Scalable Inter-Domain Routing Architecture

Deborah Estrin Yakov Rekhter
Steve Hotz[1]

Abstract

As internets grow, both in size and in the diversity of routing requirements, providing inter-domain routing that can accommodate both of these factors becomes increasingly crucial. The combinatorial explosion of mixing and matching different routing criteria weighs heavily on the mechanisms provided by conventional hop-by-hop routing architectures. We expect that over the next 5 to 10 years, the types of services available will continue to evolve and that specialized facilities will be employed to provide new services. While the number and variety of routes provided by hop-by-hop routing architectures with type of service support (i.e., multiple, tagged routes) may be sufficient for a large percentage of traffic, it is important that mechanisms be in place to support efficient routing of specialized traffic types via special routes.

Our desire to support special routes efficiently led us to investigate the dynamic installation of routes ([Breslau-Estrin 91, Clark 90, IDPR90]). In a previous paper ([Breslau-Estrin 91]), we evaluated the algorithmic design choices for inter-domain policy routing with specific attention to accommodating source-specific and other "special" routes. The conclusion was that special routes are best supported with source-routing and extended link-state algorithms (we refer to this approach as source-demand routing). However, a source-demand routing architecture, used as the **only** means of inter-domain routing, has scaling problems because it does not lend itself to general aggregation and abstraction of routing and forwarding information. On the other hand, while hop-by-hop approaches may have better scaling characteristics, they do not provide support for specialized routing requirements as flexibly and efficiently as source-demand routing.

We want to support special routes **and** we want to exploit aggregation when a special route is not needed. Therefore, we propose a scalable inter-domain routing architecture consisting of two major components: source-demand routing (SDR) and node routing (NR). The NR component computes and installs routes that are shared by a significant number of sources. These *generic routes* are commonly used and warrant wide propagation. Consequently, aggregation of routing information is critical. The SDR component computes and installs *specialized routes* that are not shared by enough sources to justify computation by NR. The potentially large number of different specialized routes, combined with their sparse utilization, make them too costly to support with the NR mechanism.

To complement earlier discussions of SDR design choices [Breslau-Estrin 91], this paper evaluates the algorithmic design choices for the NR component in terms of scalability and functionality. In addition, we discuss mechanisms for improving the scaling properties of link-state SDR, and for integrating the two components of the architecture.

[1]D. Estrin and S. Hotz are reachable c/o Computer Science Department, University of Southern California, Los Angeles, California 90089-0782, e-mail: estrin@usc.edu, hotz@usc.edu. Y. Rekhter is reachable c/o IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY, 10598, e-mail: yakov@ibm.com. The work of D. Estrin and S. Hotz was supported by the National Science Foundation under contract number NCR-9011279, with matching funds from GTE Laboratories. The work of Y. Rekhter was supported by the Defense Advanced Research Projects Agency, under contract DABT63-91-C-0019. Views and conclusions expressed in this paper are not necessarily those of the Defense Advanced Research Projects Agency and National Science Foundation.

1 Motivation

The global internet can be modeled as a collection of hosts interconnected via transmission and switching facilities. Control over the collection of hosts and the transmission and switching facilities that compose the networking resources of the global internet is not homogeneous, but is distributed among multiple administrative authorities. Resources under control of a single administration form a **domain**. In order to support each domain's autonomy and heterogeneity, routing consists of two distinct components: **intra-domain** (interior) routing, and **inter-domain** (exterior) routing. Intra-domain routing provides support for data communication between hosts where data traverses transmission and switching facilities within a *single* domain. Inter-domain routing provides support for data communication between hosts where data traverses transmission and switching facilities spanning *multiple* domains. The entities that forward packets across domain boundaries are called **border routers** (BRs). The entities responsible for exchanging inter-domain routing information are called **route servers** (RSs). RSs and BRs may be co-located.

As the global internet grows, both in size and in the diversity of routing requirements, providing inter-domain routing that can accommodate both of these factors becomes more and more crucial. The number and diversity of routing requirements is increasing due to: (a) transit restrictions imposed by source, destination, and transit networks, (b) different types of services offered, and (c) the presence of multiple carriers with different charging schemes. The combinatorial explosion of mixing and matching these different criteria weighs heavily on the mechanisms that had been provided by conventional hop-by-hop routing architectures ([ISIS10589, OSPF, Hedrick 88, EGP]).

We expect that over the next 5 to 10 years, the types of services available will continue to evolve and that specialized facilities will be employed to provide new services. While the number and variety of routes provided by hop-by-hop routing architectures with **type of service** (TOS) support (i.e., multiple, tagged routes) may be sufficient for a large percentage of traffic, it is important that mechanisms be in place to support efficient routing of specialized traffic types via special routes. Examples of *special* routes are: (1) a route that travels through one or more transit domains that discriminate according to the source domain, (2) a route that travels through transit domains that support a service that is not widely or regularly used. We refer to all other routes as *generic*.

Our desire to support special routes efficiently led us to investigate the dynamic installation of routes ([Breslau-Estrin 91, Clark 90, IDPR90]). In a previous paper ([Breslau-Estrin 91]), we evaluated the algorithmic design choices for inter-domain policy routing with specific attention to accommodating source-specific and other "special" routes. The conclusion was that special routes are best supported with source-routing and extended link-state algorithms[2] (we refer to this approach as source-demand routing). However, a source-demand routing architecture, used as the **only** means of inter-domain routing, has scaling problems because it does not lend itself to general hierarchical clustering and aggregation of routing and forwarding information. For example, even if a particular route from an intermediate transit domain **X**, to a destination domain **Y** is shared by 1,000 source-domains, IDPR requires that state for each of the 1,000 routes be setup and maintained in the transit border routers between **X** and **Y**. In contrast, an alternative approach to inter-domain routing, based on hop-by-hop routing and a distributed route-computation algorithm (described later), provides extensive support for aggregation and abstraction of reachability, topology, and forwarding information. The Border Gateway Protocol (BGP) and Inter-Domain Routeing Protocol

[2]The Inter-Domain Policy Routing (IDPR) architecture uses these techniques [IDPR90].

(IDRP) use these techniques ([BGP91, IDRP91]). However, the BGP/IDRP architecture does not provide support for *specialized* routing requirements as flexibly and efficiently as IDPR-style routing.

