NIRA: A New Inter-Domain Routing Architecture

Xiaowei Yang, *Member, IEEE*, David Clark, *Fellow, IEEE*, and Arthur W. Berger

***Abstract*—In today's Internet, users can choose their local Internet service providers (ISPs), but once their packets have entered the network, they have little control over the overall routes their packets take. Giving a user the ability to choose between provider-level routes has the potential of fostering ISP competition to offer enhanced service and improving end-to-end performance and reliability. This paper presents the design and evaluation of a new Internet routing architecture (NIRA) that gives a user the ability to choose the sequence of providers his packets take. NIRA addresses a broad range of issues, including practical provider compensation, scalable route discovery, efficient route representation, fast route fail-over, and security. NIRA supports user choice without running a global link-state routing protocol. It breaks an end-to-end route into a sender part and a receiver part and uses address assignment to represent each part. A user can specify a route with only a source and a destination address, and switch routes by switching addresses. We evaluate NIRA using a combination of network measurement, simulation, and analysis. Our evaluation shows that NIRA supports user choice with low overhead.**

***Index Terms*—Inter-domain routing, Internet architecture, routing, source routing, user-controlled routing.**

I. Introduction

THIS paper is concerned with the question of how Internet traffic is routed at the domain level [at the level of the autonomous system (AS)][1] as it travels from source to destination. Today, users can pick their own Internet service providers (ISPs), but once their packets have entered the network, the users have no control over the overall routes their packets take. ISPs make business decisions to interconnect, and technically the BGP routing protocol [48] is used to select the specific route a packet follows. Each domain makes local decisions that determine what the next hop (at the domain level) will be, but the user cannot exercise any control at this level. In this context, a *user* could be a human user, or a program.

In [12], Clark *et al.* argued that a better alternative would be to give the user more control over routing at the domain level. User-controlled routes are a key factor in maintaining the competitiveness of the ISP marketplace [12]. An analogy can be seen in the telephone system, which allows the user to pick his long-distance provider separately from his (usually monopolist) local provider. Allowing the user to select his long-distance provider has created the market for competitive long-distance service, and driven the price to a small fraction of its pre-competition starting point. For the consumer, especially the residential broadband consumer, there is likely to be a very small number of competitive local ISPs offering service [8], [12], [15].[2] With cable competing only with DSL, the market is a duopoly at best (at the facilities level) and often a monopoly in practice. If users cannot choose their backbone ISPs separately from their local ISPs, local providers can then control the selection of backbone providers and capture the market power of consumers. This will reduce the competitive pressures on the backbone providers and lead to a vertically integrated ISP market. The recent merger of SBC and AT&T (to use the old names) [5], and Verizon and MCI [63] only add to this concern. In the worst case, SBC sends all its traffic to the AT&T backbone, and Verizon sends its traffic to MCI. We have the re-emergence of market power in the backbone market. Conversely, when users can control the sequence of providers his packets take, the power of user choice fosters competition. In a competitive market, ISPs that are more efficient attract more users, which creates a positive loop for them to further advance technologies and to improve efficiency. In the long term, competition disciplines the market, drives innovation, and lowers the costs to provide services [12], [66].

Moreover, recent studies on overlay networks and multihoming show that letting the user choose routes also brings technical benefits. The default routing path chosen by BGP [48] may not be the best in terms of performance, reliability, or cost. End hosts on an overlay network often find alternative Internet paths with lower latencies, lower losses, or higher reliability than the default routes chosen by BGP [3], [27], [51]. For instance, Detour found that for almost 80% of the default paths, an alternative route offers a lower loss rate [51]. Similarly, recent studies also show that multihomed sites can improve path quality and reduce monetary cost by intelligently choosing their upstream providers [2], [25].

The prevalence of these alternative paths suggests that giving the user the ability to choose routes can lead to improved performance, reliability, or user satisfaction. Only users know whether a path works for their applications or not. A user may choose a path that has a high throughput and a low latency for playing online games, even if the path may cost more. In contrast, a user may prefer a low cost path for peer-to-peer file downloads. Furthermore, letting the user choose routes also improves reliability. A user can use multipath routing to improve the reliability for mission-critical applications, such as 911 calls, or quickly switch to an alternative route if the default routing path fails.

Manuscript received October 24, 2005; revised April 4, 2006; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor O. Bonaventure.

X. Yang is with the University of California at Irvine, Irvine, CA 92697 USA (e-mail: xwy@uci.edu).

D. Clark is with MIT CSAIL, Cambridge, MA 02139 USA.

A. W. Berger is with MIT CSAIL and Akamai Technologies, Cambridge, MA 02139 USA.

Digital Object Identifier 10.1109/TNET.2007.893888

[1]In this paper, an AS is also referred to as a *domain*. An AS that provides transit service is sometimes called a *provider*.

[2]Although the Telecommunications Act of 1996 requires that incumbent local exchange carriers (iLEC, the local phone companies that own the wires) to provide open access of their networks to competitors at reasonable costs, the implementation of the Act has been difficult [15]. The iLECs have disincentives to open up the local market. To block market entries, they may create obstacles such as significant nonrecurring costs in the leasing of their network elements. As a result, the DSL market shares of the competitive local exchange carriers keep decreasing [8]. In the mean time, facility-level competition has not taken place widely, due to the high capital requirements for market entry [8], [15].

As a starting point, this paper presents the design and evaluation of a new Internet routing architecture (NIRA), an architecture that gives a user the ability to choose domain-level routes. A domain-level route refers to the sequence of domains a packet traverses, as opposed to a router-level route. We focus on domain-level rather than router-level routes because choice at this level fosters competition [12], and because providing choice at this level is more tractable than providing choice at the router level. Our design leaves it as an option to a domain to decide whether to offer router-level choice to users. In the rest of our paper, without explicit explanation, a *route* refers to a domain-level route.

A key contribution of our work is that it addresses a broad range of problems that arise from supporting user choice, including practical provider compensation, scalable route discovery, efficient route representation, fast route fail-over, and security. To the best of our knowledge, our work is the first to address all of those issues.

Our design shows that it is technically feasible to support user choice in the Internet. NIRA allows each individual user to choose its provider-level routes and allows practical ISP payment schemes: a user is restricted to choose from the set of providers that he pays for directly or indirectly. NIRA supports user choice without expanding packet headers with a source routing option in the common case. We break an end-to-end route into three parts: a sender part, a receiver part, and a *Core* part (Section II-H), and use an address to represent a sender part or a receiver part. NIRA provides mechanisms for a user to discover his part of a route. A user may use a NRLS to discover a destination's part of a route. The user can then use a source and a destination address to represent an end-to-end route, and switch routes by switching addresses.

We evaluate our design using a combination of measurement, simulation, and analysis. We implemented in ns-2 [42] a prototype of NIRA and used domain-level topologies inferred from BGP tables for simulation. We use both simulation and an analytical model to evaluate how well NIRA scales and how well NIRA works under various route failure conditions. Our evaluation shows that NIRA supports user choice with low overhead.

