling Dimensions

The functionality provided by IDPR comes at a price. In particular, there are four dimensions of scaling that require serious study. As defined above, each AD must have access to a Route Server (RS) that maintains a connectivity database of topology and policy information. This database must have information about every transit AD in the internet in order to guarantee valid policy routes to all destination ADs in the internet. For the sake of this paper we will assume that every AD maintains a complete connectivity database of configured policy and topology information, and we will investigate the memory requirements that are thus implied.⁴

In order to maintain such a global database, policy and topology information must be flooded throughout the internet. Therefore the second dimension of IDPR scaling is the **Update Distribution Overhead**, i.e., the network resources consumed by distributing updates. Update distribution is partially a function of the update size (which will be discussed in this paper), but is largely affected by the flooding protocol and the internet topology and is a subject for further study.

Related to the size of the connectivity database and to the nature of the topology and policies supported, is the **Route Synthesis Overhead**; in other words, the average time complexity to compute policy routes. This is an area that demands significant research attention but will not be discussed directly here. However, we hope to shed some light on this scaling dimension since one fac-

³For most of this paper we assume a flat space of ADs across which all update information is flooded (i.e., no hierarchy).

⁴The question of global distribution of dynamic status information can be considered separately and has primary impact on the update protocol overhead; the size of the database is approximately the same in either case.

tor in route computation time is the size of the connectivity database. Moreover, the methods used in this paper to model policy and topology are applicable to analysis of the route synthesis problem as well.

A fourth dimension of scale is the size of the **forwarding information base** that PGs must maintain in order to forward packets. This dimension is as much a function of traffic patterns as of internet size. PG state requirements and caching strategies are under investigation, but are beyond the scope of this paper.

The primary purpose of this paper is to investigate the scaling properties of inter-domain routing mechanisms. We focus on the first dimension of scaling, **Connectivity Database size** and include a brief analysis of update overhead. The other two dimensions are the subject of ongoing work and will be addressed elsewhere. Based on requirements articulated in [10], we choose as our scaling target an internet that includes on the order of 10^5 stub ADs and 10^4 transit ADs, i.e., well beyond the size of the current Research Internet!

1.3 Connectivity Database Size Factors

It is fundamental to the functionality of the IDPR protocol that policy routes are computed based on global information about transit AD topology and policies. However, as the global internet grows, the size of this database also grows. This study investigates the limits of an architecture based on route servers with complete global information.

The size of the database is not simply a function of the number of ADs in the internet. Because of the policy information distributed, database size is also a function of **internet configuration** (e.g., the number of transit and stub ADs), the **connectivity among ADs**, and **number of policies expressed by each AD**. Internet configuration determines the number of advertising nodes and also affects the average AD connectivity. AD connectivity and policies together determine the amount of information held for each AD in the internet.

Different models of the future global internet lead to different assumptions regarding configura-

tion, connectivity, and policy. We assume a hybrid model in which private networks will coexist along side commercial carrier networks. Consequently, we envision an extensive hierarchical structure (i.e., campus networks connected to metropolitan area networks (MANs), MANs connected to regionals, and regionals connected to wide area network (WAN) backbones), but with significant and persistent lateral and vertical **bypass** links (i.e., special purpose connections between campuses, and special purpose connections between campuses and wide area networks, respectively).

This hybrid topology will require support for a range of policies. Many transit networks will express policies regarding type of service and charging policies. Hybrid ADs are likely to express access control related policies as well in order to control the use of their internal resources via bypass links. Therefore, an inter-AD routing architecture will be used to support several types of policies, e.g., those based on type of service, charging, AD source, User Classification, etc. For a detailed discussion of different policy requirement models, see [4].

1.4 Comparison to BGP Connectivity Database Requirements

Before describing our model of connectivity database size we describe similar requirements for an alternative inter-domain routing protocol that supports a more limited notion of policy, with less overhead. The Border Gateway Protocol (BGP)[11] is based on hop-by-hop routing and a distance vector protocol in which complete AD-path information is distributed with each routing update entry. BGP supports a range of path and destination based policies but was not intended to support source-specific policies, i.e., transit policies that discriminate according to the source of the packet.⁵ Source specific transit policies can be supported only by replicating the Forwarding Information Base (FIB) in each border gateway. Keeping these limitations in mind, we identify the approximate connectivity database size required to support BGP in a large

⁵Stub policies regarding path preferences are supported to a limited extent. Stubs can choose from among a number of available paths according to local criteria. However, there is a much smaller set of routes available to the source than in the IDPR architecture.

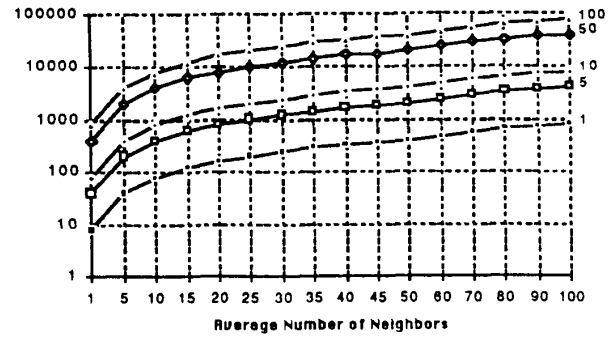


Figure 1: *BGP connectivity database size estimate (in KBytes) relative to the average number of neighbors per AD. Graphs represent different numbers of TOSs.*

internet with TOS policies.