In summary, we want to support special routes **and** we want to exploit aggregation when a special route is not needed. Therefore, our scalable inter-domain routing architecture consists of two major components: **source-demand routing** (SDR), and **node routing** (NR). The NR component computes and installs routes that are shared by a significant number of sources. These generic routes are commonly used and warrant wide propagation, consequently, aggregation of routing information is critical. The SDR component computes and installs specialized routes that are not shared by enough sources to justify computation by NR[3]. The potentially large number of different specialized routes, combined with their sparse utilization, make them too costly to support with the NR mechanism.

A useful analogy to this approach is the manufacturing of consumer products. When predictable patterns of demand exist, firms produce objects and sell them as "off the shelf" consumer goods. In our architecture NR provides "off-the-shelf" routes. If demand is not predictable, then firms accept special orders and produce what is demanded at the time it is needed. In addition, if a part is so specialized that only a single or small number of consumers need it, the consumer may repeatedly special order the part, even if it is needed in a predictable manner, because the consumer does not represent a big enough market for the producer to bother managing the item as part of its regular production. SDR provides such special order, on-demand routes.

By combining NR and SDR routing we propose to support inter-domain routing in internets of practically-unlimited size, while at the same time providing efficient support for specialized routing requirements. The development of this architecture does assume that routing requirements will be diverse and that special routes will be needed. On the other hand, the architecture does not depend on assumptions about the particular types of routes demanded or on the distribution of that demand. The routing will adapt naturally over time to shifting traffic patterns and new services by shifting computation and installation of particular types of routes between the two components of the hybrid architecture[4].

In a previous paper Breslau and Estrin ([Breslau-Estrin 91]) evaluated the algorithmic design choices for inter-domain policy routing with specific attention to accommodating source-specific routing. We adopt their conclusions for the SDR component: special routes are best supported with source-routing and extended link-state algorithms. Consequently, this paper focuses on the algorithmic design choices for the NR component with emphasis on scalability and functionality. In addition, we discuss mechanisms for improving the scaling properties of link-state SDR.

The remainder of this paper is organized as follows. Section 2 outlines the requirements and priorities that guide the design of both NR and SDR components, with particular attention to

[3] Routes that are only needed sporadically (i.e., the demand for them is not continuous or otherwise predictable) are also candidates for SDR.

[4] Before continuing with our explanation of this architecture, we wish to state up front that supporting highly specialized routes for **all source-destination pairs** in an internet, or even anything close to that number, is not feasible in **any** routing architecture that we can foresee. In other words, we do not believe that any foreseeable routing architecture can support unconstrained proliferation of user requirements and network services. At the same time, this is not necessarily a problem. The capabilities of the architecture may in fact exceed the requirements of the users. Moreover, some of the requirements that we regard as infeasible from the inter-domain routing point of view, may be supported by means completely outside of routing. Nevertheless, the caveat is stated here to preempt unrealistic expectations.

scaling issues. Section 3 describes the NR design choices in light of these requirements. Section 4 briefly describes scaling techniques for SDR and Section 5 concludes the paper with a brief discussion of SDR-NR integration.

2 Architectural Requirements and Priorities

In order to justify our design choices for a scalable inter-domain routing architecture, we must articulate our evaluation criteria and priorities. This section defines complexity, abstraction, policy, and type of service requirements.

2.1 Complexity

Inter-domain routing complexity must be evaluated on the basis of the following performance metrics: (1) storage overhead, (2) computational overhead, and (3) message overhead. This evaluation is essential to determining the scalability of any architecture.

2.1.1 Storage Overhead

The storage overhead of an entity that participates in inter-domain routing comes from two sources: **Routing Information Base** (RIB), and **Forwarding Information Base** (FIB) overhead. The RIB contains the routing information that entities exchange via the inter-domain routing protocol; the RIB is the input to the route computation. The FIB contains the information that the entities use to forward the inter-domain traffic; the FIB is the output of the route computation. For an acceptable level of storage overhead, the amount of information in both FIBs and RIBs should grow significantly slower than linearly (e.g. close to logarithmically) with the total number of domains in an internet. To satisfy this requirement with respect to the RIB, the architecture must provide mechanisms for either aggregation/abstraction of routing and forwarding information, or retrieval of a subset of this information *on demand*. To satisfy this requirement with respect to the FIB, the architecture must provide mechanisms for either aggregation of the forwarding information (for the NR computed routes), or dynamic installation/tear down of this information (for the SDR computed routes).

2.1.2 Computational Overhead

The NR component will rely primarily on precomputation of routes. If inter-domain routing is ubiquitous, then the precomputed routes include *all* reachable destinations. Even if policy constraints make fully ubiquitous routing impossible, the precomputed routes are likely to cover a very large percentage of all reachable destinations. Therefore the complexity of this computation must be as small as possible. Specifically, it is highly desirable that the architecture would employ some form of partial computation, where changes in topology would require less than complete recomputation. Even if complete recomputation is necessary, its complexity should be less than linear with the total number of domains.

The SDR component will use on-demand computation and caching. Therefore the complexity of this computation can be somewhat higher. Another reason for relaxed complexity requirements for SDR is that SDR is expected to compute routes to a smaller number of destinations than is NR (although SDR route computation may be invoked more frequently).

Under *no circumstances* is computational complexity allowed to become exponential (for either the NR or SDR component).

2.1.3 Bandwidth Overhead

The bandwidth consumed by routing information distribution should be limited. However, the possible use of data compression techniques and the increasing speed of network links make this less important than route computation and storage overhead. Bandwidth overhead may be further contained by using incremental (rather than complete) exchange of routing information.

While storage and bandwidth overhead may be interrelated, if incremental updates are used then bandwidth overhead is negligible in the steady state (no changes in topology), and is independent of the storage overhead. In other words, use of incremental updates constrains the bandwidth overhead to the dynamics of the internet. Therefore, improvements in stability of the physical links, combined with techniques to dampen the effect of topological instabilities, will make the bandwidth overhead even less important.

2.2 Aggregation

Aggregation and abstraction of routing and forwarding information provides a very powerful mechanism for satisfying storage, computational, and bandwidth constraints. The ability to aggregate, and subsequently abstract, routing and forwarding information is essential to the scaling of the architecture[5]. This is especially true with respect to the NR component, since the NR component must be capable of providing routes to *all* or *almost all* reachable destinations.

At the same time, since preserving each domain's independence and autonomy is one of the crucial requirements of inter-domain routing, the architecture must strive for the maximum flexibility of its aggregation scheme, i.e., impose as few preconditions, and as little global coordination, as possible on the participating domains.

The Routing Information Base (RIB) carries three types of information: (1) topology (i.e., the interconnections between domains or groups of domains), (2) network layer reachability, and (3) transit constraint. Aggregation of routing information should provide a reduction of all three components. Aggregation of forwarding information will follow from reachability information aggregation.

