

# A Protocol for Route Establishment and Packet Forwarding Across Multidomain Internets

Deborah Estrin, *Member, IEEE*, Martha Steenstrup, *Member, IEEE*, and Gene Tsudik, *Member, IEEE*

**Abstract**—Internetworks that are global in scale, contain multiple administrative domains, and support a range of services present special requirements for routing. Multiple administrative domains introduce the need for policy-sensitive routing. Service heterogeneity intensifies the requirement for type of service (TOS) routing, as well as other protocol support for handling a range of services, from datagrams to multimedia streams.

This paper summarizes the key concepts and protocols developed as part of the Interdomain Policy Routing (IDPR) architecture. We place particular emphasis on the route installation and packet forwarding mechanisms because they are critical to protocol performance and differ significantly from current practice in datagram wide area networks.

## I. INTRODUCTION

RESEARCH in computer network protocols currently faces two major challenges: high speed and large scale. This paper addresses the combined problem of routing in a very large *global* internetwork that supports a range of services. Three requirements motivate our activities: 1) coexistence of many diverse administrative domains within a single internetwork; 2) support of heterogeneous network resources and services; and 3) very large numbers of constituent networks.

This paper summarizes the key concepts and protocols developed to support *Interdomain Policy Routing* (IDPR) in the future global internetwork with emphasis on the design and implementation of the route installation and packet forwarding protocols. Section II provides background on the problems of interdomain routing, summarizes other work in the area, and defines the basic concepts and terminology used in the IDPR architecture and throughout this paper. Sections III and IV summarize the function of IDPR's four constituent protocols and security architecture. Section III-C provides a more detailed description of route installation and packet forwarding mechanisms and presents results from our experimental prototype. The last section of the paper, Section VI,

Manuscript received July 1991; revised January 1992; recommended for transfer from the IEEE TRANSACTIONS ON COMMUNICATIONS by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Jonathan Smith. The work of D. Estrin and G. Tsudik was supported by the National Science Foundation under contract NCR-9 011 279, with matching funds from AT&T Bell Laboratories and GTE Laboratories. This work was conducted while both D. Estrin and G. Tsudik were at the University of Southern California. The work of M. Steenstrup was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract MDA903-91-D-0019

D. Estrin is with the University of Southern California. (email: estrin@usc.edu)

M. Steenstrup is with Bolt, Beranek and Newman, Inc., Cambridge, MA. (email: msteenst@bbn.com)

G. Tsudik is with IBM Zurich Research Laboratory, Zurich, Switzerland. (email: tsudik@ibm.com)

IEEE Log Number 9206190.

discusses open issues related to route installation and packet forwarding.

For a more detailed discussion, we refer the reader to [23],[39],[2]. For more background on interdomain routing requirements and alternative architectures, see [6],[11],[1],[24].

## II. INTERDOMAIN ROUTING

Network routing has received a lot of attention since the late 1950's as evidenced by the enormous amount of literature in the field. Several fundamental routing algorithms were developed, most notably Dijkstra's Shortest Path [10] and Ford and Fulkerson's Max Flow algorithms [17]. The former gave rise to a family of routing protocols known as *link state*, and the latter to a collection of protocols known as *distance vector*. Link state protocols are characterized by each node keeping a map of the entire network over which it computes shortest paths to all destinations. Each node contributes to this map by flooding the network with a link state packet, i.e., a packet that contains the status of all incident links. In distance vector protocols, nodes keep tables of the best paths and associated metrics for all possible destinations and periodically exchange the contents of this table with neighbors.

The DARPA internet evolved into the first large decentralized and dynamic datagram network. At first, its routing protocol was of a distance vector variety, as described in [26]. However, as the network grew, shortcomings of the distance vector protocol became more apparent. Frequent oscillations and otherwise unstable behavior that it exhibited were due mostly to long propagation delays with respect to changes in topology. The successor [25], was a link state protocol with a relatively short convergence period and looping avoidance. Neither protocol incorporated any notion of security or policy.

As the internet grew, it began to encompass a greater number of autonomous networks or administrative domains (AD's). An AD is a collection of links, routers, and end systems governed by a single administrative authority [18]. Fig. 1 illustrates a sample internet of connected AD's.

With regard to routing, the growing number of AD's presented three problems.

- **Autonomy:** While electing to become part of the Internet, AD's do not necessarily wish to expose their internal network structure to the rest of the world. In other words, there is a need to limit the dissemination of routing information.
- **Scale:** The size of the internet makes the deployment of a global routing protocol undesirable. Since routing

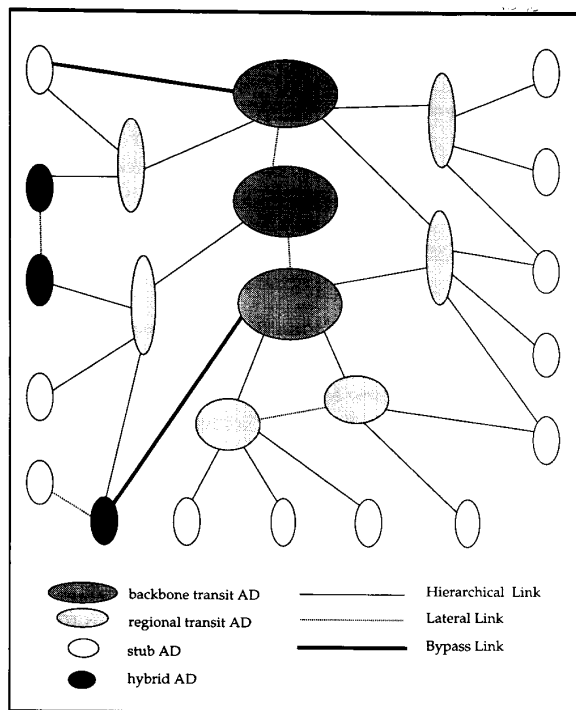


Fig. 1. Example of AD interconnection.

information is typically propagated throughout the entire domain of a routing protocol, routing tables in participating gateways grow in proportion to the size of the Internet. Therefore, in order to avoid an *information explosion*, routing information granularity must be coarser than end systems or networks.

- **Heterogeneity:** New types of service, diverse administrative interests, diverse user communities, and a wide range of underlying network technologies makes global internets inherently heterogeneous. As a result, not every user of the network has the same criteria for selecting one route over another. Routing must provide new flexibility to provide diverse services to diverse users over diverse networks.

Because of these issues, we consider two types of routing protocols.

- **Interior Gateway Protocols (IGP's)** are intended for use within a single administrative entity, i.e., AD. Routers employing an IGP exchange reachability information pertaining to entities within an AD [29].
- **Exterior Routing Protocols (EGP's)** are used by AD border routers to learn about reachability of networks in other AD's [29]. (We discuss EGP's in the following subsections).

Some of the notable IGP's are RIP [27], IGRP [19], OSPF [28], and DEC IS-IS [9].

These IGP's are well-suited for their intended application domain, i.e., a single-AD environment. However, they do not scale to a large enough number of networks. For example, although OSPF groups networks into areas, each and every

area must connect to a common root; interdomain routing must allow more flexible topologies. Existing IGP's do not support large enough numbers of TOS's or routing criteria either, and require route computation to be repeated for each TOS supported [2]. Moreover, the predominant forwarding mechanism used in conjunction with existing IGP's is hop-by-hop routing, in which each router that receives a given message makes an independent forwarding decision. In general, hop-by-hop routing alone does not support special routes efficiently on a global scale (see Section II-E).

#### A. Exterior Gateway Protocol

As the Internet incorporated a more diverse organizational mix, coexistence of multiple administrations (and the associated security implications) was recognized as an important problem. The Exterior Gateway Protocol (*EGP*) [38] was the first routing protocol to address this issue.<sup>1</sup>

*EGP* was designed to communicate reachability information among administrative regions that do not enjoy mutual trust. It includes an authentication facility for validating routing information exchanged among the regions. The regions hooked together by *EGP* can be viewed as AD's.

*EGP* supports a very limited notion of policy. Individual AD's are allowed to hide portions of their routing database that they are not willing to share. Also, AD's are free to manipulate route metrics that they assign to other AD's in order to favor or preclude certain transit AD hops. However, *EGP* does not provide for TOS-based or other fine-grained policy enforcement. In *EGP*, an AD makes routing decisions based only on its own policy since *EGP* provides no facility for the distribution of policy information across AD boundaries.

Furthermore, in order to avoid routing loops, *EGP* imposes a topological restriction on AD interconnection in the form of a cycle-free hierarchy. As Clark points out in [6], *EGP*'s restriction on the interconnection topology has proved unsatisfactory. In general, topological restrictions are undesirable as they inhibit autonomy and are nearly impossible to enforce [2],[11].

#### B. Border Gateway Protocol

BGP is a recently proposed addition to the Internet Protocol family [24]. It was designed to be a successor to *EGP*, and a variant has been submitted as an international standard [1]. Its foremost goal is to provide efficient and robust inter-AD routing with rapid convergence and loop detection for arbitrary internetwork topologies.<sup>2</sup> It is primarily intended for use by transit AD's and interoperates with other interior routing protocols.

BGP is designed under the following assumptions: Policies can be expressed using information about the full AD path that packets will travel to a destination, and transit policies apply uniformly to all sources.