The BGP architecture could be extended to very large internets by adding additional level(s) of hierarchy (i.e., grouping ADs) to reduce the overhead of routing information distribution and storage. For example, in an internet of 50,000 ADs we could group ADs into 500 AD groups and thereby reduce the FIB storage requirements by two orders of magnitude. This grouping mechanism is proposed in [1].

BGP's FIB maintains only one entry per destination-AD. However, the connectivity database out of which the FIB is generated is somewhat larger. In particular, BGP stores the most recent update from each neighbor. Moreover, if multiple types of service (TOS) are supported, then a routing update is maintained per-neighbor, per-TOS. Although TOS is not addressed in the BGP specification explicitly, it appears to be a straight forward extension of the architecture. The overall memory requirements for the connectivity database (CDB) can be expressed as:

$$CDB = N_{tos} * N_{nbors} * ADs_{top} * entry - size,$$

N_{tos} is the number of type of services supported, N_{nbors} is the number of neighbors of a top-level AD, ADs_{top} is the total number of top-level ADs in the internet. Figure 1 illustrates this function in terms of bytes; where *entry - size* is the average number of bytes in each BGP database entry.

3D.2.4.

We use a log scale to facilitate presentation of the curve. The graph assumes 5,000 transit ADs and 50,000 stub ADs. ADs are grouped such that there are approximately 500 top level ADs. To calculate the entry size, we assume that each BGP update carries the following information: origin, sequence of domains that constitutes the path, next hop, metric. If we assume an average path length of 5 ADs, 2 bytes per AD identifier, and 2 bytes for the metric, the average path information requires 16 bytes. For 10 types of service and an average of 10 neighbors per top-level AD, the approximate database size for a top-level AD border gateway is 800 KBytes. This analysis does *not* take into account the space used for network address lists in the current version of BGP. We assume that future BGP-like protocols would either advertise AD level information and/or collapse information using destination sets as in [1].

As stated earlier, it is somewhat misleading to compare BGP and IDPR because they provide different functionality. Nevertheless, with this in mind, it is useful to use BGP's connectivity database requirements as a reference point in our analysis of IDPR.

2 The model

This section describes our model of IDPR connectivity database size.

Traditional link state protocols maintain a connectivity database at each node. The database consolidates information based on one update per network node. Each node update contains the neighbor list for that node and associated status (up/down) and metric information. The IDPR connectivity database consolidates similar information per node (in this case per AD). However, the amount of information carried in each update may be much larger and is not simply a function of the number of neighbors per node (i.e., it includes the policy term information described earlier).

The number of entries per IDPR update is a function of policy, as well as internet configuration and AD connectivity, for example:

- For each VG represented, there may be mul-

iple entries in the database, one for each of the Policy Conditions specified for the VG. Actually, policy conditions are applied to VG pairs—specifying which VG the transit AD will carry traffic to and from.

- If special policy conditions are applied to each VG pair, then there will be a larger number of entries in the ADs update. If policy is more uniform then a single policy condition will be applied to communication among a set of VGs and the update size is reduced.

2.1 Expressions for the size of the Connectivity Database

We can begin by specifying the size of the connectivity database, CDB, as:

$$(1) CDB = f(ADs, ConD, Policies),$$

where ADs is the number of ADs, and ConD is the connectivity degree of an AD, and Policies is the number of policies expressed by each AD.

Since stub and hybrids are the ones that need the connectivity database to synthesize routes, and transit ADs are assumed to be the ones advertising policy conditions, we write this as:

$$(2) CDB_{ST-HY} = f(AD_{TR}, ConD_{TR}, Policies_{TR}),$$

where AD_{TR} , $ConD_{TR}$, and $Policies_{TR}$ are the number of transit ADs, the average connectivity degree per transit AD, and the average number of policies expressed by a transit AD.

Two different analyses were conducted for the storage overhead. In the first case, an average number of policies per AD is considered, while in the second case, each transit AD expresses a policy for each pair of stub ADs. Although the second case does not represent a realistic scenario, it is provided to motivate the need for attention to database size.

2.2 Database entry format and size

In the IDPR architecture, transit AD policies appear in the routing updates and connectivity database as sets of policy conditions and the corresponding sets of VGs within the AD that support the policy conditions. We refer to the policy set and the VG set together as a policy term entry. Therefore, the $ConD_{TR}$ and $Policies_{TR}$ terms in equation (2) are expressed together as the average number of policy terms advertised by a transit AD.

In the average case, we will consider that each stub and hybrid AD's connectivity database stores an average number of policy term entries, PTs , per transit AD. The size of a PT entry is represented by, PT_{entry} . A complete copy of the policy and topology information in the connectivity database can therefore be represented as:

$$(3) CDB_{ST-HY_{av}} = AD_{TRs} * [PTs * PT_{entry}]^6$$

The maximum AD memory requirements arise when each transit AD specifies different policy conditions for carrying traffic between each pair of stub ADs, as well as between each pair of VGs belonging to the AD. Therefore, the second, pessimistic, case considers all possible combinations of stub AD pairs ($ST * [ST - 1]$) and VG pairs ($VGs * [VGs - 1]$); where ST is the number of stub ADs. Equation (2) must be rewritten as

$$(4) CDB_{ST-HY_{ps}} = AD_{TRs} * [ST * [ST - 1]] * [VGs * [VGs - 1]] * PT_{entry}$$

The number of VGs can be expressed as a constant times the number of neighboring ADs ($c * nbors$). For our initial analysis we assumed that there is a single VG per neighboring AD, i.e., $c = 1$.

⁶ AD_{TRs} should be added to the total equation (3) to represent the space needed for each transit AD identifier. However, the space is negligible compared to the rest of the equation and thus we leave it out for simplicity.