We note that this paper focuses on addressing the technical problems to support user choice. The companion business or technical issues that are caused by user choice are outside the scope of this paper. There is a question on whether ISPs would permit user choice. But this is a separate policy issue. If in the future ISPs decide to support user choice, or regulators mandate them to do so, our design will be a viable technical solution. We are also aware that letting users choose routes independently may lead to performance problems such as temporary route oscillation [35] or suboptimal routes [49]. Despite these potential problems, recent research suggests that in the Internet environment, user-selected routes are close to optimal [45] and route oscillations happen under rare conditions and can be quickly stopped by user backing off [35]. It is our future work to look into these performance problems, and we are optimistic that they can be mitigated.

The rest of the paper is organized as follows. Section II discusses the design rational of NIRA. In Sections III and IV, we describe the key design components of NIRA in detail. We evaluate our design in Section V, and compare our work with related work in Section VII. Section VIII concludes our work.

II. Design Overview

In this section, we motivate the key components of NIRA. The overall goal of NIRA is to support user choice. To achieve this goal, the design of NIRA must satisfy several implied requirements.

First, to be practical, we require that NIRA must recognize the need for payment and allow practical payment schemes. In a commercialized Internet that involves multiple parties, an architecture design must respect users' economic interests as well as those of ISPs. A technical design will not be feasible if there is no practical mechanism for users to pay ISPs for their service. So in our design, we do not require new payment schemes such as micro-payments. Second, we require that our design be highly efficient and scale well. A future Internet must support a wide range of devices, ranging from resource-constrained sensor nodes to super computers, and a wide range of communication technologies, ranging from low bandwidth wireless connections to fiber optics. An efficient and scalable design would accommodate this variety and allow resource-constrained end systems to take advantage of the design without additional infrastructure support. Third, we require that our design be able to accommodate future evolution and enhancement. With the advance of technologies and the evolution of the society, the design assumptions of today may well change. For example, today we assume small devices such as sensor nodes have limited resources. In another ten years, they may have as abundant resources as today's desktop computers. Our design must allow for future evolution and enhancement. Finally, the design must be incrementally deployable.

A. Design Rationale

To support user-controlled routes, our design must address the following questions. 1) How does a user discover a failure-free route? 2) How is a route encoded in a packet header? 3) How do users pay for ISPs' service?

Observe that the design problems we face have little dependency between each other. In our design, we make them into separate design modules. This reduces the amount of architectural constraints to minimal, and makes the design readily adapt to future evolution and enhancement. For each design module, we analyze whether a global consistent mechanism is needed to implement that module, or multiple mechanisms can be developed and deployed locally. We design mechanisms to implement the basic functionality of each module, and make it clear whether the functionality can be enhanced by other local mechanisms.

We constrain users to choose from the set of providers they agree to pay for by contractual agreements to allow for practical payment schemes. Our design assumes bilateral contracts, as used in the present Internet [30], [31], but we assume ISPs may use different pricing models [67]. Bilateral contracts are proven to work and are much simpler than establishing contracts over an exponentially large set of groups of ISPs. In a bilateral contract, two ISPs negotiate business relationships to interconnect. The common business relationships include customer-provider and peer-to-peer relationships. In a customer-provider relationship, a customer pays a provider, and the provider provides transit service between the customer and its neighbors; in a peer-to-peer relationship, a peer provides transit service between the

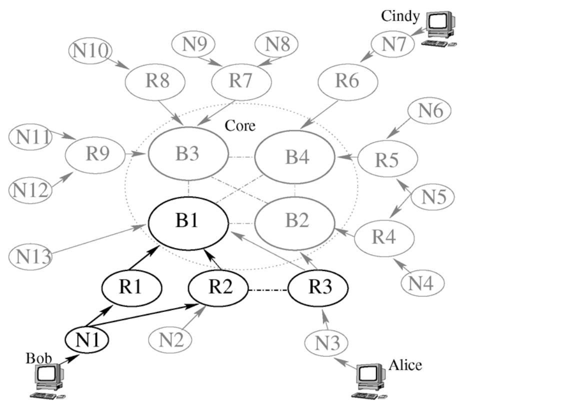


Fig. 1. Dark lines draw the user Bob's up-graph. In this and all other figures, an arrowed line represents a provider-customer connection, with the arrow ends at the provider. A dashed line represents a peering connection.

other peer and its customers. Our design also works with other contractual relationships (Section VI). Next, we describe each design module in more detail.

B. Route Discovery

Before we describe how a user discovers a failure-free route, we first define a few terms. We refer to a provider that does not purchase transit service from other providers as a tier-1 provider. We define the region of the Internet where tier-1 providers interconnect as the *Core* of the Internet. We discuss a more general definition about the *Core* in Section II-H.

To formulate a valid route, a user needs to know which routes he can use and whether they are failure free. Our design provides a basic mechanism for users to discover routes and route failures, and allows users to use any general mechanism to enhance the basic mechanisms. To allow practical payment schemes, our design exposes to a user the region of the network that consists of his providers, his providers' providers, and so on until the providers in the *Core*. In addition, the region also includes the peering connections of the providers outside the *Core*. This region of the network is the user's "access network" to the rest of the Internet. We refer to this region as a user's "up-graph." Fig. 1 shows a sample network, where the dark lines depict the user Bob's up-graph. In this paper, we abstract each domain as having one router and the multiple links between two domains as one domain-level link. Without specific mentioning, the term *link* in this paper refers to a domain-level link. More realistic situations are considered in [67].

We design a topology information propagation protocol (TIPP) to let a user discover his up-graph. TIPP has two components: a path-vector component to distribute the set of provider-level routes in a user's up-graph, and a policy-based link state component to inform a user of the dynamic network conditions. The path-vector component informs a user of his direct and indirect providers and the transit routes formed by those providers. A tier-1 provider advertises itself to its customers, and the customers append themselves and further advertise the routes to their customers. Different from a path-vector routing protocol such as BGP, the path-vector component of TIPP does not select paths. In the example of Fig. 1, a user Bob learns from TIPP that he can use two routes to reach the *Core* : $N_1 \to R_1 \to B_1$ and $N_1 \to R_2 \to B_1$. Each domain on the route is his direct or indirect provider. The link-state component of TIPP informs a user of the dynamics of the network. Unlike a global link-state routing protocol, TIPP's link-state messages can be configured to propagate within a provider hierarchy, instead of globally. A domain can control the scope within which a link state message is propagated (**scope enforcement**) as well as to which neighbor to expose an adjacent domain-level link (**information hiding**). With this policy, a user Bob only learns the domain-level links on his up-graph, which includes $N_1 \to R_1$, $N_1 \to R_2$, $R_1 \to B_1$, $R_2 \to B_1$, and $R_2 \to R_3$. This flexibility allows the design to be efficient and to be readily used by resource-constrained devices such as PDAs or aged PCs. However, if in the future, efficiency (memory, bandwidth, and computation power) is not a concern even for small or low-end devices, the link-state component of TIPP can be configured to propagate link-state messages globally. A user with a more complete network topology can avoid more dynamic failures in the network. We describe TIPP in more detail in Section III.