<sup>1</sup> In this section, the term *EGP* denotes a specific protocol.

<sup>2</sup> BGP and *EGP* use the term *Autonomous System* and *Routing Domain*, respectively. We use the term *Administrative Domain*. They are not completely equivalent but, for the sake of this discussion, they can be interchanged. See [18] for further discussion.

BGP uses hop-by-hop routing and a distance vector algorithm for the next hop selection [26]. One common benefit of traditional distance vector algorithms is the ability to hide network structure. Neighboring nodes exchange reachability information for a specific destination in the form of distance metrics corresponding to each next hop. Nodes do not exchange information about subsequent hops to the destination. BGP augments this traditional approach by distributing full AD-level paths. In other words, for each destination advertised, nodes specify the AD-level path to that destination. As a result, BGP provides less information hiding in return for the ability to detect routing loops quickly. By using full AD paths to detect loops BGP avoids convergence problems [25] without imposing topological restrictions on AD interconnection. In addition, AD path information can be used as policy criteria for route selection.

BGP allows for limited policy-based route selection. A BGP router can select its next hop based on the information provided in the full AD path, in addition to the distance metric. For example,  $AD_A$  can reject all routes through  $AD_B$ . On the other hand, each AD must apply the same route selection decision to all packet sources, including itself. For example,  $AD_A$  cannot reject all routes through  $AD_B$  for itself without affecting its neighbors, and vice versa. Similarly, an AD cannot apply one policy to one neighbor and a second policy to another neighbor. Since BGP was not intended to implement policies that discriminate between traffic end points with arbitrary granularity, the approach achieves its goals [24].

Each BGP router can be configured according to its AD's local policy. Even though local policy is not distributed among AD's, it is represented in a universal *policy language*. A policy in this language is an expression

$$[Network\text{-}list, AD\text{-}path]=preference.$$

The semantics of a policy are as follows: If a routing update for a network in *Network-list* is received via *AD-path* and its *preference* metric is better than that of a path currently in use, then this update must be used for subsequent routing.

### C. Interdomain Routing Protocol

Interdomain Routing Protocol (IDRP)[1] is an extension of BGP that has been proposed as an international standard. IDRP augments the BGP protocol by including (among other features) distribution lists along with route information [1]. The list may be inclusive or exclusive and is propagated along with next hop and full-AD path information. Each border router along a path may further restrict a distribution list before advertising a route, i.e., AD's may be deleted from the inclusive list or added to the exclusive list but no router can relax or ignore the list.<sup>3</sup> This feature allows IDRP to support some source-specific policies only to the extent that the routing policy is consistent with a single partial ordering of the participating domains. In addition, IDRP has no built-in support for enforcing source-specific policies at packet forwarding time. Another departure from BGP is IDRP's

<sup>3</sup>The proposed standard includes several other extensions which are not directly relevant to our discussion.

ability to include policy-related (e.g., TOS or User Class) information in routing updates.

IDRP is, thus, able to support a wider range of policies than BGP. Nevertheless, because IDRP is a hop-by-hop protocol, it only allows a single route per every  $[destination, TOS]$  to be advertised. However, multiple routes for some  $[destination, TOS]$  combination may be necessary in order to allow traffic sources to apply route selection policies. (See [2] for an in-depth discussion of this and other related issues).

### D. Routing with Multiple Hierarchical Addresses

A novel approach to policy routing is the use of multiple hierarchical addresses (MHA). In [42], Tsuchiya suggests that multiple addresses be assigned to end systems (stub AD's, in our parlance). A single address is formed as  $[stub.regional.backbone]$  indicating that the corresponding route:  $[backbone \Rightarrow regional \Rightarrow stub]$  satisfies the policies of its component AD's. A given end point may have a number of such addresses differing in the *regional* and/or *backbone* fields.

Routing in this approach can be viewed as a variant of source routing. More specifically, a route between  $AD_A$  and  $AD_B$  is the combination of  $AD_A$ 's address and the *inverse* of  $AD_B$ 's address, e.g.,  $AD_A.regional_1.backbone_1$  followed by  $backbone_2.regional_2.AD_B$ . To route a packet, a stub AD simply selects (according to its policy) one of the addresses for the intended destination.

The main benefit of MHA is its simplicity and low overhead with regard to route computation. There are a few important drawbacks, though. First, a *shallow* (three-level) hierarchy is assumed. As pointed out above, such restriction is undesirable for two reasons: i) lateral and bypass links must be supported as the Internet is not expected to conform to a strict hierarchy, and ii) even if a strict hierarchy is possible, limiting it to three levels may be inadequate in the context of a global Internet. A related problem is the assumption regarding backbones. If the backbone components of a route are not identical, multiple transit backbones have to be traversed. Such backbones, however, are not included in either of the two addresses. Consequently, policy enforcement is severely limited from the perspective of the end points (since transit backbones are *hidden*) from them. Conversely, transit backbones not included in the route must enforce their policy on a per-packet basis, i.e., they are denied any opportunity of restricting traffic in advance of actual communication.

### E. IDPR Architecture

*Interdomain Policy Routing* (IDPR) was designed as a policy-routing architecture that accommodates transit service provider policies as well as source-specific policies. Participating AD's express their policies and advertise them throughout the internetwork. Source AD's specify policy routes for data packet forwarding. Source routing combined with policy advertised in routing updates allows transit service providers to apply source-specific policies while permitting stub networks to select routes based on local criteria.

More traditional hop-by-hop routing protocols do not efficiently support the wide range of policies and types of service (TOS) that is anticipated in a global multidomain internetwork. For example, the support of source-specific policies in a hop-by-hop routing protocol would require computation and storage of routing tables for each possible source. Even though it is possible to avoid storage costs by computing source-specific routing tables on demand (as in [8]), the cost of on-demand (i.e., real-time) computing at every AD hop would result in very high setup delays.

Fig. 2 illustrates a simple example of policies that can not be supported with hop-by-hop routing without replicating routing tables. In this example, *C* can use either *B* or *E* to get to *A*. However, *D* prefers to use *B*, and *F* prefers to use *E*. In hop-by-hop routing, *C* must make a *single* decision (assuming a single routing table). Consequently, *C*'s decision will deny a route to either *D* or *F*. Examples of policies that could lead to this situation are:

- *B* provides enhanced type of service support but charges on a usage-sensitive basis while *E* does not charge but makes no service guarantees.
- *D* is a university, *F* is a private corporation, and *B* is a subsidized network for university use only.

One possible solution to this problem is to compute special routes on the fly in the manner of incremental SPF computation proposed by Chiappa [4] or link state algorithms for IP [32] multicast proposed by Deering [8]. Early in the design stages, IDPR designers rejected this alternative because it requires that the input to the routing decision (i.e., the route selection preferences of sources and the route usage restrictions of transits) be distributed globally and consistently in order to compute loop-free routes on a hop-by-hop basis.<sup>4</sup> For this reason, the IDPR architecture is based on source routing (see [3] for further discussion and justification).

With hop-by-hop routing, the routing databases must be consistent among all routers in order to ensure loop-free message forwarding. With source routing, the source completely determines the forwarding of its messages and hence guarantees loop-free routing, even in the presence of inconsistent routing databases. In a large internet, maintaining complete routing databases at every router may be neither practical nor desirable. Source routing can work well in such an environment, because it has the advantage of relaxing the consistency requirements for routing databases across the global internet [2],[3].

IDPR uses a link-state style routing protocol, along with source routing and explicit advertisement of policy and topology information. Here, a *link* is actually an administrative domain which may include a set of physical links, networks, and routers. The *state* of a link consists of the domain transit policies and the current connectivity to adjacent domains. Source routes specify the AD's that compose a route; the particular links and routers transited within AD's are transparent to the source and may change during the lifetime

<sup>4</sup>It is possible to reduce this to global distribution of transit restrictions only if sources explicitly encoded their route selection criteria in route request packets. However, this requires much more standardization of source criteria, as well as divulging of that information.

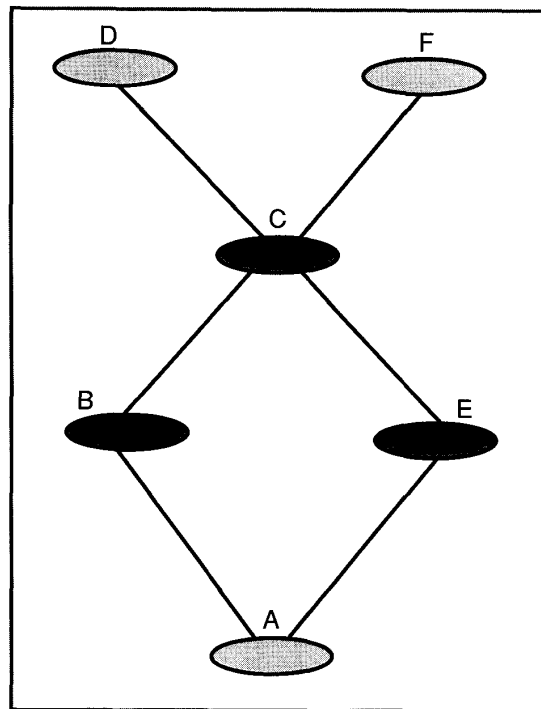


Fig. 2. Example of policies that can not be supported with hop by hop routing without replicating routing tables.

of a route. Source routes are computed based on topology and policy information advertised by each participating transit AD (as described in Section II-E below). Policy information specifies what types of traffic the particular AD will carry; for example, an AD can limit the types of service made available to particular endpoint AD's. Configured link-state information (i.e., configured AD topology and policy information) is advertised by flooding to *all* ADs; dynamic information may be flooded or distributed in a more limited fashion.