⁷While most AD neighbors will be represented by a single Virtual Gateway, some neighbor ADs may support several Virtual Gateways in order to specify different policies regarding the usage of different entry/exit points to a particular AD neighbor (for example, to differentiate between an east and west coast gateway connection between a private corporate network and the Research Internet). The upper bound on the number of VGs per AD can be expressed as the number of physical neighbors multiplied by the number of VGs per AD neighbor.

In order to obtain the size of the connectivity database in bytes, the size of the policy term entry PT_{entry} must be expressed in terms of number of bytes. As previously mentioned, a PT_{entry} is composed of a set of policy conditions and the set of VGs that support them. Therefore, we can write

$$(5) length[PT_{entry}] = Pconds_{set} * length[Pcond_{entry}] + VG_{set} * length[VG_{entry}]^8$$

$Pconds_{set}$ is the average number of policy conditions in the policy set of a policy term entry. $Pcond_{entry}$ is the average number of bytes used to represent a single policy condition within the policy set. Analogously, VG_{set} and VG_{entry} stand for the average number of VGs in a VG set and the number of bytes used to represent a VG, respectively.

For this analysis we make some assumptions about the length of the average VG set. The first assumption is that most policies are applied uniformly to all VGs in an AD. This means that VG_{set} is equal to VGs , the number of VGs in an AD. This assumption is pessimistic with respect to the length of PT_{entry} because it implies that all or most of the VGs in an AD will be listed in each of the VG sets in the AD's policy terms. If policies were less uniform, each policy condition would apply to only a small subset of the VGs in that AD. However, if policies were less uniform, it is likely that the average number of policy terms expressed by an AD would be proportionally larger.

Another factor that can influence the average VG set length is the degree of *VG pairing* supported across the AD. Communication may be supported only between select pairs of directly-connected ADs, i.e., only a subset of physically possible previous and next hops can be used together. In this case there will be a smaller number of VGs specified in a VG set, and therefore in the global connectivity database. Most transit ADs will support 100% VG pairing. However, hybrid ADs, for example, may restrict the use of bypass links and support less than 100% VG pairing. For our analysis, we assume 100% pairing for all transits. If there is less VG pairing, the number of VGs listed in each policy term would be smaller.

The length of the policy condition entry varies

⁸ $length[X]$ stands for the number of bytes of entry X .

3D.2.6.

depending on the particular policies expressed. Each AD identifier is assumed to be 2 bytes long, while User Classification is represented using 1 byte. Other components of a policy condition such as charging and other general conditions are variable in length. Therefore, whereas a policy condition that specifies a single ADsrc, ADdst, and User Classification can be represented with 5 bytes, more elaborate policies would obviously take more space.

The VG entry is composed of one AD identifier (the VG is assumed to connect to the AD that advertises it, so the one AD identifier indicates the other AD to which the VG connects), plus a one byte number to distinguish the VG from other VGs that might connect the two ADs. VG_{entry} is therefore 3 bytes long.

Using these values, we represent our results in terms of memory requirements, MR by converting CDB in equations (3), (4), (6), and (8) into bytes. However, first, we discuss three optimizations to the overall approach.

2.3 Optimizations

Three optimization techniques were considered for both the average-case and the pessimistic-case analysis to investigate how the storage overhead can be reduced.

2.3.1 Policy Condition Encoding

Policy condition encoding uses small numerical identifiers to represent policy conditions. The size of PT_{entry} used in equations (3) and (4) is thereby reduced, since the size of a policy condition identifier is assumed to be smaller than the representation of the policy condition itself. It is reasonable to assume that this optimization will be used in any implementation of the protocol. However, the mapping of these identifiers to the actual policy conditions must be distributed periodically by the update protocol and each RS must maintain this information for each transit AD. Therefore, the mapping of policy conditions to their corresponding identifiers has to be kept along with the connectivity database.

Assuming a policy condition identifier, $PCID$, is 2 bytes, $Pcond_{entry}$ in equations (5) is reduced to 2 bytes. The estimate of overall memory requirements is augmented to take into account the mapping information:

$$(6) \ CDB_{mapping} = [ADTRs * Pconds_{AD}] * [Pcond_{entry} + PCID]$$

Where $Pconds_{AD}$ is the average number of policy conditions in an AD. The average mapping entry length is the length of an average policy condition $Pcond_{entry}$ plus the length of the corresponding policy condition identifier, $PCID$.

In the worst case, the average number of policy conditions in an AD, $Pconds_{AD}$, is equal to the average number of policy terms, PTs , times the average number of policy conditions in a policy set, $Pconds_{set}$. On the other hand, the best case occurs when $Pconds_{AD}$ is equal to $Pconds_{set}$. Encoding is efficient when policy conditions are referred to multiple times within a single update (i.e., $Pconds_{AD} < Pconds_{set} * PTs$). More importantly, encoding provides a significant reduction in dynamic update size since dynamic updates do not carry PCID mapping information. Therefore encoding is clearly advantageous when dynamic update distribution overhead is considered.

To take into account the influence of encoding, the mapping database size $CDB_{mapping}$ as expressed in equation (6) must be added to $CDB_{ST-HYav}$.

2.3.2 VG set encoding

If most policies are applied uniformly to all VGs, VG set encoding could be used. In other words, to avoid explicit listing of VGs in each policy term, the sets of VGs commonly referred to in policy terms would be assigned identifiers; e.g., one identifier would indicate all VGs belonging to an AD while other VG identifiers would indicate particular subsets of VGs. However, because two VG sets must have unique identifiers, even if they differ in only a single VG, the number of VG sets identified could grow to be very large and the number of VGs listed in each entry of the CDB mapping information message would also be large. Therefore, it is possible that the mapping information for VG

encoding, could become excessive.