Clustering (by forming *routing domain confederations*) serves the following aggregation functions: (1) to hide parts of the actual physical topology, thus abstracting topological information, (2) to combine a set of reachable destination entities into a single entity and reduce storage overhead,

[5]While we can not prove that there are no other ways to achieve scaling, we are not aware of any mechanism other than clustering that allows information aggregation/abstraction. Therefore, the rest of the paper assumes that clustering is used for information aggregation/abstraction.

and (3) to express transit constraints in terms of clusters, rather than individual domains.

As argued in [Breslau-Estrin 91], the architecture must allow confederations to be formed and changed without extensive configuration and coordination; in particular, forming a confederation should not require *global* coordination (such as that required in ECMA ([ECMA89]). In addition, aggregation should not require explicit designation of the relative placement of each domain relative to another; i.e., domains or confederations of domains should not be required to agree on a partial ordering (i.e., who is above whom, etc.).

The architecture should allow different domains to use different methods of aggregation and abstraction. For example, a research collaborator at IBM might route to USC as a domain-level entity in order to take advantage of some special TOS connectivity to, or even through, USC. Whereas, someone else at Digital Equipment Corporation might see information at the level of the California Educational Institutions Confederation, and know only that USC is a member. Alternatively, USC might see part of the internal structure within the IBM Confederation (at the domain's level), whereas UCLA may route based on the confederation of IBM domains as a whole.

Support for confederations should be flexible. Specifically, the architecture should allow confederations to overlap without being nested, i.e., a single domain, or a group of domains may be part of more than one confederation. For example, USC may be part of the California Educational Institutions Confederation and part of the US R&D Institutions Confederation; one is not a subset of the other. Another example: T.J. Watson Research Center might be part of NYSERNET Confederation and part of IBM-R&D-US Confederation. While the above examples describe cases where overlap consists of a single domain, there may be other cases where multiple domains overlap. As an example consider the set of domains that form the IBM Confederation, and another set of domains that form the DEC Confederation. Within IBM there is a domain IBM-Research, and similarly within DEC there is a domain DEC-Research. Both of these domains could be involved in some collaborative effort, and thus have established direct links. The architecture should allow restricted use of these direct links, so that other domains within the IBM Confederation would not be able to use it to talk to other domains within the DEC Confederation. A similar example exists when a multinational corporation forms a confederation, and the individual branches within each country also belong to their respective country confederations. The corporation may need to protect itself from being used as an inter-country transit domain (due to internal, or international, policy). All of the above examples illustrate a situation where confederations overlap, and it is necessary to control the traffic traversing the overlapping resources.

While flexible aggregation should be accommodated in any inter-domain architecture, the extent to which this feature is exploited will have direct a effect on the scalability associated with aggregation. At the same time, the exploitation of this feature depends on the way addresses are assigned. Specifically, scaling associated with forwarding information depends heavily on the assumption that there will be general correspondence between the hierarchy of address registration authorities, and the way routing domains and routing domain confederations are organized (see Section 2.6).

2.3 Routing Policies

Routing policies that the architecture must support may be broadly classified into *transit* policies and *route selection* policies [Breslau-Estrin 91, Estrin 89].

Restrictions imposed via transit policies may be based on a variety of factors. The architecture should be able to support restrictions based on the source, destination, type of services (TOS), user class identification (UCI), charging, and path [Estrin 89, Little 89]. The architecture must allow expression of transit policies on **all** routes, both NR and SDR. Even if NR routes are widely used and have fewer source or destination restrictions, NR routes may have some transit qualifiers (e.g. TOS, charging, or user-class restriction). If the most widely-usable route to a destination has policy qualifiers, it should be advertiseable by NR and the transit constraints should be explicit.

Route selection policies enable each domain to select a particular route among multiple routes to the same destination. To maintain maximum autonomy and independence between domains, the architecture must support heterogeneous route selection policies, where each domain can establish its own criteria for selecting routes. This argument was made earlier with respect to source route selection ([IDPR90, Clark 90, Breslau-Estrin 91]). In addition, each intermediate transit domain must have the flexibility to apply its own selection criteria to the routes made available to it by its neighbors. This is really just a corollary to the above; i.e., if we allow route selection policy to be expressed for NR routes, we can not assume all domains will apply the same policy. The support for dissimilar route selection policies is one of the key factors that distinguish inter-domain routing from the intra-domain[6] ([ECMA89, Estrin 89]).

2.3.1 Routing Information Hiding

The architecture should not require *all* domains within an internet to reveal their connectivity and transit constraints to each other. Domains should be able to enforce their transit policies while limiting the advertisement of their policy and connectivity information as much as possible; such advertisement will be driven by a "need to know" criteria. Moreover, advertising a transit policy to domains that can not use this policy will increase the amount of routing information that must be stored, processed, and propagated. Not only may it be impractical for a router to maintain full inter-domain topology and policy information, it may not be permitted to obtain this information.

2.3.2 Policies *Not* Supported

In this and previous papers we have argued that a global inter-domain routing architecture should support a wide range of policies. In this section we identify several types of policy that we explicitly do *not* propose to support. In general our reasoning is pragmatic; we think such policy types are either very expensive in terms of overhead, or may lead to routing instabilities.

Route selection policies contingent on external behavior. The route selection process may be modeled by a function that assigns a non-negative integer to a route, denoting the degree of preference. Route selection applies this function to all feasible routes to a given destination, and selects the route with the highest value. To provide a stable environment, the preference function should not use as an input the status and attributes of other routes (either to the same or to a different destination).

[6] It is a non-goal of the architecture to support *all possible* route selection policies. Specifically it is a non-goal to support any route selection policies that may potentially lead to routing instabilities. For more on unsupported route selection policies see Section 2.3.2.

Transit policies contingent on external behavior. To provide a stable environment, the domain's transit policies can not be **automatically** affected by *any* information external to the domain. Specifically, these policies can not be modified, automatically, in response to information about other domains' transit policies, or routes selected by local or other domains. Similarly, transit policies can not be automatically modified in response to information about performance characteristics of either local or external domains.

Policies contingent on external state (e.g. load). It is a non-goal of the architecture to support load-sensitive routing for generic routes. However, it is possible that some types of service may employ load information to select among alternate SDR routes.

Very large number of simultaneous SDR routes. It is a non-goal of the architecture to support a very large number of **simultaneous** SDR routes through any single router. Specifically, the FIB storage overhead associated with SDR routes must be comparable with that of NR routes, and should always be bound by the complexity requirements outlined earlier[7].