In NIRA, a sender can obtain an end-to-end route to a destination by combining routes in his up-graph and those in the destination's up-graph in reverse order. A typical domain-level route is said to be "valley-free" [24]. That is, a route has an "uphill" segment consisting of a sequence of the sender's providers, and a "downhill" segment consisting of a sequence of the receiver's providers. A sender's up-graph contains the uphill segment of a route, and the destination's up-graph contains the downhill part. A sender learns an uphill segment from TIPP, and may obtain the destination half of a route from a lookup service, the name-to-route lookup service (NRLS), which we describe in Section II-D.

C. Efficient Route Representation

Once a sender formulates an end-to-end route, he needs to encode it in a packet header. We expect that a source route will be encoded with a uniform global format. This is to ensure global communication. There are several candidates for route encoding. For instance, one encoding scheme is to use a sequence of addresses or domain identifiers such as AS numbers. The advantage of this scheme is its flexibility. It can express any route. The drawback is the variable-length header overhead.

In NIRA, we use a provider-rooted hierarchical address to encode a route segment that connects a user to a *Core* provider. This scheme has a number of advantages. First, it is efficient in the common case. With this scheme, a user can use a source and a destination address to compactly represent a valley-free route, and switch routes by switching addresses. Meanwhile, with this representation scheme, both the source address and the destination address are used for forwarding. This effectively limits source address spoofing, because a router may not find a forwarding entry for a packet with an arbitrary source address, and will instead drop the packet.

In the provider-rooted hierarchical addressing scheme, a provider in the *Core* obtains a globally unique address prefix. The provider then allocates nonoverlapping subdivisions of the address prefix to each of its customers. Each customer will recursively allocate nonoverlapping subdivisions of the address prefix to any customer it might have.

The hierarchical addressing scheme requires that an address be long enough to encode the provider hierarchy. This requirement could be met either with a fixed-length address with a large

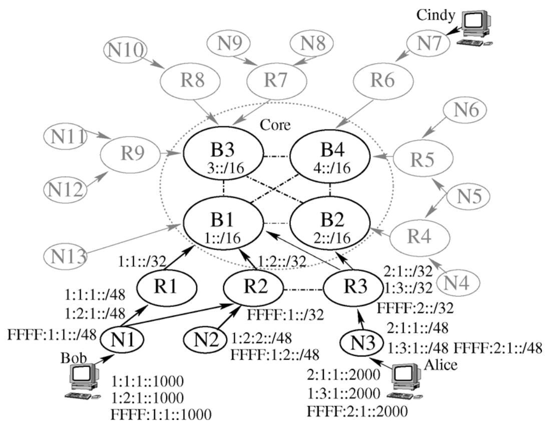


Fig. 2. Example of the strict provider-rooted hierarchical addressing. For clarity, we only show address allocation for the dark region.

address space, or with a variable length address. For the sake of concreteness, we use IPv6 address [14], which is 128-bit long. Using IPv6 addresses has the benefit that NIRA's packet header could be the same as the IPv6 header. We use the first 96-bit of an address as an inter-domain address prefix that is hierarchically allocated from a provider to a customer; the remaining 32-bit is an intra-domain address that identifies a network location inside a domain. If allocated prudently, 96 bits could address up to 2^{96} domains. In any foreseeable future, the number of domains in the Internet is unlikely to be anywhere close to that number. So we think that an IPv6 address is a reasonable choice. This design choice also allows NIRA to reuse the deployment effort of IPv6.

We note that the choice of route encoding is independent of other design modules of NIRA. Alternatively, we could use a sequence of IPv4 addresses to represent a domain-level route. This will make NIRA compatible with the current IPv4 architecture. We chose to use IPv6 addresses because we consider it advantageous to use a source and a destination address to represent a common type of route. Even if IPv6 does not completely replace IPv4 in the future, NIRA can be adapted to use a sequence of IPv4 addresses to represent the routes in the part of the network that uses IPv4 addresses. Users can still learn those routes from TIPP.

Fig. 2 shows an example of NIRA's address scheme. In this and all other examples, we represent an address and an address prefix using the IPv6 notation as described in [14], where "::" represents a variable number of contiguous zeros, and ":" is a separator that separates every 16-bit piece. An address prefix is represented in the format *address/prefix length*. In this example, the tier-1 providers B_1 obtains a globally unique address prefix 1::/16. Non-overlapping subdivisions of the prefix are allocated downwards the provider-tree rooted at B_1. B_1's customer R_1, R_2, and R_3 each gets a subdivision of 1::/16: 1:1::/32, 1:2::/32, 1:3::/32, and recursively allocate the subdivisions of the prefixes to their customers: N_1, N_2, and N_3. A user in N_1, Bob, gets two hierarchical addresses allocated from the tier-1 providers: 1:1:1::1000 and 1:2:1::1000.

We further extend the provider-rooted hierarchical address allocation mechanism and assign a noncore-visible address space to a domain-level peering link outside the *Core*. Each peer obtains an address prefix from the noncore-visible address space. A provider that has a peering address prefix will also recursively allocate the subdivisions of the address prefix to its customers. The peering address is *noncore-visible* in the sense that they are not propagated into the *Core*.

For instance, in Fig. 2, a noncore-visible peering prefix FFFF::/16 is allocated to the peering link between R_2 and R_3. R_2 gets the address prefix FFFF:1::/32, and R_3 gets the address prefix FFFF:2::/32. Recursively, $R2$ and R_3 allocate subdivisions of the address prefixes to their customers N_1, N_2, and N_3. Domain N_1 gets a prefix FFFF:1:1::/48, and the user *Bob* in domain N_1 gets a noncore-visible peering address FFFF:1:1::1000.

With this addressing scheme, a user can use a source and a destination address to represent a valley-free route. The sequence of the providers that allocate the source address maps to the uphill route segment; similarly, the sequence of the providers that allocate the destination address maps to the downhill segment. A user Bob may use the source address 1:1:1::1000 and the destination address 1:3:1::2000 to represent the route $N_1 \rightarrow R_1 \rightarrow B_1 \rightarrow R_3 \rightarrow N_3$.

With the hierarchical address allocation scheme, one can infer the provider-customer relationship from the address prefixes a domain has. We do not think it will become a deployment obstacle because this relationship can be largely inferred today [24], [57].

D. Bootstrap a Communication

To bootstrap a communication, a user also needs to know what routes a destination is allowed to use. Similar to bootstrapping a communication in today's Internet, in which a user obtains the IP address of a destination, a user can obtain this information via multiple approaches, e.g., a directory lookup, a web search, or out of band communication. In our design, we propose an infrastructure service, the NRLS to map the name of a destination to the route segments the destination is allowed to use. NRLS is a separate design module. Our design does not constrain the form of the name space or the implementation of the NRLS service. The namespace can be hierarchical like the domain name service (DNS) [39], or flat as proposed in [7], or take any other form that is used in the future Internet. Similarly, the implementation can either be hierarchical, or flat [13], or a mixture [28].

If an end system wants to be reached by other hosts, it stores its route segments together with its preference at an NRLS server. With the encoding scheme described in Section II-C, the routes a destination can use are encoded as addresses. When a user intends to communicate with a destination, the user queries the NRLS to retrieve the route information of the destination, similar to querying a DNS server in today's Internet. Combining his route information with that of the destination, a user is able to choose a source and a destination address to reach the destination. Two users may exchange subsequent packets to negotiate a better route.