We will not consider this optimization further in this paper. However, for implementation purposes it might be worth considering encoding of a few special VG set identifiers, e.g., a wild card to indicate all VGs registered for the advertising AD.⁹

2.3.3 Policy Filtering

Policy filtering assumes that a stub or hybrid AD can pre-filter and discard connectivity information that is not applicable (e.g., discard policies whose AD source field precludes usage by the stub or hybrid AD in question). For our initial investigation we assumed that a certain percentage, G , of all transit AD policy conditions are applicable to all stub ADs. Consequently, the average stub AD that does filtering would store $G\%$ of the total number of policy conditions expressed by all transit ADs, plus the additional source-specific policy conditions that apply to the stub AD. If we assume that source-specific transit ADs' policies are equally distributed among all the stubs and hybrids, then the number of source-specific transit policies that apply to a particular stub or hybrid AD, X , is the total number of source-specific transit AD policies divided by the total number of stub, ST , and hybrid, HY , ADs.

Policy filtering directly affects the size of the PT_{entry} , by reducing the number of policy conditions, $Pconds_{set}$, that need to be stored by a particular stub or hybrid AD. To incorporate the effects of policy filtering, equation (5) is rewritten as follows.

$$(7) \text{ length}[PT_{entry}] = (G\% * Pconds_{set} + (1 - G\%) * Pconds_{set} / (ST + HY)) * \text{ length}[Pcond_{entry}] + VGs_{set} * \text{ length}[VG_{entry}]$$

Assuming encoding is used in conjunction with filtering, the MR calculation must also include the corresponding $CDB_{mapping}$.

⁹If this mechanism is implemented, our analysis is an overestimate to the extent wild card VG sets are used in practice.

2.4 Update distribution overhead

Global flooding of update information also presents scaling problems. In this section we provide a very simple model of update distribution overhead.

To analyze update overhead we need to make additional assumptions about the number of PGs in an AD, PGs_{AD} , and the number of inter-AD links supported by each of the PGs, PG_{links} . We will represent update overhead by the number of update packets handled by each PG in the internet, $Update_{PG}$.¹⁰ We assume that every transit AD generates a dynamic update at an average frequency of $update_{freq}$. Each PG receives one original update and some number of duplicates (the number may be 0) as a result of the flooding protocol and the non-hierarchical topology. Each PG will receive at most one duplicate copy of each update from each of the PGs to which it is connected in neighboring ADs. In the worst case, the PG might also receive a duplicate from each of the other PGs in its own AD, $PGs_{AD} - 1$. Therefore, we write the worst case update distribution overhead per PG as in equation (8).

$$(8) Update_{PG} = AD_{TRs} * update_{freq} * [PGs_{links} + (PGs_{AD} - 1)]$$

Using equation (8) we can represent the worst case intra-AD bandwidth consumed by update distribution, $Update_{ADlinks}$. Intra-AD links carry the updates among the PGs within an AD.

$$(9) Update_{ADlinks} = AD_{TRs} * update_{freq} * update_{size} * [PGs_{AD}(PGs_{AD} - 1)]$$

From equations (3) and (6), we know $update_{size} = PTs * PT_{entry}$. However, equation (9) must be combined with equation 6 to represent the use of policy condition encoding and the distribution of associated mapping information in configuration updates. We assume that configuration updates are sent at an even lower frequency than regular updates.

¹⁰For this simple analysis we assume that each PG sees the same update overhead, as a result of the global flooding protocol.

3D.2.8.

$$(10) \text{Update}_{ADlinks} = AD_{TRs} * PGs_{AD} (PGs_{AD} - 1) * [\text{update}_{freq} * PTs * PT_{entry} + \text{configupdate}_{freq} * [PTs * PT_{entry} + Pconds_{AD} * (Pcond_{entry} + PCID)]]$$

In general operation, there are mechanisms in the IDPR update protocol that will make this worst case scenario, of multiple duplicate updates from every possible neighbor, very unlikely.

2.5 What the model tells us

The numerical results shown in the next section illustrate how the storage (i.e., route server's memory requirements) and update overhead vary with several of the parameters described above:

- The number of ADs.
- The number of neighbors (and VGs) per AD.
- The number of policy conditions per VG pair.
- The percentage of transit ADs.

Equations (3), (4), and (6) are refined by decomposing the transit ADs into backbones, regionals and hybrids. The goal is to have a better intuition for the values assigned to the variables in the expressions of CDB_{ST-HY} . For example, backbones are expected to support a much larger number of neighbors (i.e., VGs) than hybrids do, while many hybrids may support a larger number of fine grain policies.

In our model, the average number of neighbors per backbone was calculated as the ratio between the number of regionals and the number of backbones. This assumes that on the average, each regional is connected to one backbone. Analogously, the average number of neighbors per regional was calculated as the ratio between the number of stubs and the number of regionals, plus 1 (the neighboring backbone). For simplicity, the average number of neighbors per hybrid was assumed to be the same as the number of neighbors per regional; although this is probably an overestimate.