2.4 Support for TOS Routing

Throughout this document we refer to support for type of service (TOS) routing. There is a great deal of research and development activity currently underway to explore network architectures and protocols for high-bandwidth, multimedia traffic. Some of this traffic is delay sensitive, while some requires high throughput. It is unrealistic to assume that a single communication fabric will be deployed homogeneously across the internet (including all metropolitan, regional, and backbone networks) that will support all types of traffic uniformly. To support diverse traffic requirements in a heterogeneous environment, various resource management mechanisms will be used in different parts of the global internet (e.g., resource reservation of various kinds) [ST2-90, Zhang 91].

In this context, it is the job of routing protocols to locate routes that can *potentially* support the particular TOS requested. It is explicitly *not* the job of the general routing protocols to locate routes that are guaranteed to have resources available *at the particular time of the route request.* In other words, it is not practical to assume that instantaneous resource availability will be known at all remote points in the global internet. Rather, once a TOS route has been identified, an application requiring particular service guarantees will attempt to use the route (e.g., using an explicit setup message if so required by the underlying networks). In Section 4.3 we describe additional services that may be provided to support more adaptive route selection for special TOS[8].

While the NR component is optimized to satisfy the common case routing requirements for an extremely large population of users, this does not imply that routes produced by the NR component would be used for only a single type of service. To the contrary, as a new type of service becomes sufficiently widely used (i.e., by multiple domains and predictably over time), it may warrant being computed and installed by the NR component.

[7] As discussed earlier, theoretically the state overhead could grow $O(N^2)$ with the number of domains in an internet. However, there is no evidence or intuition to suggest that this will be a limiting factor on the wide utilization of SDR, provided that NR is available to handle the generic routes.

[8] Adaptive route selection implies adaptability *only* during the route selection process. Once a route is selected, it is not going to be a subject to any adaptations, except when it becomes infeasible.

2.5 Commonality between Routing Components

While it is acceptable for the NR and SDR components to be dissimilar, we do recognize that such a solution is less desirable – all other things being equal. In theory, there are advantages in having the NR and SDR components use similar algorithms and mechanisms. Code and databases could be shared and the architecture would be more manageable and comprehensible. On the other hand, there may be some benefit (e.g. robustness) if the two parts of the architecture are heterogeneous, and not completely inter-dependent. In Section 5 we list several areas in which opportunities for increased commonality (unification) will be exploited.

2.6 Interaction with Addressing

The proposed architecture should be applicable to various addressing schemes. There are no specific assumptions about a particular address structure, except that this structure should facilitate information aggregation, without forcing rigid hierarchical routing.

Beyond this requirement, most of the proposals for extending the IP address space, for example, can be used in conjunction with our architecture. But our architecture itself does not provide (or impose) a particular solution to the addressing problem.

The potential reduction of routing and forwarding information depends very heavily on the way addresses are assigned and how domains and their confederations are structured. "If there is no correspondence between the address registration hierarchy and the organization of routeing domains, then ... each and every routeing domain in the OSIE would need a table entry potentially at every boundary IS of every other routeing domain" ([Oran 89]). Indeed, scaling in the NR component is almost entirely predicated on the assumption that there will be general correspondence between the hierarchy of address assignment authorities and the way routing domains are organized, so that the efficient and frequent aggregation of routing and forwarding information will be possible. Therefore, given the nature of inter-domain routing in general, and the NR component in particular, scalability depends on the flexibility of the scheme for information aggregation and abstraction, and on the preconditions imposed by such a scheme. Moreover, given a flexible architecture, the operational efficiency (scalability) of the global internet, or portions thereof, will depend on tradeoffs made between flexibility and aggregation.

3 Design Choices for Node Routing (NR)

This section addresses the design choices made for the NR component in light of the above architectural requirements and priorities. *All of our discussion of NR assumes hop-by-hop routing.* Source routing is subject to different constraints and is used for the complementary SDR component.

3.1 Overview of NR

The NR component is designed and optimized for an environment where a large percentage of packets will travel over routes that can be shared by multiple sources and that have predictable traffic patterns. The efficiency of the NR component improves when a number of source domains share a particular route from some intermediate point to a destination. Moreover, NR is best suited to provide routing for inter-domain data traffic that is either steady enough to justify the existence of a route, or predictable, so that a route may be installed when needed (based on the prediction, rather than on the actual traffic). Such routes lend themselves to *distributed route computation and installation procedures.*

Routes that are installed in routers, and information that was used by the routers to compute these routes, reflect the known operational state of the routing facilities (as well as the policy constraints) at the time of route computation. Route computation is driven by changes in either operational status of routing facilities or policy constraints. The NR component supports route computation that is dynamically adaptable to both changes in topology and policy. The NR component allows time-dependent selection or deletion of routes. However, time dependency must be predictable (e.g., advertising a certain route only after business hours) and routes should be used widely enough to warrant inclusion in NR.

The proposed architecture assumes that most of the inter-domain conversations will be forwarded via routes computed and installed by the NR component. Moreover, the exchange of routing information necessary for the SDR component depends on facilities provided by the NR component; i.e., NR policies must allow SDR reachability information to travel. Therefore, the architecture requires that all domains in an internet implement and participate in NR. Since scalability (with respect to the size of the internet) is one of the fundamental requirements for the NR component, it must provide multiple mechanisms with various degree of sophistication for information aggregation and abstraction.

3.2 Routing Algorithm Choices for NR

Given that a NR component based on hop-by-hop routing is needed to provide scalable, efficient inter-domain routing, the remainder of this section considers the fundamental design choices for the NR routing algorithm.

Typically the debate surrounding routing algorithms focuses on link state and distance vector protocols. However, simple distance vector protocols (i.e., Routing Information Protocol [Hedrick 88]), do not scale because of convergence problems. Improved distance vector protocols, such as those discussed in [Jaffee 82, Zaumen 91, Shin87], have been developed to address this issue using synchronization mechanisms or additional path information. In the case of inter-domain routing, having additional path information is essential to supporting policy. Therefore, the algorithms we consider for NR are link state and one we call **path vector** (PV). Whereas the characteristics of link state algorithms are generally understood[9] (for example, [Zaumen 91]), we must digress from our evaluation discussion to describe briefly the newer concept of the PV algorithm.

[9] We assume that some form of SPF algorithm will be used to compute paths over the topology database when LS algorithms are used [Dijkstra 59, OSPF].

3.2.1 Path Vector Protocol Overview

The Border Gateway Protocol (BGP) (see [BGP91]) and the Inter Domain Routing Protocol (IDRP) (see [IDRP91]) are examples of path vector (PV) protocols[10].