If the domain-level topology changes, a user's provider-level routes may change. TIPP will notify a user of these changes. When a server's provider-level routes or its route preference changes, it needs to update its NRLS record. This process is similar to dynamic DNS [64] updates or other directory service updates. NRLS can use any mechanism that secures DNS updates (or other directory service updates) [29], [65] to provide security to NRLS updates. We do not think NRLS updates would cause any scaling problem for two reasons. First, the Internet topology (at the domain level) changes at a low frequency [10]. Only those changes would affect a user's route segments. Second, static topology changes (excluding temporary failures) are caused by the changes in business relationships. These changes happen in a controlled manner. Network administrators and users can deploy creative scheduling algorithms to reduce the service disruption and the NRLS server update overhead. A grace period may be granted before a provider terminates its service so that a user has sufficient time to update his route information. Randomized algorithms may be used to prevent users from simultaneously updating their route information, and standard load-balancing techniques can be used to shed load from one NRLS server to multiple machines.

Like DNS or any other directory service, an NRLS resolver needs to be hard-coded with the route segments of the root NRLS servers. When the route segments of the root servers change, the hard-coded information need to be updated. The process of updating root server information can be inconvenient. One approach to alleviate this problem is to place the root NRLS servers at the tier-1 providers. A server inside a tier-1 provider only has one route segment that consists of the tier-1 provider. The likelihood that the route would change is significantly reduced.

E. Handling Route Failures

The route a user chooses to use may suffer dynamic failures. The basic mechanism we provide for route failure discovery is a combination of proactive and reactive notification. TIPP proactively notifies a user of the working conditions of routes in the user's up-graph. As TIPP messages may not propagate globally, a user in general does not know the working conditions of routes on a destination's up-graph. A user relies on reactive mechanisms, such as a timeout or a router feedback, to discover route failures.

Our design requires that if a router detects that a route specified in a packet header is unavailable, the router try its best to send a rate-limited ICMP message to inform the original sender. Such an ICMP message may include reasons why the route is unusable. In cases where a router is unable to send a failure notification, e.g., a router is overloaded, users should use timeout to detect route failures. When a user receives a reactive notification, as he knows his addresses, his up-graph, and the addresses of the destination, he could switch to an alternative route. The fail-over time is on the order of a round trip time in the case of a router feedback, and depends on the timeout parameters in the case of a timeout.

The combination of proactive and reactive notification reduces the amount of dynamic routing information a user needs to maintain. However, reactive notification may increase the connection setup time when user selected routes suffer from failures. Users should cache recently used routes and avoid using unavailable routes for a new connection. We expect that the amortized connection set up time will be reduced with such caching. Section V provides an analytic model to estimate the average connection set up time in the presence of failures.

In our design, failures that trigger reactive notifications are those that result in inter-domain disconnections. Intra-domain failures should always be recovered using local repair mechanisms for rapid fail-over [52], [62]. For inter-domain failures, it is best for the user to decide on an alternative route, because a user has expressed his domain-level route choice in a packet header, and a router does not know the user's route preference. The dynamics of route selection may not be high since the domain-level routes may not be so fragile to single link failures.

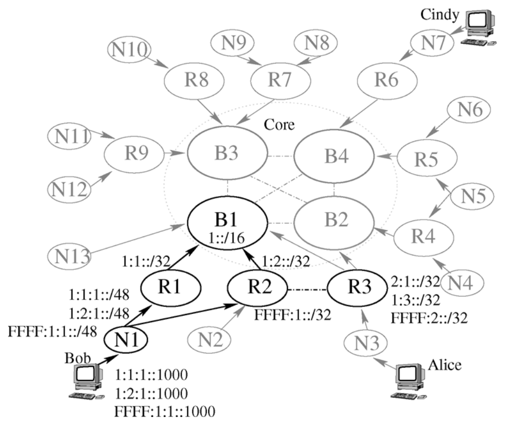


Fig. 3. What Bob learns from TIPP is shown in dark lines.

In addition to the combination of TIPP messages, router notifications, and timeouts, users may use any general mechanism to discover route failures. For instance, a local provider may offer a route monitoring service to its users. The provider may run a server that actively probes the dynamic attributes of popular routes and provides timely information on route conditions to its customers. In addition, popular servers may choose to include dynamic topology information in their NRLS records and update the information at a rate they could afford. When a user retrieves a server's route information, he may already know which addresses of the server are unreachable, thereby saving the connection setup time.

F. How Users May Choose Routes

After a user is equipped with the ability to discover routes and to handle route failures, he is able to choose routes. We present a concrete example to illustrate how a user may choose routes. Suppose a user Bob in Fig. 2 wants to communicate with Alice. The following steps may happen.

1) Bob learns his addresses: 1:1:1::1000, 1:2:1::1000, FFFF:1:1::1000, his up-graph, and the dynamic failure information on his up-graph from TIPP, as shown in Fig. 3. Note that the up-graph tells Bob the provider-level routes his addresses map to.
2) Bob queries NRLS servers to obtain Alice's addresses: 2:1:1::2000, 1:3:1::2000, and FFFF:2:1::2000, and Alice's preference over these addresses.
3) Combining his addresses and Alice's addresses, Bob knows that he has five routes to reach Alice. Four of them come from the combination of his global addresses and Alice's global addresses. One of them comes from the combination of the noncore-visible peering addresses.
4) Suppose Alice prefers the peering address FFFF:2:1::2000 to other global addresses. From his up-graph, Bob knows that he has an address FFFF:1:1::1000 allocated from the same peering link, and the route segment in his up-graph $N_1 \rightarrow R_2 \rightarrow R_3$ is in good condition. Bob then sends a packet to Alice using a source address FFFF:1:1::1000 and a destination address FFFF:2:1::2000.

5) After the packet reaches Alice, Bob, and Alice may exchange subsequent packets to negotiate a better route. If Bob detects a failure from a router feedback or a timeout, he can switch to an alternative route by switching addresses.

The default usage model of NIRA is that a software agent running on a user's computer selects routes based on the user's preference. The agent can either learn a user's preference from static configured policies, or from an adaptive learning approach by observing a user's behavior, as described in [17], [37]. For instance, if an ISP offers a voice over IP expedited forwarding service, a user may configure the agent to choose the ISP when he runs voice over IP applications.

In addition, NIRA supports multiple usage models. Choice is not restricted to be made only by end users. In situations where a domain does not want to give route choice to its users, for instance, a company network might not want its employees to select routes according to their preferences, the domain does not need to propagate TIPP messages to those users. Instead, the domain could have a route server to select routes for its users. If a domain uses a NAT [54] box to isolate hosts in the domain from the rest of the Internet, the NAT box will select routes on behalf of the hosts in the domain. Alternatively, a domain could have its border routers act as name-to-route translation boxes, as described in the IPNL [23] architecture. A user sends all his packets with the destination's domain name. A border router selects routes on behalf of users. When a border router receives the first packet to a destination from a user, the router does an NRLS query, caches the results, selects a route for the user, attaches the route representation to the packet, and sends the packet. For subsequent packets, the router can translate the destination name into a previously selected route representation using cached results.