As an example, equation (8) below is the result of this decomposition applied to equation (3).

$$(11) CDB_{ST-HY_{av}} = BB * [PTs_{BB} * PT_{entry-BB}] + RG * [PTs_{RG} * PT_{entry-RG}] + HY * [PTs_{HY} * PT_{entry-HY}]$$

Where BB , RG , and HY are the number of backbones, regionals, and hybrids, respectively. PTs_{BB} , PTs_{RG} , and PTs_{HY} are the average number of policy terms expressed by backbone, regional, and hybrid ADs, respectively. $PT_{entry-BB}$, $PT_{entry-RG}$, and $PT_{entry-HY}$, are the average sizes of a policy term entry for a backbone, regional, and hybrid AD's, respectively.

For our calculations we assumed that 1% of the transit ADs were backbones, 5% were hybrids, and the remaining majority were regional (i.e., mid-level) transit networks.

3 Data

Using the model described above, we generated several graphs to illustrate the relationship between various parameters, and the overall scaling properties of the architecture. The model is not intended to predict precise numeric values.

Figure 2 shows the IDPR architecture scalability problem in the pessimistic, and unrealistic, case when transit ADs specify a different policy restriction for carrying traffic between each pair of stub ADs, as well as between each pair of VGs that connect to the AD (see equation (4)). Therefore, for this case, we are assuming one policy condition, and two VGs per PT_{entry} (i.e., $Pconds_{set} = 1$, and $VGset = 2$ in equation (5)). We also assume that the average policy condition entry is 5 bytes (i.e., 5 bytes is required to represent an two AD endpoints and a User Classification). Consequently, for equation (4), the length of a PT_{entry} is 11 bytes. The graph shows that even using encoding, the size of the database becomes enormous. It achieves $2 * 10^{10}$ Mbytes at 5,000 transits, and $1.6 * 10^{11}$ Mbytes at 10,000 transits.

Figure 3 shows that when an average number of policies per transit AD is considered, the memory requirements become considerably smaller (see equation (3)).

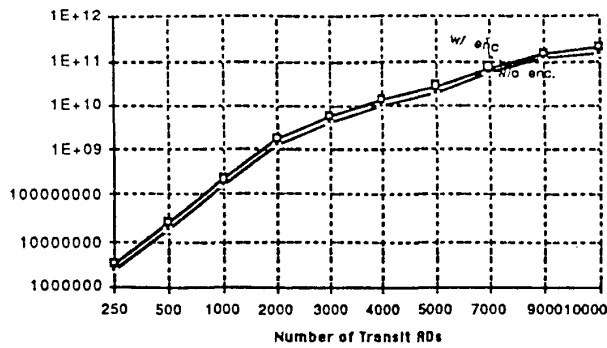


Figure 2: Pessimistic-Case Equation (4). Different policy condition for each stub AD pair. Variation of MR (in MBytes) with the number of transits, with and without encoding. Other parameters and values: 10% of the total number of ADs are transit ADs; number of neighbors (equal to the number of VGs): 95, 10, 10 for backbones, regionals and hybrids, respectively; $PT_{entry} = 11$ bytes.

This graph shows the influence of the number of policy conditions supported by each transit AD, and of the internet configuration (number of transits). The AD connectivity (number of neighbors) was kept constant at 95, 10, and 10 neighbors for backbones, regionals and hybrids, respectively. As expected, MR increases with the number of policy conditions, and the number of transits. As an example, for 5,000 transits (50,000 stub ADs), and an average of 10 policy term entries with 3 policy conditions per entry (and 5 bytes per policy condition entry) for each transit AD, MR is approximately 2.4 MBytes. For the same number of transits and an average of 100 policy term entries, MR increases by an order of magnitude to 24 Mbytes. These numbers do not use encoding of PTs.

Figure 4(a) expresses the same variation as Figure 3, except that PT encoding is used (see equations (3) and (5)). Figure 4(b) represents the mapping database size in terms of $Pcond_{AD}$ (see equation (6)).

Total connectivity database size for the case with encoding must therefore add together values from these two graphs. Comparing the results with the non-encoded case, the shape of the curves remain the same. For 5,000 transit ADs and 10 policy term entries, and an average $Pconds_{set} = 3$, $Pcond_{entry} = 5$ bytes, and $Pconds_{AD} = 10$, the ap-

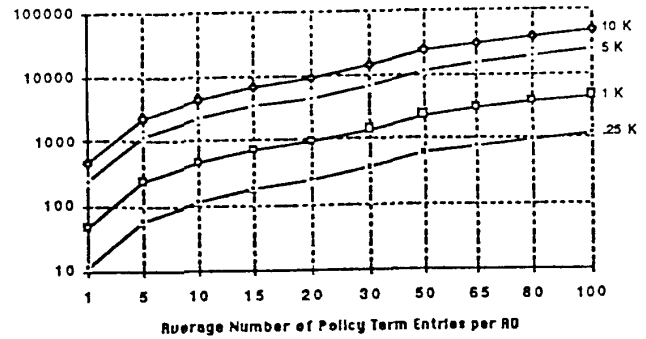


Figure 3: Average-Case Results. Variation of MR (in KBytes) with the average number of policy term entries per AD, for different numbers of transits. Other parameters and values: 10% of the total number of ADs are transits; 1% and 5% of transits are backbones and hybrids, respectively; number of VGs (equal to the number of neighbors) are: 95, 10, 10 for backbones, regionals, and hybrids, respectively; average $Pconds_{set} = 3$ and $Pcond_{entry} = 5$ bytes.

proximate connectivity database size is 2.5 MBytes.

Figure 5 shows the variation of the connectivity database size as the size of a policy condition entry increases. For $Pcond_{entry} = 10$ bytes, $PTs = 20$, and $Pconds_{AD} = 20$, $CDB = 5$ Mbytes; if encoding is not used, $CDB = 6.2$ Mbytes.