The routing algorithm employed by PV bears a certain resemblance to the traditional Bellman-Ford routing algorithm in the sense that each BR[11] advertises the destinations it can reach to its neighboring BRs. However, the PV routing algorithm augments the advertisement of reachable destinations with information that describes various properties of the paths to these destinations. This information is expressed in terms of *path attributes*. To emphasize the tight coupling between the reachable destinations and properties of the paths to these destinations, PV defines a route as a *pairing* between a destination and the attributes of the path to that destination. Thus the name, **path-vector** protocol, where a BR receives from its neighboring BR a *vector* that contains *paths* to a set of destinations[12]. The path, expressed in terms of the domains (or confederations) traversed so far, is carried in a special path attribute which records the sequence of routing domains through which the reachability information has passed. Suppression of routing loops is implemented via this special path attribute, in contrast to LS and distance vector which use a globally-defined monotonically-increasing metric for route selection [Shin87].

Because PV does not require all routing domains to have homogeneous criteria (policies) for route selection, route selection policies used by one routing domain are not necessarily known to other routing domains. To maintain the maximum degree of autonomy and independence between routing domains, each domain which participates in PV may have its own view of what constitutes an optimal route. This view is based solely on local route selection policies and the information carried in the path attributes of a route. PV standardizes only the results of the route selection procedure, while allowing the selection policies that affect the route selection to be non-standard[13].

3.3 Complexity

Given the above description of PV algorithms, we can compare them to LS algorithms in terms of the three complexity parameters defined earlier.

3.3.1 Storage Overhead

Without any aggregation of routing information, and ignoring storage overhead associated with transit constraints, it is possible to show that under some rather general assumptions the average case RIB storage overhead of the PV scheme for a single TOS ranges between $O(N)$ and $O(N \times \log(N))$, where N is the total number of routing domains ([Rekhter 91]). The LS RIB, with no aggregation of routing information, no transit constraints, a single homogeneous route selection

[10] BGP is an inter-autonomous system routing protocol for TCP/IP internets. IDRP is an OSI inter-domain routing protocol that is being progressed toward standardization within ISO. Since in terms of functionality BGP represents a proper subset of IDRP, for the rest of the paper we will only consider IDRP.

[11] Routers that participate in IDRP are called **Border Intermediate Systems - BISs**. In this document we refer to them as Border Routers (BRs).

[12] The term "path-vector protocol" bears an intentional similarity to the term "distance-vector protocol," where a BR receives from each of its neighbors a *vector* that contains *distances* to a set of destinations.

[13] This succinct observation is attributed to Ross Callon (Digital Equipment Corporation).

policy across *all* the domains, and a single TOS, requires a complete domain-level topology map whose size is $O(N)$.

Supporting heterogeneous route selection and transit policies with hop-by-hop forwarding and LS requires each domain to know *all* other domains route selection and transit policies. This may significantly increase the amount of routing information that must be stored by each domain. If the number of policies advertised grows with the number of domains, then the storage could become unsupportable. In contrast, support for heterogeneous route selection policies has *no* effect on the storage complexity of the PV scheme (aside from simply storing the local policy information). The presence of transit constraints in PV results in a restricted distribution of routing information, thus further reducing storage overhead. In contrast, with LS no such reduction is possible since each domain must know every other domain's transit policies. Finally, some of the transit constraints (e.g. path sensitive constraints) can be expressed in a more concise form in PV (see aggregation discussion below).

The ability to further restrict storage overhead is facilitated by the PV routing algorithm, where route computation precedes routing information dissemination, and only routing information associated with the routes selected by a domain is distributed to adjacent domains. In contrast, route selection with LS is decoupled from the distribution of routing information, and has no effect on such distribution.

While theoretically routing information aggregation can be used to reduce storage complexity in both LS and PV, only aggregation of topological information would yield the same gain for both. Aggregating transit constraints with LS may result in either reduced connectivity or less information reduction, as compared with PV. Aggregating heterogeneous route selection policies in LS is *highly* problematic, at best. In PV, route selection policies are not distributed, thus making aggregation of route selection policies a non-issue[14].

Support for multiple TOSs has the same impact on storage overhead for both LS and for PV.

Potentially the LS FIB may be smaller if routes are computed at each node on demand. However, the gain of such a scheme depends heavily on the traffic patterns, memory size, and caching strategy. If there is not a high degree of locality, severely degraded performance can result due to excessive overall computation time and excessive computation delay when forwarding packets to a new destination. If demand driven route computation is not used for LS, then the size of the FIB would be the same for both LS and PV.

3.3.2 Route Computation Complexity

Even if all domains employ *exactly* the same route selection policy, computational complexity of PV is smaller than that of LS. The PV computation consists of evaluating a newly arrived route and comparing it with the existing one[15]. Whereas, conventional LS computation requires execution of

[14] Although a domain's selection policies are not explicitly distributed, they have an impact on the routes available to other domains. A route that may be preferred by a particular domain, and not prohibited by transit restrictions, may still be unavailable due to the selection policies of some intermediate domain. The ability to compute and install alternative routes that may be lost using hop-by-hop routing (either LS of PV) is the critical functionality provided by SDR.

[15] Some additional checks are required when an update is received to insure that a routing loop has not been created. However these checks are much simpler than executing a full SPF algorithm.

an SPF algorithm such as Dijkstra's.

With PV, topology changes only result in the recomputation of routes affected by these changes, which is more efficient than complete recomputation. However, because of the inclusion of full path information with each distance vector, the effect of a topology change may propagate farther than in traditional distance vector algorithms. On the other hand, the number of affected domains will still be smaller with PV than with conventional LS hop-by-hop routing. With PV, only those domains whose routes are affected by the changes have to recompute, while with conventional LS hop-by-hop routing *all* domains must recompute. While it is also possible to employ partial recomputation with LS (i.e., when topology changes, only the affected routes are recomputed), "studies suggest that with a very small number of link changes (perhaps 2) the expected computational complexity of the incremental update exceeds the complete recalculation" ([ANSI 87-150R]).

Support for heterogeneous route selection policies has serious implications for the computational complexity. The *major* problem with supporting heterogeneous route selection policies with LS is the computational complexity of the route selection itself. Specifically, we are not aware of any algorithm with less than exponential time complexity that would be capable of computing routes to all destinations, with LS hop-by-hop routing and heterogeneous route selection policies. In contrast, PV allows each domain to make its route selection autonomously, based only on local policies. Therefore support for dissimilar route selection policies has *no* negative implications for the complexity of route computation in PV. Similarly, providing support for path-sensitive transit policies in LS implies exponential computation, while in PV such support has no impact on the complexity of route computation.