G. Forwarding

In NIRA, a packet is first forwarded along the sequence of domains that allocate the source address, and then forwarded along the sequence of domains that allocate the destination address. The path the packet takes from the top-level provider that allocates the source address to the one that allocates the destination address is chosen by the top-level providers inside the *Core*.

For instance, in Fig. 2, if a user Bob chooses a source address 1:1:1::1000 (allocated from $B_1 \rightarrow R_1 \rightarrow N_1$) and a destination address 2:1:1::2000 (allocated from $B_2 \rightarrow R_3 \rightarrow N_3$) to send a packet, the packet will be forwarded along the route $N_1 \rightarrow R_1 \rightarrow B_1 \rightsquigarrow B_2 \rightarrow R_3 \rightarrow N_3$, where the symbol $\rightsquigarrow$ denotes that the top-level providers in the *Core* decide how to forward the packet from B_1 to B_2.

Routers must have correct forwarding state in order to forward packets along the specified routes. In NIRA, domains run TIPP to establish the forwarding state within a provider hierarchy, and top-level providers may run an inter-domain routing protocol such as BGP to determine the forwarding state inside the *Core*. For instance, TIPP informs a router in N_1 that the address prefix 1:1:1::/48 is allocated from R_1. Thus, packets with a source address 1:1:1::1000 will be forwarded to R_1. Section IV discusses packet forwarding in detail.

We made the design choice that the path a packet takes within the *Core* is chosen by providers in the *Core* for two practical reasons. First, this will facilitate the deployment of NIRA, because in the present Internet, routes are chosen by ISPs running BGP, instead of by users. We expect that the migration from the present routing architecture to NIRA gradually happens from the edge towards the center. Local providers and regional providers can incrementally deploy NIRA to let users choose the backbone providers. Second, letting a user choose routes in the *Core* may not bring much flexibility in choice, but will expose to the user the dense topology of the *Core*. For instance, tier-1 ISPs of the Internet form a clique, with any tier-1 provider having a peering agreement with all other tier-1 providers, but only the direct peering connection is the policy-allowed route between any two tier-1 providers. Thus, we made the design choice not to let users choose routes inside the *Core*.

We note that the strict provider-rooted hierarchical address allocation scheme makes the *Core* a scalable routing region. With this addressing scheme, inside the *Core*, a provider in the *Core* only needs to announce one aggregated address prefix to other providers in the *Core*. The number of routing table entries in the *Core* scales with the number of providers in the *Core*. Because the ISP market is a high-barrier market, financial factors will limit the market entries. Therefore, routing in the *Core* is unlikely to run into the scaling problems faced by today's Internet.

H. Definition of the Core

The concept of the *Core* in NIRA is more general than in today's Internet. At the minimum, the *Core* in NIRA will include the tier-1 providers that do not purchase transit service from other providers. A nontier-1 ISP can decide whether to join the *Core* or not. To join the *Core*, it needs to obtain a globally unique address prefix, connect to those providers with globally unique address prefixes, and convince those providers to announce its own prefix into the *Core*. To not join the *Core*, the ISP accepts address allocations from the providers it connects to.

We expect that the *Core* will be a dynamic structure that is shaped by the market force as NIRA deploys. If users welcome route choice, then a nontier-1 provider has the incentive to stay outside the *Core* and to let users choose among the providers it connects to. Otherwise, the provider may choose to stay inside the *Core*. For instance, in Fig. 2, if users welcome choice, then the provider R_3 may decide to stay outside the *Core*, letting users choose between B_1 and B_2. ISPs could make their decisions based on the cost to obtain a globally unique address prefix, the cost to get that prefix announced to other providers in the *Core*, and the benefits and drawbacks of doing so. It is worth noting that even in the worse case, where every provider has decided to join the *Core*, with NIRA, a multihomed edge domain will not poke a hole in the global routing tables. Thus, the growth of the global routing state is still limited by the growth of the *Core*, instead of the growth of the Internet. We think this rate of growth will scale well.

In the next two sections, we describe two of the key components of NIRA: TIPP and the forwarding algorithm in detail.

III. Topology Information Propagation Protocol (TIPP)

TIPP is an inter-domain protocol that runs between border routers of domains. It operates outside the *Core* of the Internet, and has three functionalities.

1) TIPP automatically propagates the mapping between addresses and the provider-level routes in users' up-graphs to users.
2) TIPP propagates up-graphs to users.
3) TIPP helps routers to establish inter-domain forwarding entries. (We describe this in the next section.)

Fig. 3 depicts what a user, Bob, learns from TIPP. Bob learns the address prefixes allocated to his domain N_1, his up-graph, and the mapping between the address prefixes and the uphill route segments on his up-graph.

We design TIPP to use separate protocol messages to propagate address information and topology information. This design choice minimizes the coupling between the two functionalities. The part of TIPP that propagates address information is straightforward: if a domain receives an address prefix allocation (or withdrawal) message from a provider, it sends a message to allocate (or withdraw) a subdivision of the prefix to a customer. Each domain also appends its own domain identifier in the message to provide the mapping between an address and a route segment. This part is similar to a path-vector protocol.

The part of TIPP that distributes topology information is a policy-controlled link-state protocol. A domain can control what to distribute to a neighbor at the granularity of a link. A domain has two types of control: **scope enforcement** and **information hiding**. Scope enforcement ensures the scalability of TIPP. A domain can choose not to propagate anything heard from a customer to its neighbors so that a link-state message will only be sent downward a provider hierarchy. Information hiding supports policy routing. A domain may only send a link-state message received from one neighbor to another neighbor if it provides transit service between the two neighbors. With this policy, the domain's providers or other peers will not receive the link-state messages regarding to a private domain-level peering link the domain has with a peer. Therefore, those providers or peers may not use the peering connection.

For link-state protocols to work correctly, link state messages must be broadcasted reliably. This is not a trivial task [44]. Link-state messages that travel along different paths may be lost or arrive out of order. A link-state protocol needs to ensure that each router update the topology using the correct sequence of messages.

We made two design choices that significantly simplified TIPP. First, TIPP uses an improved version of the Shortest Path Topology Algorithm (SPTA) developed by Spinelli and Gallager [53] as its topology update algorithm. Unlike OSPF [40] or IS-IS [43], SPTA solves the link state consistency problem without using sequence numbers, periodic link-state refreshments, or link-state flooding. Second, TIPP messages between adjacent routers are sent over a reliable transport connection. Unlike OSPF and IS-IS, TIPP does not need built-in mechanisms to handle in-order and reliable message delivery.

The main idea of TIPP's topology update algorithm is that a node updates its topology database using messages heard from its neighbors, and resolves inconsistent update messages from different neighbors by "believing" the neighbor that is on the shortest failure-free path to the link that triggers the update messages. Messages sent along the same failure-free path are in-order and reliable, because messages sent between adjacent nodes are in-order and reliable. Therefore, the sequence of messages coming from the neighbor on the shortest failure-free path reflect the sequential status change of that link. A failure-free shortest path can be recursively computed, because a node knows whether its adjacent link to a neighbor is failure-free, and the neighbors in turn know which of their adjacent links are failure-free. Recursively, the node can determine a failure-free path to a link at any distance.