However, the real advantage of PT encoding becomes evident when distribution of dynamic updates is considered. Dynamic updates do not carry mapping information and are distributed more frequently than the configuration updates; the latter incur the mapping overhead associated with PT encoding. Therefore the size of the most frequently distributed information is reduced using PT encoding.

In Figure 6, MR is presented as a function of the percentage transits for different number of ADs (equations (3) and (5)).

As the percentage of transits increases, MR also increases. For example, if the future global internet has between 5 and 10% transit ADs, MR will be less than 3 Mbytes (for 50,000 ADs and 10 policy term entries of the size assumed above, and including the mapping database with $Pconds_{AD} = 20$).

3D.2.10.

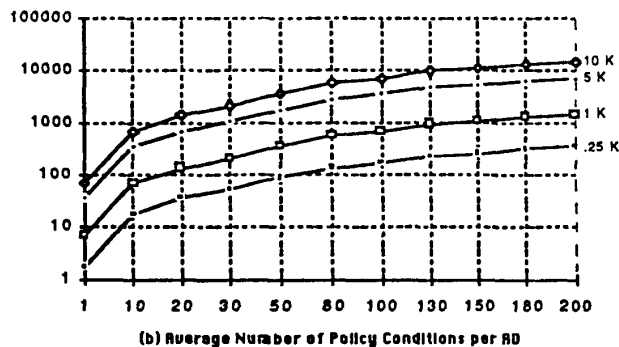
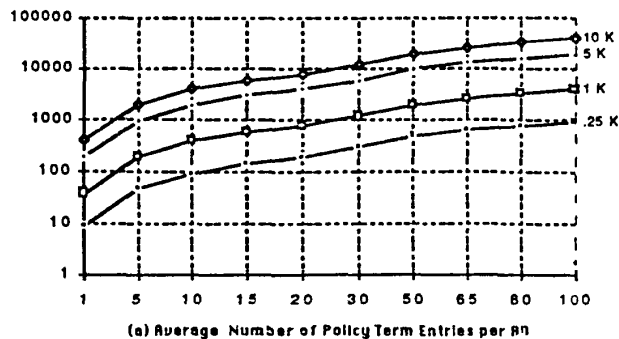


Figure 4: Average-Case Analysis using Encoding of PTs. (a) Variation of MR (in KBytes) with the average number of policy term entries per AD, for different numbers of transit ADs. (b) Variation of MR (in KBytes) for mapping database with the average number of policy conditions defined per AD, for different numbers of transit ADs. Other parameters and values: 10% transit ADs; 1% and 5% of transits are backbones and hybrids, respectively; number of VGs (equal to the number of neighbors) are: 95, 10, 10 for backbones, regionals, and hybrids, respectively; average $Pcond_{set} = 3$ and $Pcond_{entry} = 5$ bytes.

In Figure 7, the effects of policy filtering are analyzed. We present MR as a function of the number of policies, since it is the parameter directly affected by the optimizations. In this case an AD does not bother storing information that it can not use, i.e., policy terms that exclude it as an endpoint AD.

Policy filtering constrains the variation of MR as a function of the number of policies, since it reduces the number of policy conditions, i.e., the parameter, $Pcond$, in equation (7) is multiplied by $G\%$. For example, if G is 50% and $PTs = 10$, MR is 2.0 Mbytes when $Pcond_{set} = 3$ and $Pconds_{AD} = 10$ (including the filtered mapping

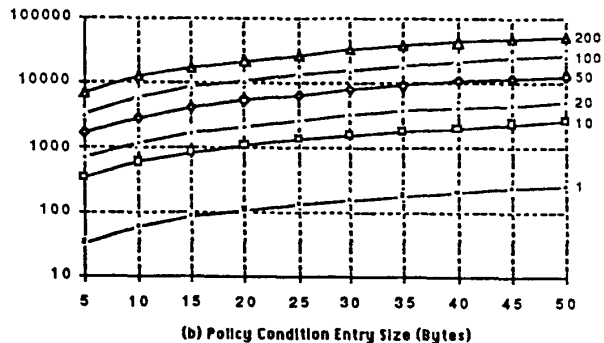
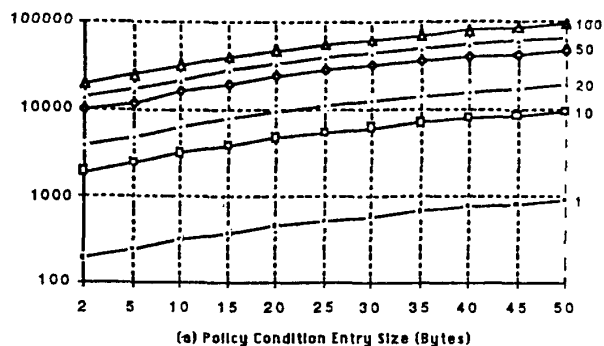


Figure 5: Average-Case Analysis with PT Encoding. (a) Variation of MR (in KBytes) with the size of an average policy condition entry and the number of policy conditions per AD. (b) Variation of MR (in KBytes) for mapping database with the size of an average policy condition entry and the number of policy conditions per AD. Parameters and values: 1% and 5% of transits are backbones and hybrids; average $Pcond_{set} = 3$ and $Pcond_{entry} = 5$ bytes in the mapping and 2 bytes in the update itself.

database). Therefore, when 50% of the policy conditions are endpoint specific, filtering can result in a 20% savings in storage.