In order for hosts to communicate across AD boundaries, a source route must be computed by an entity in the source AD. This route is then installed from the source AD to the destination AD. Successive data packets between the two AD's, that meet policy and type of service requirements specified at setup time, do not carry the full route. Instead, they carry a route identifier that maps to route-related state information (e.g., next hop). The state information is used to facilitate authorization and accounting of data packets, in addition to its more obvious role of reducing data packet header length. In effect, the IDPR architecture installs routes *on demand* in routers through this setup function.

A policy route resembles a loose source route in two ways. First, only certain routers at the borders of domains are bound at route setup time. The intradomain routers are not bound before actual data packet transfer. Moreover, intradomain dynamic routing is preserved, thus allowing successive packets belonging to the same policy route to traverse different intradomain routers. Adaptive routing across AD's increases the availability of a particular AD's resources in the face of dynamic, intradomain, node, and link status.

Each packet traveling along a policy route, between successive border routers, is encapsulated with a local header at the domain entry border router, according to the network protocol of that domain. For example, across IP networks, IDPR packets are encapsulated within IP [32] headers; the IP source address is set to that of the sending border router and the destination address to that of the next-hop border router. However, the encapsulating protocols can vary on an AD-by-AD basis, so long as it is agreed upon by neighboring border routers. The encapsulating local header is later removed by the domain exit border router. Local encapsulation of packets at each domain means that a policy route may include domains with diverse network protocols, without requiring translation and manipulation of packet headers at each domain. For more information on header encapsulation for network protocols, see [41].

The setup protocol installs routes on demand. This makes it well-suited for an environment where there are a large number of possible policies and associated policy routes. Such an environment makes hop-by-hop route table replication impractical and requires more global coordination. Unlike with some connection-oriented protocols, these routes can be shared among multiple transport sessions, and even among different host pairs, to avoid the latency of setup for transport sessions that exhibit locality. Moreover, expedited data can be included in setup packets to accommodate datagram traffic that is traveling to uncommon destinations and that is not frequent enough to warrant maintaining state. Setup always establishes a bidirectional route for control traffic and, usually (although optionally), a bidirectional route for data as well.

Both setup and packet forwarding protocols are described in greater detail in [44]. There is much more to the IDPR architecture than can be covered in a single paper. In particular, route computation and information distribution mechanisms are the most critical architecture components not discussed here. The architecture can work in today's networks with very simple approaches to these latter problems; therefore, we assume as much here and focus on the setup and packet forwarding components which would work with a range of computation and distribution strategies. Future versions of the protocol will employ more sophisticated computation and distribution mechanisms in order to scale efficiently to orders of magnitude larger networks of the future. For example, the IDPR style of routing may be used in conjunction with hop-by-hop forwarding of generic packets over generic routes in order to take advantage of the aggregation of routing and forwarding information supported in hop-by-hop routing [13].

The following terms are used throughout the remainder of this paper.

- **Transit and Stub AD:** A transit AD provides transit services (i.e., bandwidth and switching) to other AD's. Transit AD's must advertise their policies to allow transit service customers (stub AD's) to make appropriate routing decisions. A stub AD contains end systems that serve as the end points of communication; stub AD's do not provide transit services and need not advertise policy, TOS, and connectivity even if they are multihomed. Stub AD's may distribute policy information to other stub

AD's on demand (e.g., telling a source the conditions under which the source can send packets to hosts within the target AD).

- **Policy Gateway (PG):** A router at the border of an AD that implements the IDPR protocol suite. Transit policy is enforced by PG's, i.e., every PG takes part in policy route setup and packet forwarding. Every PG is uniquely identified by the combination of: i) its AD identifier, and ii) its PG number. (The latter is unique within the AD).
- **Virtual Gateway (VG):** A set of physical policy gateways that form a connection between two adjacent AD's. Every VG has two *sides*, one in each AD that it connects. The simplest VG consists of only two PG's directly connected by an inter-AD link. This abstraction reduces the amount of detailed (i.e., PG-specific) status information that must be distributed and maintained when there is redundant connectivity between two adjacent AD's. For example, two distinct physical links between the same AD pair (or rather the network interfaces that are the end points of these links) may be collapsed into a single VG. A VG is denoted by a triple:  $[AD_1, AD_2, VG_{num}]$  where  $AD_1$  and  $AD_2$  are the AD identifiers and  $VG_{num}$  is the VG identifier.  $VG_{num}$  is needed in cases when there are multiple VG's between the same pair of AD's (e.g., supporting different TOS's). Multiple virtual gateways may be configured between two adjacent AD's in order to meet performance and reliability requirements. As an example, consider the situation where two adjacent AD's are doubly connected by a special high-speed link and a regular low-speed link. The two links provide different types of service even though both may terminate in the same pair of physical PG's. For example, the access rights for the two links may be quite different, e.g., the use of high-speed link may be restricted. Also, the charging rates may differ as well. In this situation, the logical division into two VG's is justified but it is not made imperative by the IDPR architecture.<sup>5</sup>
- **Policy Term (PT):** A unit of policy information. It embodies conditions specified by an AD for use of its resources. PT's may express restrictions in terms of end point (stub) AD's, types of service, or previous and next-hop AD via which resources may be accessed. For further discussion of policy types, see [11]. Policy terms are included in routing updates.
- **Policy Route (PR):** An ordered sequence of AD segments (see Fig. 3). Each segment carries an identifier of an intervening AD and a VG number. The VG number specifies which VG should be used to reach the next-hop AD contained in the next AD segment. Each segment also includes one or more PT's that justify the use of the transit AD's facilities by the source AD.

In the sections to follow, we describe the protocols developed and implemented to support IDPR. We concentrate primarily on route installation and packet forwarding.

<sup>5</sup>Another potential justification for multiple parallel VG's is in the case when one set of inter-AD links is used to carry direct  $AD_1 \leftrightarrow AD_2$  traffic while a different set of links is used for transit traffic.

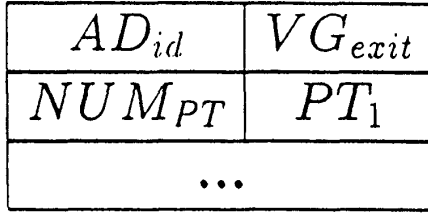


Fig. 3. AD segment in a PR.

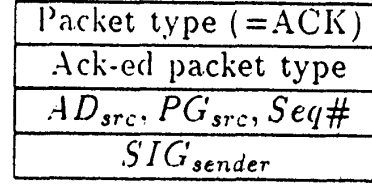


Fig. 5. ACK packet.

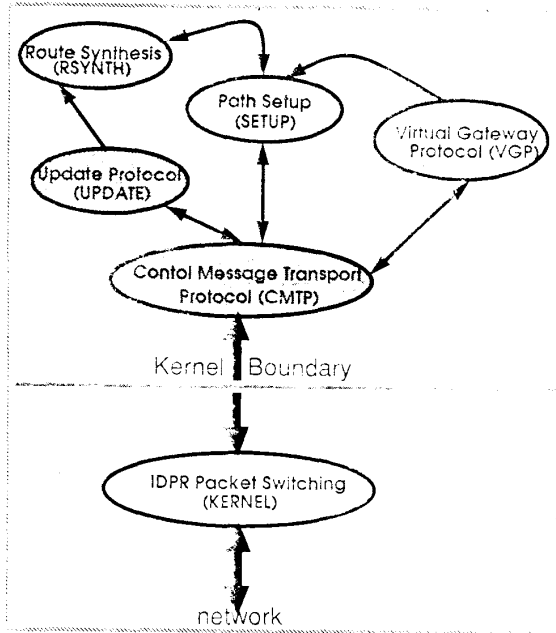


Fig. 4. Interplay among IDPR components.

### III. PROTOCOLS

Four protocols form the core of the IDPR architecture.

- **Virtual Gateway Protocol (VGP)** performs *neighbor polling*, typical of link state protocols.
- **Update Distribution (Update)** handles reliable flooding of link state updates throughout the internetwork.
- **Route Setup (Setup)** is responsible for installing and maintaining PR's in intervening AD's.
- **Packet Forwarding (Kernel)** performs switching of IDPR data packets along previously established PR's.

A fifth component, **Route Computation**, computes PR's in accordance with the local policies. While it is critical to the architecture, it is not a protocol per se; each AD can implement its own version of route computation so long as it interfaces to the other protocols. For example, Update provides the inputs to route computation and route computation provides policy routes to setup. Another component, **Control Message Transfer Protocol (CMTP)**, implements packet transfer functions common to all IDPR components.

The interaction among these IDPR protocols is illustrated in Fig. 4.

#### A. Packet Types

IDPR distinguishes between two types of packets: control and data. Control packets are used for route setup and maintenance (Setup), dissemination of link-state PT updates (Update), and PG status messages (VGP). Data packets are used to transport user data over established policy routes.