The tradeoff for simplicity is that SPTA has a per-message processing overhead linear to the number of domain-level links in a user's up-graph. But we do not think that the computation overhead will become a performance issue, because a user's up-graph is a small region of the Internet, consisting of only a user's direct and indirect providers, and their peering links.

The primary improvement we made to SPTA is the transit policy support. Topology information in TIPP is represented by a set of link records. A link record describes the reachable address prefixes for packets coming from each direction of the link. These reachable address prefixes encode the transit policy of a domain. For instance, let P denote a provider domain, and C denote a customer domain. In the link record (P, C), the reachable address prefixes from C to P is a wildcard, meaning that packets coming from a customer domain C to a provider domain P can reach all neighbors of P; the reachable address prefixes from P to C is just C's address prefixes, meaning that packets coming from a provider domain P to a customer C can only reach C and C's customers. Pseudocode of TIPP's topology update algorithm is shown in Pseudocode 1. TIPP specification can be found in [67].

Pseudocode 1 : Topology update algorithm

```
    $T_M$: Input topology database from a neighbor $M$
    $e_M$: the adjacent link record for $M$
1:  foreach neighbor $M$ do
2:      if the connection to $M$ is up then
3:          $e_M.parent = T_M$.
4:          set $e_M$'s attributes using the record in $T_M$
5:          push $e_M$ into $queue$.
6:      else
7:          clear $e_M$'s attributes
8:          set the link status in $e_M$ to be down
9:      end if
10: end for
11: while $queue$ is not empty do
12:     $fLink = queue.pop()$
13:     if $fLink$ is processed before then
14:         continue
15:     end if
16:     foreach adjacent $e$ of $fLink$'s end domain do
17:         if $fLink$ can reach $e$ && $fLink$ is up then
18:             $e.parent = fLink.parent$
19:             set $e$'s attributes using the record in $e.parent$
20:             if $e$ is up then
21:                 push $e$ into $queue$
22:             end if
23:         end if
24:     end for
25: end while
26: send changes to neighbors
```

IV. Packet Forwarding

In this section, we discuss how a router forwards a packet with a source and a destination address along a valley-free route. NIRA also supports nonvalley-free forwarding, which we describe in Section VI.

A router obtains its forwarding entries from TIPP. As described in Section III, TIPP messages include address alloca-

R_2's downhill table

1:2::/96	$self$
1:2::/32	$blackhole$
1:2:1::/48	N_1
1:2:2::/48	N_2
FFFF:1::/96	$self$
FFFF:1::/32	$blackhole$
FFFF:1:1::/48	N_1
FFFF:1:2::/48	N_2

R_2's uphill table

1:2::/32	B_1
FFFF:1::/32	$bridge$

R_2's bridge table

FFFF:2::/32	R_3

Fig. 4. Contents of R_2's forwarding tables.

tion information and reachable address prefixes via a domain. A router uses these messages to establish forwarding entries.

The key issue a router's forwarding algorithm needs to resolve is to choose the correct address to determine the next hop. That is, on the uphill part of a route, a router should use the source address to determine the next hop, and similarly, on the downhill part, a router should use the destination address. In our design, a router divides the forwarding entries it learns from TIPP into multiple forwarding tables, and uses the lookup results from different tables to determine at which part of a route a packet is, and thus chooses the corresponding address to determine the next hop.

A. Forwarding Tables

A router at a domain groups its forwarding entries learned from TIPP into three tables: an uphill table, a downhill table, and a bridge table. A domain's uphill forwarding table has entries for address prefixes allocated from the domain's providers. The next hop of each address prefix points to the provider that allocates the address prefix. Similarly, the downhill forwarding table has entries for address prefixes allocated to the domain's customers and the domain's own address prefixes. The bridge table contains the private addresses of the domain's neighbors with whom the domain has a peering relationship.

If a router also participates in a separate routing protocol, e.g., a *Core* router in domain B_1 in Fig. 2 may speak BGP with other *Core* routers, the router keeps forwarding entries learned from the routing protocol in a routing forwarding table. Routing forwarding tables and TIPP forwarding tables are kept separately to minimize the interaction between the routing protocol and TIPP.

Figs. 4 and 5 show the sample forwarding tables of domain R_2 and domain B_1 in Fig. 2. The tier-1 provider B_1 has a routing table that has entries for other tier-1 providers in the *Core*. The next hop *self* indicates that the destination is within the current domain; we'll soon explain the other special next hops: *routing*, *bridge*, and *blackhole*.

B. Forwarding Algorithm

The basic operation of the forwarding algorithm involves two steps. A router first uses the longest prefix match to look up the destination address of a packet in its downhill table. If a match is found, it indicates that the domain has allocated the destination address. So the packet is already on the downhill part of its route, and should be forwarded to the customer to which the destination address is allocated. If no match is found, the router looks up the source address of the packet in its uphill table, and forwards the packet to the provider that allocates the source address.

B_1's downhill table

1::/96	$self$
1::/16	$blackhole$
1:1::/32	R_1
1:2::/32	R_2
1:3::/32	R_3

B_1's uphill table

0::/1	$routing$

B_1's routing table

2::/16	B_2
3::/16	B_3
4::/16	B_4

Fig. 5. Forwarding tables of B_1 in Fig. 2.

A packet does not always go "up" or "down." It may cross the *Core* or a peering link. To indicate a "turn," we add special entries with the next hops pointing to either *routing* or *bridge* in the uphill table of a router. If the source address of a packet matches a special entry, it indicates that the packet has reached the "root" of its source address, and cannot be pushed up any more. The packet will be forwarded across the *Core* or a peering link. The next hop of a special entry specifies which forwarding table the router should use to determine the next hop. For a router in the *Core*, it has a special entry that matches all routeable addresses (0::/1 in our example) in the *Core* with the next hop pointing to its routing table. For a router with a peering link, it has a special entry that matches the private peering address with the next hop pointing to its bridge table.

We show one forwarding example. Suppose Bob in Fig. 2 sends a packet to Alice with a source address 1:1:1::1000 and a destination address 2:1:1::2000. The router at N_1 will not find a match for the destination address 2:1:1::2000 in its downhill table, as N_1 does not allocate that address. The router will find a match for the source address in its uphill table, as R_1 allocates the address prefix 1:1:1::/48 to N_1. The packet will be forwarded to R_1. Similarly, at R_1, the packet will be forwarded to B_1. At B_1, the destination address will not match any entry in B_1's downhill table, but the source address will match the special entry 0::/1 with the next hop pointing to the routing table of B_1. The router will then look up the destination address in B_1's routing table and find a match for the destination address with the next hop pointing to B_2. The packet will be forwarded across the *Core* to B_2, and then be forwarded along the path $B_2 \rightarrow R_3 \rightarrow N_3$. The pseudocode of our forwarding algorithm is shown in Pseudocode 2 (shown on the next page).

C. Correctness

Using induction, we show in [67] that our forwarding algorithm has the following properties: if a user specifies a route using our route representation scheme, then i) the packet will be forwarded along the route to reach its destination if the route is failure free; ii) if at a router, the next hop to forward a packet is unreachable due to failures, the packet will be dropped at that router instead of looping around in the network; iii) if at domain N, the packet is forwarded to a neighbor domain M, then at domain M, a packet with the reverse route representation will be forwarded to N.