In Figure 8, we illustrate the number of update distribution packets handled by each PG in the internet as a function of the update frequency and the average number of PGs per AD.

For example, assume there are 5,000 transit ADs, updates are generated once a day, each PG is connected to 5 PGs in other ADs, and each AD has an average of 10 PGs. In this case, there are an average of 5,000 updates sent per day. Each PG will receive at most one copy of each update from each of the PGs to which it is connected in neighboring ADs (5 in this example). In the worst case each PG also receives one copy of the update from

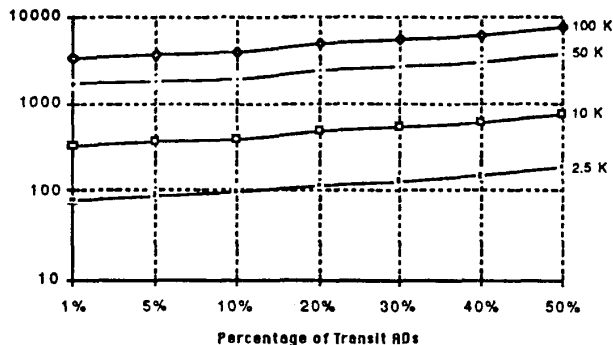


Figure 6: Average-Case Analysis with PT Encoding. Variation of MR (in KBytes) with the percentage of transit ADs for different numbers of ADs. Parameters and values: 1% and 5% of transits are backbones and hybrids; 10 policy term entries per AD; average $Pcond_{set} = 3$ and $Pcond_{entry} = 5$ bytes in the mapping and 2 bytes in the update itself.

each of the other PGs in its own AD (9 in this example). Therefore, in this pessimistic example, each PG would see at most 14 duplicates of each new update—averaging to 50 per minute or less than 1 per second. The worst case aggregate intra-AD link bandwidth consumed using these assumptions is 3 Kbytes per second, assuming that on average, $PT_s = 10$ and $PT_{entry} = 40$ bytes.

4 Discussion

According to our model, for an internet of 5,000 transit ADs (and 50,000 stub ADs), and an average of 10 policy term entries per AD, global information will occupy approximately 2.5 Mbytes of storage in the route server. Since the route server need not be co-located with policy gateways, this requirement is not necessarily prohibitive. However, the connectivity database size predicted by the model is quite large in terms of computing routes over this database, and distributing updates to maintain the database. Thus particular attention is needed to develop efficient route computation and update distribution techniques. To conclude, we discuss briefly some possible methods of addressing these scaling issues.

One possibility is to introduce additional level(s)

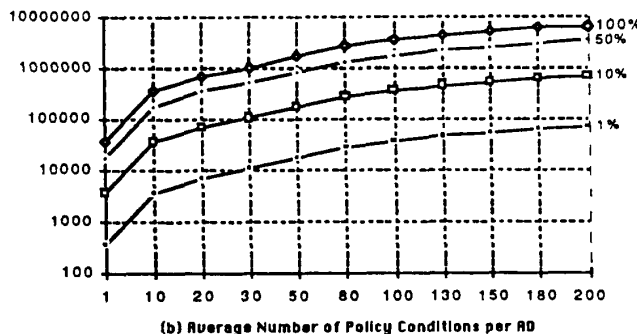
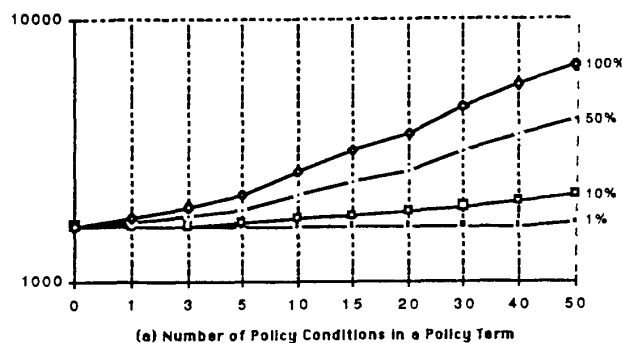


Figure 7: Average-Case Results (in KBytes) with PT Encoding, and Policy Filtering. (a) Variation of MR with the number of policy conditions per AD, for different percentages of G ($G\%$ of the policy conditions are assumed to be applicable to all stub ADs). (b) Variation of MR for mapping database, for different percentages of G . Parameters and values: 5,000 transits; 1% and 5% of transits are backbones and hybrids, respectively; number of VGs per AD: 95, 10, 10 for backbones, regionals, and hybrids, respectively; 10 policy term entries per AD; average $Pcond_{set} = 3$ and $Pcond_{entry} = 5$ bytes in the mapping and 2 bytes in the update itself.

of hierarchy in the internet configuration so that information distributed globally across the entire internet is at an even higher level of abstraction (i.e., AD groups). However, we are at risk of losing the benefits of IDPR flexibility if ADs are grouped together. IDPR is a very expressive protocol for representing policy when the units of abstraction (e.g., ADs or AD groups) reflect policy. In other words, a transit policy expressed for a unit must reflect the transit policy of all constituent components. And, a source-specific policy applied to a unit, must apply to all components of that unit. Therefore, if, for example the ADs that make up some AD-group, X , do not have similar policies