Control and data packets differ in the manner they are processed by PG's. However, both IDPR control and data packets require local encapsulation when crossing domain boundaries. Control packets are transmitted reliably between adjacent PG's in a policy route. If a control packet is not acknowledged within a predetermined interval, the sender retransmits. After attempting  $n$  ( $n$  is a locally defined value) successive retransmissions, the sender PG *gives up* and informs the control packet's source of the failure to deliver. All control packets are identified by the combination of packet type and packet handle. A handle is a unique sequence number assigned by the packet's source. This combination is *guaranteed* to be unique, i.e., no two control packets of the same type carry the same handle between the same two PG's.

A PG receiving a control packet subjects it to rigorous integrity and authenticity checks before acting on its contents. Control packets contain routing-related information. If this information were corrupted and then acted upon, it could adversely affect a PG's ability to generate and establish feasible policy routes. Every control packet may be protected by a digital signature computed with a secret key of its originator. This allows all interested parties to authenticate both the origin and the contents of a control packets. Refer to Section IV for more information about security in interdomain routing.

After receiving and authenticating a control packet, a PG replies to the previous-hop PG with an ACK packet containing the identifier of the control packet being acknowledged. An ACK is used only between adjacent PG's. It may be protected by a signature of its sender and contains the packet type and a handle to associate the ACK with the protocol and the policy route. The recipient of the ACK can unambiguously associate the ACK handle with the original packet. The handle syntax is defined as part of the transport protocol used between adjacent PG's (see [44]). The handle in our implementation is made up of the source AD identifier, the source entity (i.e., PG) identifier, and a 32-bit sequence number assigned by the transport protocol.

Unlike control packets, data packets are not acknowledged at every PG hop. Their transmission is inherently unreliable, since a PR may traverse a number of unreliable datagram subnets. One similarity between control and data packets

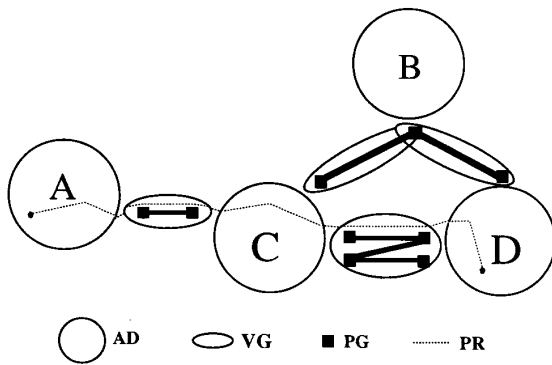


Fig. 6. Depiction of several interconnected AD's.

is that both carry handles that are globally unique. A data packet handle is a PR identifier that allows transit PG's to associate data packets with PR table entries and forward packets according to the information stored therein (i.e., the next-hop PG). Like the control packet handle, the PR identifier is made up of the source AD identifier, the source entity (i.e., PG) identifier, and a 32-bit sequence number assigned by the Setup protocol.

### B. Virtual Gateway Protocol

The Virtual Gateway Protocol (VGP) is the protocol that allows groups of PG's to appear to the Update protocol and Route Computation procedure as single entities. A VG is comprised of at least two PG's, one in each of two directly connected AD's. There may be more than one PG on either side of the VG, where a side refers to each AD. For example, in Fig. 6,  $VG_{AC}$  is comprised of two PG's, whereas  $VG_{CD}$  includes 4 PG's. A particular policy route is a sequence of VG's. However, at any point in time, the PR passes through only one PG on each side of the VG, as indicated by the dotted line from AD A to AD D in Fig. 6. We refer to two PG's as *directly connected* if they are part of the same VG, belong to different AD's, and are able to exchange packets without traversing other PG's.

VGP information has both global and local significance. The update protocol (described in Section III-D) includes VG reachability information garnered from VGP in dynamic routing updates; in this sense, VGP carries globally relevant information. At the same time, policy gateways within an AD use information collected from VGP to make local forwarding decisions across a VG or across an AD. This detailed forwarding information is only distributed within the AD or VG and not to the whole internet. In addition to the VGP-derived information, each PG uses a simple up/down protocol to monitor reachability of directly connected PG's within a VG. PG's use a similar up/down protocol to monitor the reachability of other PG's and VG's that belong to their same domain. VGP packets are encapsulated in the internet protocol used among PG's. (Other methods of communicating interdomain reachability information within the domain are discussed in [34].)

In summary, the information obtained from VGP is used by each PG to carry out the Setup and Update protocols, as described in the rest of this section.<sup>6</sup>

### C. Route Setup and Packet Forwarding

The route setup protocol (Setup) establishes a logical connection along the sequence of PG's that initiate a PR. Consequently, when an outbound packet arrives at a PG (i.e., the packet is exiting the AD to which the PG belongs), the source PG looks to see if the packet is part of a source/destination association that has already been bound to a policy route. In order for the association to match, it must have compatible type of service, user class identifiers, and other global conditions in addition to the same source and destination addresses. If the association is bound, then it has already been assigned a unique PR identifier by the source PG. In this case, the packet is encapsulated in an IDPR data packet header containing the PR identifier, and the packet is forwarded to the next PG on the already established policy route. If the packet is not part of a currently mapped association, the source PG compares the policy conditions of the packet with active policy routes. If they match, then the association is added to the table and the packet is encapsulated as described above. In other words, many transport sessions between different host pairs may share a policy route, provided they have the same service requirements and travel between the same source and destination domains, and consequently will use the same PR identifier. Finally, if no active policy route applies, the PG invokes Setup.

Setup generates a SETUP packet which travels, hop-by-hop, along a sequence of PG's that correspond to the policy route. The policy route is included within the control packet so that each PG can make the correct next forwarding decision. The policy route format is depicted in Fig. 7.

When a PG receives a SETUP packet, it checks the listed policy conditions, verifies the data integrity, and authenticates the packet origin. Data integrity and origin authentication are necessary to prevent fraudulent routes from consuming PG resources (i.e., both table space and bandwidth).<sup>7</sup>

If all of these checks pass, the necessary PR state information is installed (i.e., PR ID, next hop) and the SETUP packet is forwarded to the next PG in the PR.

Through the use of VGP (described above), each PG is kept informed of reachable PG's within its VG and of other PG's that represent the AD's other VG's. When a PG receives an *outgoing* SETUP packet (where outgoing means that it is exiting the AD to which the PG belongs), it must decide to which PG on the other side of the VG to forward the packet. When a PG receives an *incoming* SETUP packet, it must select a PG in the next VG specified in the route. In the first case, the next PG is in a directly connected AD; in the second case, the next PG is in the same AD as the sender.

<sup>6</sup>In the current implementation, we support two PG's per VG and therefore have actual experience only with the simplest behavior of this protocol.

<sup>7</sup>The particular signature scheme used for data integrity/origin authentication is indicated in the setup packet. Section IV provides more details on the security mechanisms provided. See also [39],[16].

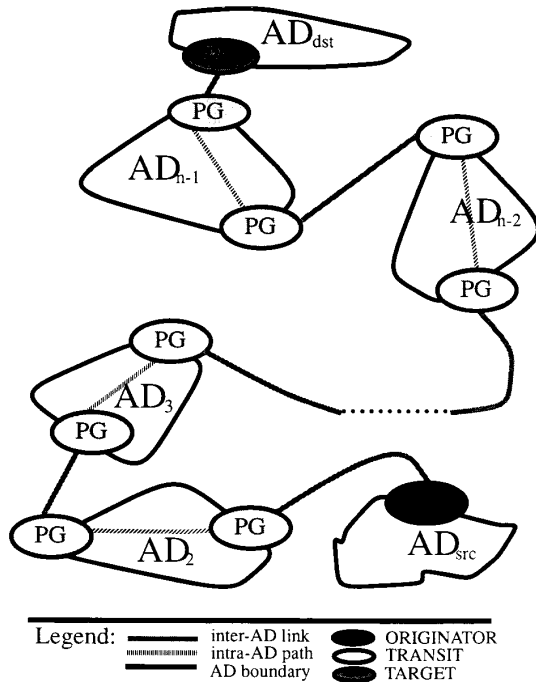


Fig. 7. PR traversing a sequence of AD's.

The source PG expects to receive an indication of whether the route was set up successfully. It may retry the route setup if no response is received, or it may try to set up an alternate route if a negative response is received.

#### D. Update

The Update protocol is a reliable link-state update protocol similar to those described in [25],[31],[28]. As with Setup packets, routing updates are sent from PG to adjacent-PG in a reliable manner.<sup>8</sup> UPDATE packets are encapsulated in the internet protocol used between PG's (e.g., within AD's).

The UPDATE packets include sequence numbers which allow duplicate detection. If the topology includes cycles, a PG may receive an UPDATE more than once. The duplicate UPDATE is detected and dropped so that duplicates are not flooded further. Routing updates are not modified en route. Each UPDATE includes a signature (computed and appended by the originating AD and checked by the recipient Update process within each AD) to provide integrity and authenticity.

Routing updates are generated by a representative PG in each transit AD (i.e., the lowest-numbered live and reachable PG in the AD).<sup>9</sup> The ultimate recipients of this information

<sup>8</sup>However, in this case, retransmission of unacknowledged update packets may be overridden by availability of a new UPDATE packet that obsoletes the retransmitted information. The current implementation does not implement this feature.