The special entry with the next hop pointing to *blackhole* in a router's downhill table is added to ensure the second property.

Pseudocode 2 : Next-hop lookup

```
Down: downhill forwarding table;
Up: uphill forwarding table;
Bridge: bridge forwarding table;
self: the current router;
src: source address;
dst: destination address;
Lookup(a, T): longest prefix match for address a in table T.
1: nextHop = Lookup(dst, Down).
2: if nextHop == self then
3:     return self
4: end if
5: if nextHop == blackhole then
6:     goto Drop
7: else if nextHop ≠ noMatch then
8:     return nextHop
9: else
10:     nextHop = Lookup(src, Up)
11:     if nextHop == noMatch then
12:         got Drop
13:     else if nextHop == bridge then
14:         nextHop = Lookup(dst, Bridge)
15:     else if nextHop == routing then
16:         nextHop = Lookup(dst, Routing)
17:     else
18:         return nextHop
19:     end if
20:     if nextHop ≠ noMatch then
21:         return nextHop
22:     else
23:         goto Drop
24:     end if
25: end if
26: Drop:
27: drop p; send an ICMP error notification
28: return noMatch
```

A domain adds a *blackhole* entry for every address prefix it has. When the domain is disconnected from a customer due to failures, the domain may not have an entry for the customer in its downhill table. A packet destined to the customer will match the *blackhole* entry, and will be dropped. The third property ensures that a route representation for a reply packet or an ICMP packet can be generated from a packet header without invoking a route discovery process.

D. Computational Cost Analysis

NIRA's route encoding scheme is able to represent a valley-free route with only two addresses, regardless of how many domains the route consists of.

As a tradeoff, the forwarding algorithm needs to inspect not only the destination address of a packet, but sometimes the source address of a packet. Suppose the computational cost for a longest prefix match is L. For a packet with a valley-free route representation, in the best case, our forwarding algorithm finds the next hop to forward the packet with one lookup (L) in a router's downhill table; in the worst case, our forwarding algorithm will find the next hop with three lookups ($3L$): one lookup in a router's downhill table, one lookup in a router's uphill table, and one lookup in a router's bridge or routing table. The cost for finding the next hop to forward a packet with a source and a destination address ranges from L to $3L$. In contrast, the current Internet uses a destination-based forwarding scheme. The forwarding cost is one lookup L' for a 32-bit address.

We do not expect this computational cost will become a performance problem. Advanced hardware technology such as parallel lookup on the destination address field and the source address field may speed up the lookup latency. In addition, forwarding tables of a NIRA router only contain entries for its own address prefixes and those of its neighbors, and therefore are expected to be much smaller than BGP tables. A small table size may also help to reduce the longest prefix lookup latency.

V. Evaluation

Our overall goal is to design a routing system that practically supports user choice. In this section, we evaluate the technical aspects of NIRA that impacts its feasibility: scalability and efficiency. How our design satisfies other practical constraints such as provider compensation and incremental deployment are discussed in Section II. We evaluate the scalability of NIRA's route discovery mechanism. The evaluation includes the amount of state a user obtains from TIPP, the overhead to maintain the state, and the convergence speed of TIPP. We have discussed why NRLS will not become a scaling bottleneck in Section II-D, and will not discuss it in this section.

Our evaluation on efficiency focuses on the connection setup latency incurred by the reactive failure detection mechanism. This setup latency affects the performance of interactive applications and short transfers. We have discussed the header overhead and the forwarding cost of NIRA in Section IV.

We evaluate our design using a combination of network measurement, simulation, and analysis. Our evaluation shows that NIRA is scalable; TIPP has low overhead and converges fast; and the setup latency is negligible.

A. Amount of State a User Obtains From TIPP

A user in NIRA maintains a number of hierarchical addresses and an up-graph. Theoretically, the number of addresses a user has or the size of the up-graph might grow exponentially with the level of provider hierarchy. However, in practice, we expect that financial factors will limit the provider hierarchy to be shallow, and the number of providers to which a domain is connected to be small. Therefore, the number of addresses or the size of the up-graph a user has should be small.

Verifying this intuition requires knowledge of domain topology and domain relationships. Unfortunately, such information is not publicly available. The best current practice is to infer domain topology and relationships by analyzing BGP tables. There are two well-known inference algorithms: degree based [24] and ranking based [57]. We have compared the two algorithms and found their results are mostly consistent (differing by about 10%).

The results we show here are based on the inference results from the ranking based approach. The results obtained using the degree-based approach are similar. We have downloaded four representative data sets between year 2001 to year 2004 [1], one for each year. Each data set summarizes the domain-level topology and domain relationships using BGP table dumps from multiple vantage points. The inference results categorize three types of core domains: the dense core, the transit core, and the outer core. In our evaluation, we approximate the *Core* structure in NIRA by including all domains in the dense core and the transit core into the *Core*. Fig. 6 summarizes the data sets. Each domain-level link is unidirectional.

We have built a simulator to simulate the case in which NIRA is deployed on the inferred domain-level topologies. We collect data to measure the number of addresses and the number of link records a domain (or a user in a domain) obtained from TIPP.

Date	# domain	# link	# core
2001/04/18	10915	47514	150
2002/04/06	13155	56634	141
2003/01/09	14695	61630	140
2004/01/13	16809	74368	167

Fig. 6. Domain-level topologies.

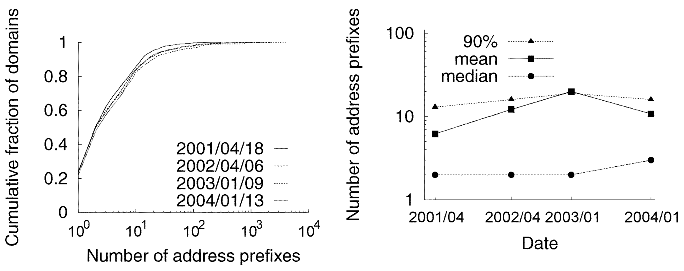


Fig. 7. Number of hierarchically allocated prefixes of each domain as a cumulative distribution, and the mean, median, and the 90th percentile.

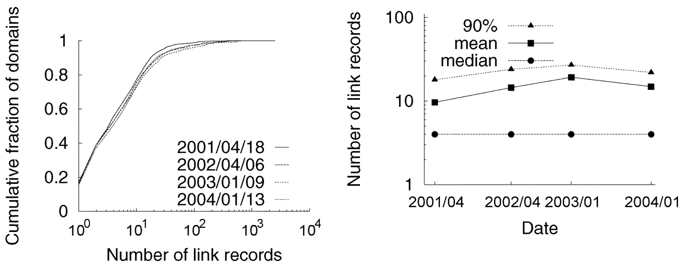


Fig. 8. Number of link records in a domain's main topology database as a cumulative distribution, and the mean, median, and the 90th percentile.

We also measure the number of forwarding entries a TIPP router has.

Fig. 7 shows the number of address prefixes allocated to each domain as a cumulative distribution, and the mean, median and the 90th percentile of the distribution. It can seen that 90% of the domains will have less than 20 prefixes, and the number of prefixes a domain has does not grow with the size of the Internet. However, the largest number is more than a thousand. Hand-debugging a few examples suggest that the tail part of the distribution may be caused by inference errors, e.g., a peering relationship is mistaken into a provider-customer relationship.