In summary, because NR will rely primarily on precomputation of routes, aggregation is essential to the long-term scalability of the architecture. To the extent aggregation is facilitated with PV, so is reduced computational complexity. While similar arguments may be made for LS, the aggregation capabilities that can be achieved with LS are more problematic because of LS' reliance on consistent topology maps at each node. It is also not clear what additional computational complexity will be associated with aggregation of transit constraints and heterogeneous route selection policies in LS.

3.3.3 Bandwidth Overhead

PV routing updates include fully-expanded information. A complete route for each supported TOS is advertised. In LS, TOS only contributes a factor increase per link advertised, which is much less than the number of complete routes. Although TOSs may be encoded more efficiently with LS than with PV, link state information is flooded to *all* domains, while with PV, routing updates are distributed only to the domains that actually use them.

Therefore, it is difficult to make a general statement about which scheme imposes more bandwidth overhead, all other factors being equal.

Moreover, this is perhaps really not an important difference, since we are more concerned with the number of messages than with the number of bits (because of compression and greater link bandwidth, as well as the increased physical stability of links).

3.4 Aggregation

Forming confederations of domains, for the purpose of consistent, hop-by-hop, LS route computation, requires that domains within a confederation have consistent policies. In addition, LS confederation requires that any lower level entity be a member of only one higher level entity. In general, no intra-confederation information can be made visible outside of a confederation, or else routing loops may occur as a result of using an "inconsistent" map of the network at different domains. Therefore, the use of confederations with hop-by-hop LS is limited because each domain (or confederation) can only be a part of one higher level confederation and only export policies consistent with that confederation (see examples in Section 2.2). These restrictions are likely to impact the scaling capabilities of the architecture quite severely.

In comparison, PV can accommodate different confederation definitions because looping is avoided by the use of full path information. Consistent network maps are not needed at each route server, since routing computation precedes routing information dissemination. Consequently, PV confederations can be nested, disjoint, or overlapping. A domain (or confederation) can export different policies or TOS as part of different confederations, thus providing extreme flexibility that is crucial for the overall scaling and extensibility of the architecture.

In summary, aggregation is essential to achieve acceptable complexity bounds, and flexibility is essential to achieve acceptable aggregation across the global, decentralized internet. PV's strongest advantage stems from its flexibility.

3.5 Policy

The need to allow expression of transit policy constraints on **any** route (i.e., NR routes as well as SDR routes), by itself, can be supported by either LS or PV. However, the associated complexities of supporting transit policy constraints are noticeably higher for LS than for PV. This is due to the need to flood *all* transit policies with LS, where with PV transit policies are controlled via restricted distribution of routing information. The latter *always* imposes less overhead than flooding.

While all of the transit constraints that can be supported with LS can be supported with PV, the reverse is not true. In other words, there are certain transit constraints (e.g. path-sensitive transit constraints) that are easily supported with PV, and are prohibitively expensive (in terms of complexity) to support in LS. For example, it is not clear how NR with LS could support something like ECMA-style policies that are based on hierarchical relations between domains, while support for such policies is straightforward with PV.

As indicated above, support for heterogeneous route selection policies, in view of its computational and storage complexity, is impractical with LS hop-by-hop routing. In contrast, PV can accommodate heterogeneous route selection with little additional overhead.

3.6 Information Hiding

PV has a clear advantage with respect to selective information hiding. LS with hop-by-hop routing hinges on the ability of all domains to have exactly the same information; this clearly contradicts the notion of selective information hiding. That is, the requirement to perform selective information hiding is unsatisfiable with LS hop-by-hop routing.

3.7 Commonality between NR and SDR Components

In [Breslau-Estrin 91] we argued for the use of LS in conjunction with SDR. Therefore there is some preference for using LS with NR. However, there are several reasons why NR and SDR would **not** use exactly the same routing information, even if they did use the same algorithm. Moreover, there are several opportunities for unifying the management (distribution and storage) of routing and forwarding information, even if dissimilar algorithms are used.

In considering the differences between NR and SDR we must address several areas:

1. **Routing information and distribution protocol:** LS for SDR is quite different from the LS in NR. For example, SDR LS need not aggregate domains; to the contrary SDR LS requires detailed information to generate special routes.

   In addition, consistency requirements (essential for NR) are unnecessary for the SDR component. Therefore LS information for the SDR component can be retrieved on-demand, while the NR component must use flooding of topology information.

2. **Route selection algorithm:** It is not clear whether route computation algorithm(s) can be shared between the SDR and NR components, given the difficulty of supporting heterogeneous route selection policies in NR.

3. **Forwarding information:** The use of dissimilar route computation algorithms does not preclude common handling of packet forwarding. Even if LS were used for NR, the requirement would be the same, i.e., that the forwarding agent can determine whether to use a NR precomputed route or an SDR installed route to forward a particular data packet.

In conclusion, using similar algorithms and mechanisms for SDR and NR components does have benefits. However, these benefits do not dominate the other factors as discussed before.

3.8 Summary

Given the performance complexity issues associated with global routing, aggregation of routing information is essential; at the same time we have argued that such aggregation must be flexible. Given the difficulties of supporting LS hop-by-hop routing in the presence of (a) flexible aggregation, (b) heterogeneous route selection policies, and (c) incomplete or inconsistent routing information, we see no alternative but to employ PV for the NR component of our architecture.

Based on the above tradeoffs, our NR component employs a PV architecture, where route computation and installation is done in a distributed fashion by the routers participating in the NR component[16]. The distributed algorithm combines some of the features of link state with those

[16] Packet forwarding along routes produced by the NR component can be accomplished by either source routing or hop-by-hop routing. The latter is the primary choice because it reduces the amount of state in routers (if route setup is employed), or avoids encoding an explicit source route in network layer packets. However, the architecture

of distance vector algorithms; in addition to next hop information, the NR component maintains path attributes for each route (e.g. the list of domains or routing domain confederations that the routing information has traversed so far). The path attributes that are carried along with a route express a variety of routing policies, and make explicit the *entire route* to the destination. With aggregation, this is a superset of the domains that form the path to the destination.

4 Improving Scalability of SDR Routing Information Distribution

By using a hop-by-hop NR component based on PV to complement the source-routing SDR component, we have alleviated the pressure to aggregate SDR *forwarding* information; the large percentage of inter-domain traffic carried, simultaneously, by any particular border router will be forwarded using aggregated NR forwarding information. However, the use of NR does not address the other major scaling problem associated with SDR: that of distributing, storing, and computing over a complete domain-level topology map. In this section we describe promising opportunities for improving the scaling properties of SDR routing information distribution, storage, and computation.