<sup>9</sup>In the absence of intra-AD partitions, there is one representative PG per AD. If a partition occurs, then two PG's on either side will assume the PG representative role and receiving update processes will effectively treat them as separate AD's until the partition is healed. UPDATE REQUEST and RESPONSE messages are used to reconstitute a connectivity database after a partition. In the future, additional mechanisms for healing partitions may be included.

are the route servers in stub and hybrid AD's (i.e., any AD that must generate policy routes for traffic generated by end systems within the AD).

There are two types of UPDATE messages. Configured UPDATE's represent the policy, TOS capabilities, and topology of the network when all gateways and links are functioning properly. Dynamic UPDATE's represent changes in the status of configured network elements, i.e., nodes going up or down. This distinction is important because of the very large number of nodes in the network. The configured topology database must be distributed globally; however, it will change relatively slowly. In contrast, dynamic UPDATE's will be generated more frequently, by definition, but do not necessarily require global distribution. Even dynamic UPDATE's are expected to be low-frequency events (e.g., relative to intradomain IGP routing). For example, when an intradomain link goes down, we expect the AD's IGP to find an alternative path between any two PG's, thereby making the change transparent to IDPR. Dynamic UPDATE's may be flooded globally or distributed to a limited group (e.g., based on a hop count or using reverse path update to active sources [3]).

If a PR was computed with correct configured information but without the benefit of up-to-date dynamic information, the Setup (or data packet forwarding) process will fail with an appropriate error message alerting the originator of the SETUP but no looping will occur because of the use of loop-free source routes.

UPDATE packets express policy and topology information. The information in an UPDATE message can be thought of as a listing of the policy conditions that are supported between each pair of the AD's VG's. However, we assume that many policies will be applied uniformly to a large subset of VG's in an AD. Therefore, each entry in an UPDATE is represented as a set of policy conditions that apply to the set of VG's listed. The VG's listed indicate connectivity to directly connected neighbors so all VG's listed in an UPDATE refer to VG's that connect to the AD generating the UPDATE. (This is analogous to a traditional link-state update that advertises information about a node's neighbors.) The specific format and forwarding of IDPR UPDATE messages can be found in [39].

The information needed to generate an update packet is collected from VGP and the configuration server. VGP and IGP information allows the representative PG to determine which VG's are up and reachable. The configuration server provides the PG with configured policy and topology information. The PG modifies the configured update to indicate which VG's are unreachable or down.

#### E. Route Computation

Strictly speaking, Route Computation is not a protocol; it is the function that takes input from the Update protocol in the form of global connectivity/policy database and computes AD-level source routes. Route Computation may be invoked on demand, i.e., when the first outbound packet in a transport session (for which there are no usable active PR's) arrives at the source PG. Alternatively, Route Computation may be invoked in advance to precompute some subset of routes that

the AD expects to use (and thereby avoid the latency of on-demand computation). Whatever form of route computation is used, the input is the connectivity database built with information obtained from Update.

We store the information as a collection of nodes, where each node represents either a policy condition or a VG. The output of Route Computation is a set of policy routes and associated conditions of usage. The policy route is a sequence of VG's and the policies that allow the VG to be used in the route. General conditions of use such as type of service, time of day, etc. are appended to the route information itself. This information is stored and provided to Setup upon request.

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

Even though on-demand computation seems expensive, it still offers advantages over hop-by-hop protocols. In hop-by-hop protocols, every transit PG would have to maintain a connectivity map of the entire internetwork. Moreover, route computation would have to be performed at every transit AD hop. In contrast, IDPR does not require transit PG's to maintain any connectivity information beyond that of the neighboring PG's/VG's. Also, route computation takes place only once, at the source AD. When a route serves many sources, it makes sense for an intermediate node to do the computation. However, when routes are different for different sources, then the efficiency of computing the routes inside the network is lost.

It is not computationally feasible to precompute all possible routes to all possible destinations in a large internet with variable policies. Other routing protocols limit the expressible or discoverable routes *a priori* by restricting each routing node to advertise only a single route (or single route per type of service) to each destination. IDPR is more expressive, and allows each stub AD to control which subset of routes it will discover/compute. Each stub AD is free to design or tailor route computation to its needs. In general, we expect Route Computation to precompute some routes. Nevertheless, on-demand computation must always be supported to accommodate requirements not satisfied by precomputation (e.g., uncommon destinations, unusual TOS requirements, or unusual policy conditions). However, the problem with on-demand computation for all routes is that the computation may become time consuming relative to acceptable setup latency limits as network sizes grow. Moreover, using on-demand computation alone is wasteful in terms of potentially unexploited parallelism. Both on-demand and precomputed routes are cached to take advantage of the locality exhibited by most communication environments. However, optimal strategies for managing these caches, in particular invalidation, deserve more careful analysis. An alternative to precomputation may be a hybrid architecture where hop-by-hop routing is used for packets traveling over generic policy routes.

Because of the challenging nature of the route computation

problem, further issues associated with route computation are the subject of ongoing research, as mentioned in Section VI.

#### IV. IDPR SECURITY ARCHITECTURE

One of the key motivating factors for the existence of IDPR is the need to control access to valuable network resources. These resources include end systems, bridges, routers, and links. Access to these resources is what policies are intended to govern. It is, thus, of utmost importance for IDPR (or any other policy routing mechanism) to provide a means for *secure* routing. In this section, we describe the security mechanisms used in IDPR.

Many discussions of *network security* are actually discussions of end system protection in a network environment, e.g., [40],[30],[22],[21],[14],[45]. While this is an important consideration, we have argued elsewhere [15],[16] that it is not adequate in the interdomain context. For both stub and transit AD's, there are valuable network resources that are also the object of policy control.

If control is left to the end systems, valuable stub-AD network resources may be consumed by unauthorized traffic. Rejecting packets at the end system is *too late* from the perspective of network resource usage. Moreover, unrestricted network access increases the vulnerability of AD's to denial of service attacks in the form of packet storms. In other words, the network interfaces of end systems are themselves network resources and should be subject to access control (which can only be achieved below the transport layer). Similarly, some AD's have no relevant end systems (they provide transit services only) and therefore must implement desired controls in the network-layer routing protocol. Since routing is a network layer function, these controls must also involve network-level entities and cannot be left to transport session end points.

Transit AD's are concerned with controlling usage of, and access to, their internal routers and links. Their policies may be based upon the source or destination AD, the previous or next-hop AD, or other characteristics such as user classes, charge codes, or type of service [11]. Transit resources may be billed on a usage-sensitive basis. In addition, service quality is dependent upon adequate capacity to meet demand. Consequently, control of access to these resources is critical to their operation.

Transit resource protection cannot be left to end systems. By the time traffic reaches the end systems, the communication resources would have been used. Moreover, in the case of transit AD's, the destination end system is not within the particular transit AD's administrative control and cannot be expected to recover the transit AD's costs. Furthermore, transit AD's should not rely on stub AD's to enforce transit policies; this represents an excessive compromise of transit ADs' autonomy. Even if transit AD's are open to all paying customers, they need to monitor and charge for traffic. Monitoring and charging are just different types of network-layer control mechanisms. In any case, network layer controls are needed at the boundaries of transit AD's to protect against unauthorized resource usage and denial of service attacks [15].

The IDPR security architecture emphasizes flexibility in order to accommodate diverse needs of autonomous domains. There are two types of flexibility: i) the particular signature mechanism used is specified by the source AD at the time of route setup; and ii) the use of signatures on data packets is optional altogether. The details of the security architecture are presented in [16]. In this paper, we address only those aspects of the security architecture that influence the design of the Setup and Packet Forwarding protocols.

The key elements of IDPR security architecture are<sup>10</sup>:

- All PR Setup control packets are protected by signatures computed with the private key of the originating entity.
- Signatures cover either the entire packet or only the (timestamped) PR header. The choice is made at the time of PR setup.
- All packets (control and data) are timestamped at the origin. No two packets of the same origin and type can bear identical timestamps.
- A transit AD is free to choose on a per-PR basis whether or not to authenticate data packets.
- If the data packet authentication option is used, a secret key is distributed to all intervening PG's as part of PR setup. (This key is subsequently used to compute data packet signatures). This key is enciphered  $N$  times, once for each intervening AD using that AD's public key.
- Data packet signatures can be computed using one of many signature methods. The particular method is negotiated at the time of PR setup.

Public key encryption is used for Setup control packets as well as UPDATE packets since all of them are, in a sense, multicasts (i.e., many potential recipients), thus making conventional cryptography very costly. On the other hand, VGP packets and CMTP ACKS/NACKS can be protected with conventional signatures since they only have significance between neighbors and the number of neighbors a VG has is relatively small.

Even though public key cryptography offers advantages such as data origin authentication and nonrepudiation of origin, its cost remains prohibitively high, especially for bandwidth-intensive communication. For this reason, traditional single-key cryptography is used for computing IDPR data packet signatures.

## V. SETUP AND PACKET FORWARDING PROTOCOLS

Given the preceding overview of the IDPR routing and security architectures, this section outlines the particulars of the PR Setup and Packet Forwarding protocols and provides some results of experimentation with our prototype.