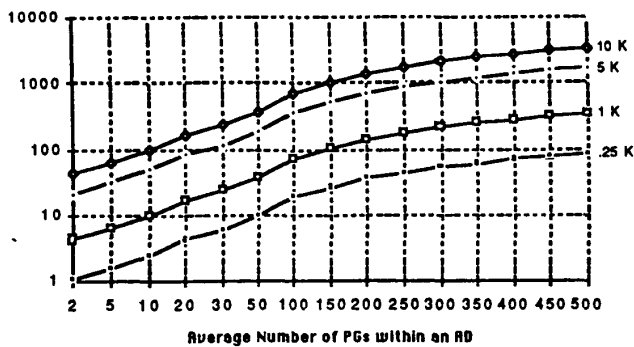


Figure 8: Update distribution overhead per PG (in number of update packets per minute), for different numbers of transit ADs. Parameters and values: 1% and 5% of transits are backbones and hybrids, respectively; number of VGs per AD: 95, 10, 10 for backbones, regionals, and hybrids, respectively; 10 policy term entries per AD; average $Pcond_{set} = 3$ and $Pcond_{entry} = 5$ bytes in the mapping and 2 bytes in the update itself.

with respect to ADs outside of X , or if other AD groups do not have the same policy with respect to all the ADs within X , then the expressiveness of the IDPR architecture is compromised. Consequently, we conclude that the highest level of abstraction that should be applied globally is the AD. Some like minded ADs may find it effective to group themselves, and the IDPR architecture does not preclude this. However, we can not assume this approach will be applicable to the general problem of scaling. A possible alternative is to use AD level abstraction and IDPR among ADs within large regions (e.g., the U.S.), and to integrate across these regions with coarser grained policy and an inter-AD protocol such as BGP or coarse-grained IDPR.[11, 1].¹¹ This approach warrants further study.

Even if we maintain a flat AD space with respect to route computation, it is desirable to avoid global distribution of routing updates. In other words, the configured topology and policy database may be distributed globally, but dynamic status information may be distributed on a selective basis. One approach is to use a hop count to limit the propagation of flooded updates. This would be ef-

¹¹This approach was suggested by Dave Oran of Digital Equipment Corporation at a meeting to discuss alternatives for inter-AD routing in the Internet, M.I.T., June 1990.

fective in keeping those ADs that are nearby (in terms of hops) up to date.

However, AD communication is unlikely to exhibit strict geographic (or internet hop based) locality. Therefore, ADs might want more up to date status information about ADs that are many hops away simply because they use those transit ADs frequently.¹² In this case it is worth exploring community mechanisms that support efficient distribution of updates to logical communities. It might be possible to exploit multicast service to support such a mechanism. This is another subject for further study.

Finally, the model makes clear that the overhead of this approach is influenced significantly by the type of policies that ADs express. Therefore, it is essential that we develop tools for network managers to use in creating and evaluating the appropriateness and overhead associated with their policies.

5 Acknowledgments

We are very grateful to Lee Breslau and Martha Steenstrup for insightful comments on a previous draft. We are also grateful to our co-designers and developers of the IDPR protocol architecture.

References

- [1] ANSI, *Intermediate System to Intermediate System Inter-domain Routing Information Exchange Protocol*, Document Number X3S3.3/90-132, June 1990.
- [2] L. Breslau and D. Estrin, *Design of Inter-Administrative Domain Routing Protocols*, ACM SIGCOMM '90 Symposium, Philadelphia, PA, September 1990.

¹²One of the underlying assumptions of this architecture is that, in general, the rate of change in VG status will be significantly slower than the maximum propagation delay for the updates; so we are not concerned about the validity of the updates upon arrival.

- [3] D. Clark, *Policy Routing in Internet Protocols*, **RFC 1102**, SRI Network Information Center, May 1989.
- [4] D. Estrin, *Policy Requirements for Inter Administrative Domain Routing*, **RFC 1125**, SRI Network Information Center, November 1989.
- [5] European Computer Manufacturers Association, *Inter-Domain Intermediate Systems Routing*, **Technical Report ECMA/TC32-TG10/89/56**, May 1989.
- [6] S. Hares, D. Katz, *Administrative Domains and Routing Domains, a Model for Routing in the Internet*, **RFC 1136**, SRI Network Information Center, December 1989.
- [7] ISO, *OSI Routing Framework*, **ISO/TF 9575**, 1989.
- [8] J. M. Jaffe, *Hierarchical Clustering with Topology Databases*, **Computer Networks and ISDN Systems**, 1988, pp 329-339.
- [9] M. Lepp and M. Steenstrup. *An Architecture for Inter-Domain Policy Routing DRAFT RFC*, January, 1990.
- [10] M. Little, *Goals and Functional Requirements for Inter-Autonomous System Routing*, **RFC 1126**, SRI Network Information Center, October 1989.
- [11] K. Lougheed and Y. Rekhter, *Border Gateway Protocol*, **RFC 1163**, SRI Network Information Center, June 1989.
- [12] IBM, *A Comparison of Various Aspects of ECMA and BRP*, **X3S3.3/90-84**, April 1990.
- [13] N. F. Maxemchuck and M. E. Zarki, *Routing and Flow Control in High Speed, Wide Area Networks*, **Proceedings of the IEEE**, January 1990, vol.78 no.1, pp 204-221.
- [14] Open Routing Working Group, *Inter-Domain Policy Routing Protocol Specification and Usage: Version 1*, **DRAFT RFC**, April 1990.
- [15] E. Rosen, *Exterior Gateway Protocol (EGP)*, **RFC 827**, SRI Network Information Center, October 1982.
- [16] J. Seeger and A. Khanna, *Reducing Routing Overhead in a Growing DDN*, **IEEE MIL-COM '86**, Monterey, CA, October 1986, vol.1, pp 15.3.1-15.3.13.

3D.2.14.