Note that we do not propose to solve this problem in the same way that we solve it for NR. A priori abstraction will not be employed since different domains may require different methods of abstracting the same routing information. For example, if we aggregate routing information of domains that do **not** share the same policy and TOS characteristics (i.e., services), then outside of the aggregate, only those services that are offered by **all** domains in the aggregate will be advertised. In order to locate special routes, SDR only uses aggregates when the component domains (and in turn the aggregate) advertise the required TOS and policy descriptions. When the required TOS or policy characteristics are not offered by an aggregate, full information about the component domains is used to construct a route through those domains that do support the particular characteristics. Consequently, we need some other, more flexible, means of reducing the amount of information distributed and held. We address three issues in turn: distribution of configured topology and policy information, distribution of dynamic status information, and installation of forwarding information.

4.1 Configured Information

Information about the existence of inter-domain links, and policies maintained by domains, changes slowly over time. This is referred to as configured information. In the *current* IDPR specification complete, global, configuration information is kept by a route server in each domain. **Route servers** (RS) are the entities that compute source routes. On startup a RS can download the connectivity database from a neighbor RS; as domains, inter-domain links, or policies change, the changes are flooded to a RS in each domain.

We have not yet specified the exact mechanisms for distributing configured connectivity information for SDR. However, unlike the current IDPR specification, the SDR component will **not** flood all configured information globally. Several alternate methods for organizing and distributing information are under investigation.

does not preclude the use of source routing (or route setup) along the routes computed, but not installed, by the NR component.

Configured information may be regularly distributed via an out-of-band channel, e.g., CD/ROM. In a similar vein, this information could be posted in several well-known locations for retrieval, e.g., via FTP. Between these "major" updates, aggregated collections of changes may be flooded globally. Moreover, limited flooding (e.g. by hop-count) could be used as appropriate to the "importance" of the change; while a policy change in a major backbone may still be flooded globally, a new inter-regional link may be flooded only within those regions, and information about an additional link to a non-transit domain may not be available until the next regularly-scheduled "major" distribution. Changes that are not distributed as they occur will not necessarily be discovered. However, a route server may learn pertinent information by direct query of remote servers, or through error messages resulting from traffic sent along failed routes. Complete global flooding may be avoided by using some combination of these mechanisms.

Even if an initial implementation uses a simple global flood, we must study the problem of structuring connectivity information such that it can be retrieved or distributed in a more selective manner, while still allowing sources to discover desired routes. For example, we imagine RSs requesting *filtered* information from each other. How the RSs should define filters that will get enough information to find special routes, while also effectively limiting the information, is an open question. Again, the question is how to effectively anticipate and describe what information is needed in advance of computing the route.

The essential dilemma is that networks are not organized in a nicely geographical or topologically consistent manner (e.g., it is not effective to ask for all networks going east-west that are within a certain north-south region of the target), hence a source domain does not know what information it needs (or should ask for) until it searches for, and discovers, the actual path. Even with a central database, techniques are needed to structure configuration information so that the potential paths that are most likely to be useful are explored first, thereby reducing the time required for route computation.

One promising approach organizes information using route fragments (partial paths)[17]. Although the number of route fragments grows *faster* than the number of domains (at least $O(N^2)$), we can selectively choose those that will be useful to compute routes. In particular, for each stub domain, fragments are constructed to several well-known backbones[18]. Among its benefits, this approach aggregates domain information in a manner useful for computing source-routes, and provides an index, namely the destination, which facilitates on-demand reference and retrieval of information pertinent to a particular route computation. At this point, it is not clear how route fragments will affect SDR's ability to discover non-hierarchical routes.

4.2 Dynamic Status Information

Assuming a technique for global or partial distribution of configured information, a second issue is whether, and how, to distribute dynamic status information (i.e., whether an inter-domain connection is up or down).

In the current version of IDPR, dynamic status information is flooded globally in addition to configuration information. We propose to distribute status information based strictly on locality.

[17] Route fragments were first suggested by Dave Clark and Noel Chiappa.

[18] Route fragments may be computed by a destination's route server and either made available via information service queries or global flooding. In addition, NR computed routes may be used as SDR route fragments.

First, dynamic information will be advertised within a small hop-count radius. This simple and low-overhead mechanism exploits topological locality. In addition to flooding status updates to nearby nodes, we also want to provide more accurate route information for long distance communications that entails more than a few network hops. **Reverse path update** (RPU) is a mechanism for sending dynamic status information to nodes that are outside the k-hop radius used for updates, but that nevertheless would obtain better service (fewer failed setups) by having access to the dynamic information [Estrin-etal91].

RPU uses the existing active routes (represented by installed setup state or by a cache of the most recent source routes sent via the node in question) as a hint for distribution of event notifications. Instead of reporting only the status of the route being used, RPU reports the status of the domain's other inter-domain connections. If source routing exhibits **route locality**, the source is more likely to use other routes going through the node in question; in any case the overhead of the information about other links will be minimal.

In this way, sources will receive status information from regions of the network through which they maintain active routes, even if those regions are more than k hops away. Using such a scheme, k could be small to maximize efficiency, and RPU could be used to reduce the incidence of failed routes resulting from inaccurate status information. This will be useful if long-path communication exhibits route locality with respect to regions that are closer to the destination (and therefore outside the k hop radius of flooded information). In such situations, flooding information to the source of the long route would be inefficient because k would have to be equal to the length of the route, and in almost all cases, the number of nodes that would use the information decreases significantly with larger k.

4.3 SDR Forwarding

SDR may be built either on top of the network layer supported by the NR component, or in parallel with it. SDR forwarding will be supported via two techniques: loose source-routing and route setup.

The first technique, loose source-routing, would allow the originator of a packet to specify a sequence of domains that the packet should traverse on its path to a destination. Forwarding such a packet within a domain, or even between domains within a confederation, would be left to intra-domain routing. This avoids per-connection state and supports transaction traffic.

The second technique, route setup, will be based on mechanisms developed for IDPR and described in [IDPR90]. It is well suited to conversations that persist significantly longer than a round-trip-time. The setup protocol defines packet formats and the processing of route installation request packets (i.e, setup packets). When a source generates a setup packet, the first border router along the specified source route checks the setup request, and if accepted, installs routing information; this information includes a path ID, the previous and next hops, and whatever other accounting-related information the particular domain requires. The setup packet is passed on to the next BR in the domain-level source route, and the same procedure is carried out[19]. When the setup packet reaches the destination, an accept message is propagated back hop by hop, and each BR en route activates its routing information. Subsequent data packets traveling along the same path to the destination include a path ID in the packet. That path ID is used to locate the

[19] The setup packet may be forwarded optimistically, i.e., before checks are completed, to reduce latency.

appropriate next-hop information for each packet.