### A. PR Setup Authorization Procedures

The Setup protocol begins with a data packet arriving at a PG in  $AD_{src}$ . When this PG discovers that it has no active PR for the source/destination end system pair in that packet, it asks Route Computation for a new PR to the destination

<sup>10</sup>Only a subset of these functions have been fully implemented and tested in our current prototype; the remainder will be included in future versions.

AD. (Hereafter, the requesting PG is known as the *originator*.) Route Computation replies with the full PR.

Now, the originator composes a new SETUP packet. Included in the packet is the unique timestamp  $TS_{src}$ . Also, the SETUP contains the *Authentication Type* and *Expiration* fields which indicate the authentication method for data packets and PR expiration conditions, respectively.

A SETUP packet is protected by the signature of its originator

$$SIG_{src} = [F_{hash}(P_{setup})]^{DK_{src}} \quad (1)$$

where  $DK_{src}$  is the secret (signature or decryption) key of  $AD_{src}$ .  $F_{hash}$  is a one-way hash function (such as MD4 or MD5 [35],[36]), and a strong encryption function (such as RSA [37]) is used to sign the digest produced by  $F_{hash}$ . The signature is sufficient to maintain the integrity of the SETUP packet. Freshness is attained by  $TS_{src}$  within the SETUP packet.

We expect that most AD's will implement authentication checks during setup only. However, if desired, data packets can also be checked. If data packet authentication is selected at setup time, the originator PG needs to generate a new data signature key,  $K_{dsig}$ , for use in whatever per-packet variation is used.<sup>11</sup>  $K_{dsig}$  must be communicated in secret to each  $AD_i$ . This requires that  $AD_{src}$  encrypt  $K_{dsig}$   $N$  times, i.e., compute  $E(K_{dsig})^{EK_i}$  for all  $AD_i$ . The resulting PR SETUP packet is depicted in Fig. 8.

As the PR SETUP packet propagates along the route, each transit PG in the PR performs the following checks:

1. Validates the timestamp  $TS_{src}$ .
2. Recomputes and verifies  $SIG_{src}$ .
3. If data packet authentication is indicated, obtains  $K_{dsig}$  by decrypting  $E(K_{dsig})^{DK_i}$ .

At this point, if the SETUP packet passes all checks, the transit PG is assured that: i) the PR is valid, i.e., does not violate local policy; ii) the PR is authentic, i.e., issued by a recognized entity; and iii) the PR is *fresh*, i.e., issued recently.

Finally, the PG creates a new table entry for the new PR and computes the next-hop PG using the next-hop AD specified in the next PR segment.

If the SETUP packet fails one of the above checks, the transit (or target) PG in question informs the originator via a REFUSE packet (see Fig. 9). A REFUSE packet contains the PR ID of the SETUP packet, which helps the originator identify the exact SETUP being rejected. A REFUSE packet is protected by a signature of its creator, so that all interested parties can verify that it has been generated by a recognized entity, i.e., by one of the PG's in the route. ( $SIG_i$  in Fig. 9 refers to the signature generated by the  $i$ th hop transit AD). Upon receipt of a REFUSE, all PG's tear down the state information pertaining to the PR mentioned therein and forward the REFUSE to the next PG in the direction of the originator PG.

After a SETUP packet successfully passes through all transit PG's, it finally reaches the last-hop target PG. After validating

<sup>11</sup>Actually,  $K_{dsig}$  is not necessarily a key, per se. It is a secret quantity to be used in data signature computation.

Packet type (=SETUP)	
$AD_{src}$	$TS_{src}$
Authentication-type	
Expiration	
PR	
$E(K_{dsig})^{EK_1}$	
$E(K_{dsig})^{EK_2}$	
...	
$E(K_{dsig})^{EK_N}$	
$SIG_{src}$	

Fig. 8. SETUP packet.

Packet type (=REFUSE)	
$AD_{src}$	$TS_{src}$
$AD_i$	Reason
$SIG_i$	

Fig. 9. REFUSE packet.

Packet type (=ACCEPT)	
$AD_{src}$	$TS_{src}$
$SIG_{dst}$	

Fig. 10. ACCEPT packet.

the PR, the target PG informs the originator of the setup completion by means of an ACCEPT packet (Fig. 10).<sup>12</sup> An ACCEPT packet carries the PR handle, indicating the originating AD and PG identifiers.

As an ACCEPT propagates through each transit PG on its way to the originator, it activates the (hereto dormant) corresponding PR table entry. In other words, an ACCEPT signifies that the PR has been fully authorized by all parties involved. When the ACCEPT reaches the originator PG, data packets can start flowing along the newly established PR. Although expedited data can be included with the SETUP packet and data packets can be sent *before* the ACCEPT is received, there is no guarantee that the intermediate PG's on the route will be ready to forward these data packets.

Failure to receive an ACCEPT within a predetermined period of time causes the originator to generate and resend a new SETUP packet for the same PR. The new SETUP is essentially the same as the original one except that the handle contains a new 32-bit sequence number and the  $TS_{src}$  field reflects the current clock reading. This is done so that

<sup>12</sup>If the particular destination host is not reachable, the target PG informs the source in the ACCEPT packet. It is up to the source whether or not to retain or tear down the route.

all parties involved can distinguish between successive setup attempts for the same route. For example, if the originator times out prematurely and generates a new SETUP while the ACCEPT for the original SETUP is still en route, it will be able to decide unambiguously that the ACCEPT corresponds to the original SETUP packet, not the retransmission.

### B. Packet Forwarding Authorization Procedures

In order to make use of an existing PR, the originator PG must be able to supply information necessary to associate each data packet with a specific PR. This information is placed in the PR header

$$PR_{hdr} = [AD_{src}, PG_{src}, TS_{src}, (TS_{packet}), (DSIG)]. \quad (2)$$

Transit PG's are able to look up the appropriate PR in their tables using the PR handle  $[AD_{src}, PG_{src}, TS_{src}]$  as a lookup key.  $TS_{packet}$  is an optional packet-level timestamp used for detecting old and out-of-order data packets. Finally,  $DSIG$  is the optional packet signature used to protect against data tampering.  $DSIG$ 's verification is subject to the authentication method agreed to upon PR setup. One example of a fast integrity check method is the use of a strong one-way hash function (e.g., MD4/5) in conjunction with a secret prefix or suffix (see [43] for details).

### C. Laboratory Experiments

Experiments with the IDPR implementation were conducted in order to evaluate the overhead incurred by PR setup and IDPR data packet processing. This section summarizes our results.

We define PR setup overhead as the time interval between the arrival of a SETUP packet at the *originator* PG and the arrival of a corresponding ACCEPT packet at the same PG.

We measured the PR setup overhead for routes ranging between 2 and 5 PG hops. (The laboratory topology was re-configured for the case of 5 hops). We measured the overhead in relation to the number of PG, not AD, hops. This is because traversing an AD may involve physically traversing one or more PG's. Therefore, it is more meaningful to measure per-PG, rather than per-AD, setup overhead. Transmission and propagation delays over the experimental local Ethernet were negligible and clearly not representative of what should be expected on interdomain connections.

The results are illustrated in Table I. They include MD4-based integrity check computation (for both SETUP and ACCEPT packets) at each PG hop.

The results show that the per PG setup cost averages about 55 ms. This number is fairly low considering that it includes the following sequence of events at each transit PG:

1. Context switch from Kernel to User mode to receive SETUP.
2. MD4 digest computation.
3. Validity checking (including timeliness, previous and next-hop reachability, and PT validation).
4. Context switch from User to Kernel mode to forward SETUP.

TABLE I  
PR SETUP OVERHEAD (IN MS)

# of PG hops	2	3	4	5
Per-hop overhead	50	57	54	58
Route computation	20	20	30	30
Setup overhead	50	170	215	290
Total overhead	70	190	245	320

5. Context switch from Kernel to User mode to receive ACCEPT.<sup>13</sup>
6. MD4 digest computation.
7. PR table lookup to locate the PR in question.
8. Context switch from User to Kernel mode to forward ACCEPT.

Moreover, the numbers include route computation overhead which is incurred at the *originator* PG.

Although the results appear reassuring, since the number of AD's in our laboratory internetwork was small and simple, overhead due to route computation at the *originator* and PT validation at all *transit* PG's was kept artificially low. Moreover, a real-world implementation might include encryption-based signatures which will greatly increase the PR setup overhead.

We also measured the overhead incurred by IDPR data packets in traversing previously established PR's. For data traffic, we used the ICMP Echo protocol [33]. The results in Table II represent the the round-trip times (in ms) for traversing routes of variable length. For each route length, we began by timing data packets without IDPR. These numbers provide a basis for comparison. We then timed the same packet flows across IDPR-controlled routes. This was done to isolate, as much as possible, the overhead due to additional packet length and IDPR-related processing in PG's. Finally, we *plugged in* MD4-based integrity checking to obtain the timings of IDPR packet switching.

Unlike PR setup overhead, the numbers for data packet overhead are quite realistic since all protocol features have been implemented. The only exception is the PR table lookup. Recall that a PG uses the PRid field present in each data packet to locate an appropriate PR table entry. Because the volume of traffic on our experimental *internetwork* was artificially low, PR table lookups were relatively fast. On the other hand, our table lookups were performed sequentially whereas a tuned implementation oriented toward supporting large volumes of inter-AD traffic would necessarily use some kind of a fast hashing technique for table lookups.