5 Conclusions and Future Work

In summary, the proposed architecture combines hop-by-hop path-vector, and source-routed link-state, protocols, and uses each for that which it is best suited: NR uses PV and multiple, flexible, levels of confederations to support efficient routing of generic packets over generic routes; SDR uses LS computation over a database of configured and dynamic information to route special traffic over special routes. In the past, the community has viewed these two as mutually exclusive; to the contrary, they are quite complementary and it is fortunate that we, as a community, have pursued both paths in parallel. Together these two approaches will flexibly and efficiently support TOS and policy routing in very large global internets.

In addition to further evaluation and implementation of the proposed architecture, future work must investigate opportunities for increased unification of the two components of our architecture. We conclude with a brief itemization of several potential areas for unification.

1. **Routing Information Base**:
   Perhaps a single RIB could be shared by both NR and SDR. NR routes can be represented as a directed graph labeled with flags (on the nodes or links) corresponding to the generic transit constraints. SDR requires that this graph be augmented by links with non-generic policies that have been discovered and maintained for computing special routes; in addition, special policy flags may be added to links already maintained by the NR component.

2. **Information Distribution**:
   The NR path vectors could include address(es) of repositories for SDR-update information for each AD (or confederation) to assist the SDR component in retrieving selective information on demand. For domains with minimal policies, where the space required for policy information is smaller than the space required for a repository address (e.g., if the policies for the domain listed are all wildcard), the NR path vectors could include a flag to that effect.

3. **Packet Forwarding**:
   We should consider replacing the current IDPR-style network layer (which contains a global path identifier used in forwarding data packets to the next policy gateway on an IDPR route) with a standard header (e.g., IP or CLNP), augmented with some option fields. This would unify the packet header parsing and forwarding functions for SDR and NR, and possibly eliminate some encapsulation overhead.

4. **Reachability Information**:
   Currently IDRP distributes **network** reachability information within updates, whereas IDPR only distributes **domain** reachability information. IDPR uses a domain name service function to map network numbers to domain numbers; the latter is needed to make the routing decision. We should consider obtaining the network reachability and domain information in a unified manner.

6 Acknowledgments

We would like to thank Hans-Werner Braun (San Diego Supercomputer Center), Lee Breslau (USC), Scott Brim (Cornell University), Tony Li (Cisco Systems), Doug Montgomery (NIST), Tassos Nakassis (NIST), Martha Steenstrup (BBN), and Daniel Zappala (USC) for their comments on a previous draft.

References

[ANSI 87-150R] "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol", ANSI X3S3.3/87-150R.

[BGP91] Lougheed, K., Rekhter, Y., "A Border Gateway Protocol 3 (BGP-3)", Internet RFC 1267, SRI Network Information Center, October 1991.

[Breslau-Estrin 91] Breslau, L., Estrin, D., "Design and Evaluation of Inter-Domain Policy Routing Protocols", To appear in Journal of Internetworking Research and Experience, 1991. (Earlier version appeared in ACM Sigcomm 1990.)

[Clark 90] Clark, D., "Policy Routing in Internetworks", Journal of Internetworking Research and Experience, Vol. 1, pp. 35-52, 1990.

[Dijkstra 59] Dijkstra, E., *A Note on Two Problems in Connection with Graphs*, Numer. Math., Vol. 1, 1959, pp. 269-271.

[ECMA89] "Inter-Domain Intermediate Systems Routing", Draft Technical Report ECMA TR/ISR, ECMA/TC32-TG 10/89/56, May 1989.

[EGP] E. Rosen, "Exterior Gateway Protocol (EGP)", Internet RFC 827, SRI Network Information Center, October 1982.

[Estrin 89] Estrin, D., "Policy Requirements for Inter Administrative Domain Routing", Internet RFC 1125, SRI Network Information Center, November 1989.

[Estrin-etal91] Estrin, D., Breslau, L., and Zhang, L., "Protocol Mechanisms for Adaptive Routing in Global Multimedia Internets", University of Southern California, Computer Science Department Technical Report, CS-SYS-91-04, November 1991.

[Hedrick 88] Charles Hedrick, "Routing Information Protocol", Internet RFC 1058, SRI Network Information Center, June 1988.

[IDPR90] Steenstrup, M., "Inter-Domain Policy Routing Protocol Specification and Usage: Version 1", Internet Draft, February 1991.

[IDRP91] "Intermediate System to Intermediate System Inter-domain Routeing Exchange Protocol", ISO/IEC/ JTC1/SC6 CD10747.

[ISIS10589] "Information Processing Systems - Telecommunications and Information Exchange between Systems - Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the protocol for providing the Connectionless-mode Network Service (ISO 8473)", ISO/IEC 10589.

[Jaffee 82] Jaffee, J., Moss, F., "A Responsive Distributed Routing Algorithm for Computer Networks", IEEE Transactions on Communications, July 1982.

[Little 89] Little, M., "Goals and Functional Requirements for Inter-Autonomous System Routing", Internet RFC 1126, SRI Network Information Center, October 1989.

[Oran 89] Oran, D., "Expert's Paper: The Relationship between Addressing and Routeing", ISO/JTC1/SC6/WG2, 1989.

[OSPF] J. Moy, "The Open Shortest Path First (OSPF) Specification", Internet RFC 1131, SRI Network Information Center, October 1989.

[Rekhter 91] Rekhter, Y., "IDRP protocol analysis: storage complexity", IBM Research Report RC17298(#76515), October 1991.

[Shin87] Shin, K., Chen, M., "Performance Analysis of Distributed Routing Strategies Free of Ping-Pong-Type Looping", IEEE Transactions on Computers, Feb. 1987.

[ST2-90] Topolcic, C., "Experimental Internet Stream Protocol, version 2 (ST II)", Internet RFC 1190, October 1990.

[Zaumen 91] Zaumen, W., Garcia-Luna-Aceves, J., "Dynamics of Link State and Loop-free Distance-Vector Routing Algorithms", ACM Sigcomm '91, Zurich, Switzerland, Sept. 1991.

[Zhang 91] Zhang, L., "Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks", ACM Transactions on Computer Systems, Vol. 9, No. 2, May 1991.