Judging by the above results, the overhead introduced by IDPR encapsulation and transit packet switching, while not negligible, appears to be sufficiently low to be of little hindrance to existing transport layer protocols. Even for large packets (e.g., 1Kbyte), IDPR-incurred delay amounts to less than 1 ms per PG hop. This is especially encouraging considering that the MD4 digest computation (performed at each hop) alone takes approximately 0.7 ms.

<sup>13</sup>This does not occur immediately after the previous step.

TABLE II  
PACKET ROUND-TRIP DELAY (IN MS)

Packet size (in bytes)	# of PG hops											
	2			3			4			5		
	a	b	c	a	b	c	a	b	c	a	b	c
16	4	4	4.5	5	6	7	8	7.5	9	10	10	12
64	4	5	5.5	5	7	8	8	9	10	10	12	16
256	6	7	8	8	9	10.5	13	12.5	16	14	17	18
512	8	9	10.5	10	11	13.5	17	17	22	19	20	26
1024	12	14	15	15	17	20	23	23.5	30	28	33	35

- a. without IDPR
- b. IDPR (no integrity checking)
- c. IDPR with MD4 integrity checking

#### D. Source Modifications and Status

The modifications to an existing SunOS 4.0 source in order to run the IDPR Kernel component are minimal. In fact, there are no changes to any of the existing protocols (including IP). The only changes are:

- IDPR System Calls—two new system calls, *IDPR.TABLE* and *IDPR.CTL*, are required to run IDPR.<sup>14</sup> The first is needed for manipulating the PR and HOST tables from the user level, and the second for setting the debugging level and other auxiliary functions. The files affected are *os/init\_sysent.c* and *os/syscalls.c*.
- IDPR Protocol Entry—the IDPR encapsulation/packet forwarding constitutes a new protocol as far as IP is concerned. Therefore, a new protocol descriptor entry has to be placed into the protocol switch table *inetsw*. This is done in *netinet/in\_proto.c*. Also, the IDPR protocol number needs to be placed in *netinet/in.h*.<sup>15</sup>

The current implementation executes packet forwarding in the Unix kernel. Setup is implemented almost completely outside of the kernel. The exceptions are: 1) kernel generates SETUP-REQUEST to Setup when it cannot find an entry in a HOST-TABLE, and 2) Setup installs PR table entries by making a system call to the kernel. Update, VGP, and Route Computation are applications outside of the kernel. A connectionless transport protocol (CMTP) was implemented to carry all IDPR control packets between adjacent PG's in a reliable fashion. All data packets travel via active policy routes. The prototype runs under SunOS 4.0 on Sun Sparcstations and 3/60s.

Future versions of the protocol will explore hierarchical configuration for the connectivity database in RS's and address the need for a community update mechanism, additional techniques for route computation, and interoperability with other inter-AD protocols such as BGP [24],[1].

<sup>14</sup>Numeric values corresponding to IDPR\_TABLE and IDPR\_CTL are system-dependent!

<sup>15</sup>An addition source change might possibly be needed in the future to trap and check all IP packets destined for the IDPR PG itself.

## VI. CONCLUSIONS AND OUTSTANDING ISSUES

The IDPR architecture supports policy and TOS-sensitive routing needed for communication in the global, multimedia internets of the future. The IDPR architecture is based on source routing, setup, and link-state advertisements with explicit policy terms. This approach supports flexible expression of policy with limited computational burden on transit AD's and a means of avoiding routing loops without strong consistency requirements. The architecture and constituent protocols described may be used alone or in combination with a hop-by-hop interdomain routing architecture such as BGP/IDRP [24],[1]. In a hybrid architecture, the source routing protocols described here would be used to support special policy and TOS routes, and hop-by-hop routing would be used for more generic traffic and routes. The Setup and Packet Forwarding protocols described herein would be applicable to this hybrid environment with little modification.

After summarizing the IDPR architecture, this paper focused on the Setup and Packet Forwarding protocols and provided initial experimental results. The performance measurements obtained from these experiments demonstrate the feasibility of implementing IDPR's controlled forwarding mechanisms.

Many outstanding issues remain to be investigated. We conclude with a list of open research issues, many of which pertain to scale.

- Connectivity Database:** The connectivity database maintains global information and therefore raises concerns regarding scale. The size of the database depends upon AD connectivity, as well as policies. In particular, the memory requirements for the database are a function of the number of transit AD's, the average number of neighbors for a transit AD, the number of policies expressed, and the uniformity with which the policies are applied. For example, if we assume that each of the 5000 transit AD's includes an average of 10 policy term entries in an update and an average of 50 VG's per transit AD, then approximately 2.5 Mbytes of storage is needed for the complete connectivity database. For further discussion, see [12]. A possible method for reducing connectivity database size is to introduce additional level(s) of abstraction or hierarchy, i.e., group AD's into super AD's and only flood update information among entities at the same level. However, in IDPR it is only meaningful to group AD's when policies are somewhat similar. When AD's are grouped, the policies advertised to other AD groups must represent policies acceptable to constituent AD's. Alternatively, the AD group must advertise multiple VG's to each neighbor group in order to indicate the different policies supported. If intra-group policies differ significantly, then the AD group will have to advertise a large collection of policies for a large number of VG's and the amount of information distributed will not be reduced. Moreover, from the perspective of logical connectivity (i.e., taking into account policy as well as topology), the more dissimilar the policies among constituent AD's, the more likely it is that the AD group will appear partitioned to external groups—an undesirable

situation. Moreover, if other transit AD's (or groups) have policies that distinguish among members of an AD group, then those AD's will still write policies at the lower granularity and policy routes must be generated and set up with the finer granularity (i.e., not for the AD group as a whole).

- Update Distribution:** Global flooding of update information also presents scaling problems. To analyze update overhead, we need to make additional assumptions about the number of PG's in an AD and the number of inter-AD links supported by each of the PG's. For the purpose of this approximation, we will assume 10 PG's per transit AD, where each PG is connected to five inter-AD links and each AD generates approximately one UPDATE per day. Although under these assumptions only 5000 routing UPDATE's are generated per day, a PG may receive duplicates because of the flooding technique used. Each PG will receive at most one copy of each UPDATE from each of the PG's to which it is connected in neighboring AD's (five in this example). In the worst case, each PG also receives one copy of the UPDATE from each of the other PG's in its own AD (nine in this example). Therefore, in this pessimistic example, each PG would see at most 14 duplicates of each new UPDATE—averaging to 50 per minute or less than 1 per second. This represents an aggregate of approximately 3 Kbytes per second of data over intra-AD resources. However, because internetwork structure more closely resembles a tree than a mesh, the general case should be much better. In other words, one PG in one VG typically will receive an UPDATE before any others in a given AD. Consequently, an AD would not experience updates from all VG's simultaneously and duplicates would be eliminated.<sup>16</sup> A possible method for reducing update distribution overhead is to limit the hop count on flooded packets. However, this assumes a topological locality of traffic that is not necessarily characteristic of much internet traffic. Consequently, we are considering use of mechanisms for distribution of routing UPDATE's along active policy routes to take advantage of route locality [3].
- Route Computation:** As described earlier, complete pre-computation of policy routes would be computationally infeasible in a large internet with multiple policies and types of service per AD. However, we wish to exploit the parallelism available when computing multiple routes simultaneously, and to reduce setup latency, by doing some precomputation. On-demand computation can use a simple breadth first search to locate a policy route that meets the conditions specified by setup. SPF algorithms such as [10] may be used instead, if some aspect(s) of policy are encoded in metrics before computation. The same approach can be used for precomputation; however, multiple routes are explored and few routes are pruned. We are exploring the use of several techniques

<sup>16</sup>We note that even the worst-case analysis results in a tolerable number, considering that the burden is spread across the AD as a whole, not carried over a single intra-AD link.

to address this problem. One approach is to make use of some distributed computation. For example, AD's may distribute several *route fragments* as part of their own with configured updates, i.e., partial source routes from the originating AD to a well-known backbone network.<sup>17</sup> Another approach is to precompute only generic routes, i.e., those that have minimal (or no) access restrictions and that support simple datagram TOS. This is an area of ongoing research. In the longer term, the best approach may be to support a hybrid architecture that uses hop-by-hop routing for generic best-effort packet delivery, and that uses source routing for delivery of special TOS streams over more restricted facilities [13]. In this context, the hop-by-hop routes might be used as route fragments or hints in source-route computation.

- **PG State:** As part of the setup protocol, there is state information established and maintained in each PG along a policy route. The state information is the forwarding information base used to forward data packets; consequently, it *cannot* be treated as *soft state* as defined by Clark [5]. The amount of state information maintained by a PG depends upon where it sits in the internet and how much traffic it sees. The memory requirements to store and maintain this state are significant for the PG's in very large transit AD's. Moreover, the more fine grained the policies expressed by transits and stub AD's, the larger the number of policy routes that individual stubs will set up and the greater the amount of state required. Caching strategies should be effective given some locality of communication. A hybrid source and hop-by-hop routing architecture would alleviate state proliferation (by aggregating state for the most common generic route) in addition to the route computation mentioned earlier. In addition, we could add an explicit source route capability to the protocol so that sources and transits could choose to operate in a stateless mode when desired.
- **Complexity and Efficiency:** We should investigate efficiency gains that could be realized by a best-effort network layer setup protocol in contrast to the reliable network layer, setup protocol described in this document. More generally, configuration proved to be one of the more troublesome aspects of the IDPR implementation. Although this is partially a matter of developing configuration management tools, we should consider designing such protocols with ease of configuration in mind from the start.
- **Relationship to New Public-Carrier Services:** It is an open question how the emergence of high-speed packet-switched communication services from the public carriers will affect the evolution of the internet and private data networking in general. While SMDS and ATM-based services will improve tremendously the availability of high-speed wide-area data communications, this does not preclude the need for an internet-wide routing architecture. It will certainly be a very long time (if ever) before the entire country is wired end to end with

<sup>17</sup>The concept of route fragments was first suggested by David Clark, personal communications.

affordable multiple-TOS switched high-speed services. Even when this does occur, it is questionable whether private networks will disappear. Computer communication requirements are diverse and continually changing. With every new application comes a new traffic distribution. Private networks allow the users to make their own tradeoffs between performance and price more flexibly than public offerings. In such a context, the need for *internetworking* and *interdomain* networking will persist and a routing architecture, with a strong flexible policy model, will be needed.

#### ACKNOWLEDGMENT

The protocols described in this paper were designed and implemented in collaboration with members of the Internet Inter-Domain Policy Routing Working Group and the Autonomous Networks Research Group. In particular, the following people contributed to the design of the IDPR architecture: L. Breslau, R. Callon, N. Chiappa, D. Clark, D. Estrin, M. Lepp, M. Little, T. Nakassis, M. Steenstrup, P. Tsuchiya, and G. Tsudik. In addition, the following people contributed to the specification and implementation of the IDPR protocols described in this document: H. Bowns, L. Breslau, K. Carlberg, I. Castineyra, D. Estrin, T. Li, M. Little, M. Steenstrup, G. Tsudik, and R. Woodburn. The authors would like to thank them all for their contributions to IDPR and, therefore, to the writing of this paper. They are also grateful to the anonymous referees who provided many constructive comments and suggestions.

#### REFERENCES

- [1] ANSI, "Intermediate system to intermediate system inter-domain routing information exchange protocol," ANSI Doc. X3S3.3/90-132, June 1990.
- [2] L. Breslau and D. Estrin, "Design and evaluation of inter-domain routing protocols," *Internetworking: Research and Experience*, vol. 2, no. 4, Dec. 1991.
- [3] D. Estrin, L. Breslau, and L. Zhang, "Exploiting locality to provide adaptive routing of real-time flows in global internets," in *Proc. IEEE Workshop on Multimedia Commun.*, Monterey, CA, Apr. 1992.
- [4] J.N. Chiappa, "A new IP routing and addressing architecture," Working Draft, 1991.
- [5] D. Clark, "Design philosophy of the DARPA internet protocols," in *Proc. ACM SIGCOMM*, Palo Alto, CA, Aug. 1988.
- [6] D. Clark, "Policy routing in internetworks," *Internetworking: Research and Experience*, vol. 1, no. 1, Sept. 1990.
- [7] D. Clark, V. Jacobson, and J. Romkey, "Analysis of TCP/IP processing overhead," *IEEE Commun. Mag.*, pp. 23-29, June 1989.
- [8] S. Deering, "Multicast routing in internetworks and extended LAN's," in *Proc. 1988 ACM SIGCOMM Symp.*, Aug. 1988.
- [9] Digital Equip. Corp., "Intermediate system to intermediate system intra-domain routing exchange protocol," Oct. 1989.
- [10] E. Dijkstra, "Self-stabilization in spite of distributed control," *Commun. ACM*, Nov. 1974.
- [11] D. Estrin, "Policy requirements for inter administrative domain routing," *Comput. Netw. and ISDN Syst.*, vol. 22, no. 3, Oct. 1991.
- [12] D. Estrin and K. Obraczka, "Connectivity database overhead for inter-domain policy routing," in *Proc. IEEE INFOCOM '91*, Miami, FL, May 1990.
- [13] D. Estrin, Y. Rekhter, and S. Hotz, "Scalable inter-domain architecture," in *Proc. ACM SIGCOMM*, Baltimore, MD, Aug. 1992.
- [14] D. Estrin, J. Mogul, and G. Tsudik, "VISA protocols for controlling inter-organizational datagram flow," *IEEE J. Select. Areas Commun.*, vol. 7, no. 4, May 1989.
- [15] D. Estrin and G. Tsudik, "An end-to-end argument for network layer inter-domain access controls," *Internetworking: Research and Experience*, vol. 2, no. 2, June 1991.
- [16] D. Estrin and G. Tsudik, "Secure control of transit internetwork traffic," *Comput. Netw. and ISDN Syst.*, vol. 22, no. 5, Oct. 1991.

- [17] L. Ford and D. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton Univ. Press, 1962.
- [18] S. Hares and D. Katz, "Administrative domains and routing domains: A model for routing in the internet," RFC 1136, SRI Netw. Inform. Cent., Dec. 1989.
- [19] C. Hedrick, "An introduction to IGRP," Rutgers Univ. Tech. Rep., Oct. 1989.
- [20] ISO, "OSI routing framework," ISO/TF 9575, 1989.
- [21] P. Karger, "Authentication and discretionary control in computer networks," *Comput. Netw. and ISDN Syst.*, Jan. 1986.
- [22] S. Kent, *Protocols and Techniques for Data Communication Networks*. Englewood Cliffs, NJ: Prentice Hall, 1980.
- [23] M. Lepp and M. Steenstrup, "An architecture for inter-domain policy routing," BBN Tech. Rep. 7345, July 1990.
- [24] K. Lougheed and Y. Rekhter, "Border gateway protocol," RFC 1163, SRI Netw. Inform. Cent., June 1990.
- [25] J.M. McQuillan, I. Richer, and E. Rosen, "The new routing algorithm for the ARPANET," *IEEE Trans. Commun.*, vol. 28, pp. 711-719, 1980.
- [26] J. McQuillan, "Adaptive routing algorithms for distributed computer networks," BBN Tech. Rep. 2831, May 1974.
- [27] C. Hedrick, "Routing information protocol," RFC 1058, SRI Netw. Inform. Cent., June 1988.
- [28] J. Moy, "The open shortest path first (OSPF) specification," RFC 1131, SRI Netw. Inform. Cent., Oct. 1989.
- [29] T. Narten, "Internet routing," in *Proc. 1988 ACM SIGCOMM Symp.*, Palo Alto, CA, Aug. 1989.
- [30] D. Nessett, "Factors affecting distributed system security," *IEEE Trans. Software Eng.*, vol. SE-13, 1987.
- [31] R. Perlman, "Fault-tolerant broadcast of routing information," *Comput. Netw. and ISDN Syst.*, vol. 7, no. 3, Dec. 1983.
- [32] J. Postel, "Internet protocol," RFC 791, SRI Netw. Inform. Cent., Sept. 1981.
- [33] J. Postel, "Internet control message protocol," RFC 792, SRI Netw. Inform. Cent., Sept. 1981.
- [34] Y. Rekhter, "Constructing intra-AS path segments for an inter-AS path," *ACM Comput. Commun. Rev.*, vol. 21, no. 1, Jan. 1991.
- [35] R. Rivest, "The MD4 message digest algorithm," in *Proc. CRYPTO'90*, Santa Barbara, CA, Aug. 1990.
- [36] R. Rivest, "The MD5 message digest algorithm," INTERNET Draft, available via anonymous FTP from RSA.COM, July 1991.
- [37] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, Feb. 1978.
- [38] E. Rosen, "Exterior gateway protocol (EGP)," RFC 827, SRI Netw. Inform. Cent., Oct. 1982.
- [39] M. Steenstrup, "Inter-domain policy routing protocol specification and usage: Version 1," BBN Tech. Rep. 7346, Feb. 1991.
- [40] J. Steiner, B.C. Neuman, and J. Schiller, "Kerberos: An authentication service for open network systems," in *Proc. USENIX Wint. Conf.*, Feb. 1988.
- [41] R. Woodburn and D. Mills, "A scheme for an internet encapsulation protocol: Version 1," RFC 1241, SRI Netw. Inform. Cent., July 1991.
- [42] P. Tsuchiya, "Efficient and robust policy routing using multiple hierarchical addresses," in *Proc. ACM SIGCOMM '91*, Zurich, Switzerland, Sept. 1991.
- [43] G. Tsudik, "Message authentication with one-way hash functions," in *Proc. IEEE INFOCOM '92*, May 1992.
- [44] G. Tsudik, "Access control and policy enforcement in internetworks," Ph.D. Dissert., Univ. of South. Cal., Apr. 1991.
- [45] V. Voydock and S. Kent, "Security mechanisms in high-level network protocols," *ACM Comput. Surv.*, June 1983.

**Deborah Estrin** (S'78-M'80-S'81-M'85), photograph and biography not available at the time of publication.

**Martha Steenstrup** (S'83-M'83-S'84-M'84), photograph and biography not available at the time of publication.

**Gene Tsudik** (S'87-M'91-M'91), photograph and biography not available at the time of publication.