Fig. 8 shows the cumulative distribution of the number of link records propagated by TIPP to a domain. The mean, median, and 90th percentile of the distribution are also shown in the figure. Again, 90% of the domains will maintain less than 30 link records.

Fig. 9 shows the cumulative distribution of the number of forwarding entries in a domain's uphill, downhill, and bridge forwarding tables, together with the mean, median, and 90th percentile of the distribution. About 90% of the domains have less than 100 forwarding entries.

These results suggest that NIRA would have been practical if deployed in today's Internet. With NIRA, the Internet topology may change. But we expect that the same financial constraints will still apply in the future. When two domains interconnect, there is cost involved for laying fibers, renting circuits, buying switches etc. A provider will always need to balance the cost

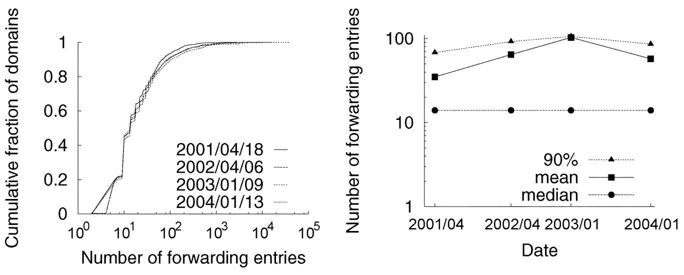


Fig. 9. Number of forwarding entries in a TIPP router's three logical forwarding tables as a cumulative distribution, and the mean, median, and the 90th percentile.

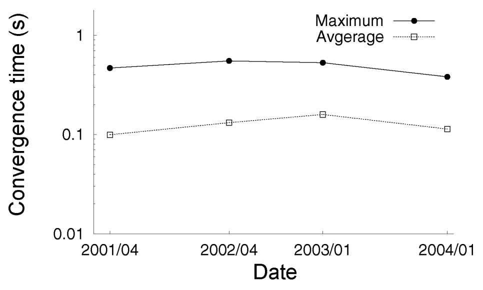


Fig. 10. Average and maximum time elapsed between the time a failure is detected and the time the last topology message triggered by the failure is received.

to interconnect with another domain and the profit that interconnection would bring. Therefore, we believe that NIRA will remain practical in the future.

B. Message Overhead and Convergence Speed of TIPP

We evaluate the convergence property and the message overhead of TIPP using measurement-based simulations. We implemented TIPP in the ns-2 simulator [42], and ran TIPP simulations on Internet-like topologies. We made a detailed implementation of TIPP: we built a socket layer into ns-2, and implemented the full TIPP state machine and message processing operations on top of the socket layer. This detailed implementation helps us evaluate the design complexity, but limits the scale of our simulation. Fortunately, because TIPP messages do not propagate globally, when collecting statistics such as how fast TIPP propagates a link state message to a user or the number of TIPP messages a user receives, one does not need to include the part of the network from which a user does not receive TIPP messages. Therefore, we use sampled topologies for our simulations. We randomly select a number of edge domains from each inferred Internet topology, and include the edge domains' up-graphs to generate a simulation topology. For our particular simulations, each sampled topology includes twenty edge domains' up-graphs.

We first look at the convergence time of TIPP. In a simulation for a sampled topology, we randomly select bidirectional links and let them sequentially fail and recover. We record the period between the time when a failure is detected and the time when the last topology message triggered by the failure is received for each sequential failure event. In the simulation, the propagation delay of a link is randomly distributed between 10 and 110 ms. Fig. 10 shows the average and the maximum convergence time over ten simulation runs. It can be seen that the average convergence time of TIPP is on the order of a link's propagation delay.

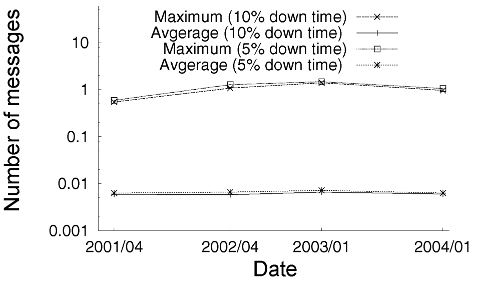


Fig. 11. Average and maximum number of messages sent per failure per link when each link randomly fails and recovers.

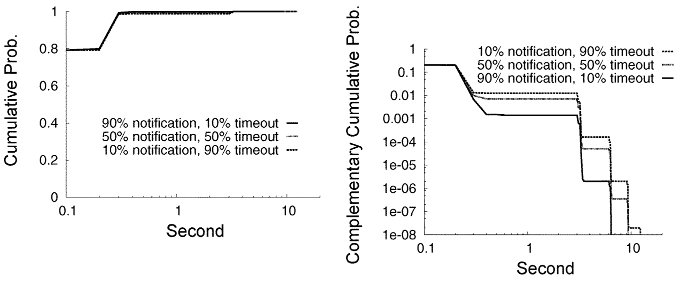


Fig. 12. Cumulative distribution ($P\{I \leq x\}$) and the complementary cumulative distribution ($P\{I > x\}$) of the connection set up latency I, with 1% route failure probability, 80% NRLS cache hit rate, 100 ms round trip delay, 3 seconds timeout value, and 3-level of NRLS name hierarchy. The fraction that a failure is detected by router notification is varied from 90%, to 50%, and to 10%. The rest of failures are detected via a timeout.

This convergence time indicates that TIPP has the fast convergence property of a link-state routing protocol: the convergence time is proportional to the message propagation delay.

Next, we look at the message overhead of TIPP. In our simulation, we let a bidirectional link randomly fail and recover. Link failures may happen concurrently. Link failure and recovery is modelled as a sequence of on/off events. Each on/off event lasts for a uniform randomly distributed time. The percentage of the off period is the link failure rate. We count the total number of messages and bytes triggered by the failures and average them over the number of failures and the number of links. We also record the maximum number seen on a link, and average it over the number of failures. Fig. 11 shows the message overhead when there are multiple link failures. The link failure rates are 5% and 10%, respectively. As can be seen, the average message overhead per link is very low, demonstrating that TIPP messages are propagated in controlled scopes. If a message is propagated globally, a failure and a recovery event will generate four messages (two for each direction of a connection) on a link. In our simulations, the largest average number of messages received by a link over multiple failures is less than 2, indicating that there is no message churning.

C. Setup Latency Incurred by Reactive Failure Detection

The basic mechanism we provide for route availability discovery is a combination of proactive and reactive notification. We analyze how connection setup latency is affected by the reactive failure notification scheme. Our analysis assumes a conservative approach for failure detection and handling: a user depends on the basic proactive and reactive notification mechanisms to detect route failures, and will re-send a packet along a different route if he discovers the original route is unavailable. This conservative assumption gives us an upper bound on the connection setup latency.

Intuitively, if routes are highly available, then most likely a packet will reach its destination the first time when a user sends it. So reactive notifications will not significantly affect connection setup latency. We use a simple analytic model to test the above hypothesis. Assuming failures on different routes are independent and identically distributed, the process of successfully sending a packet may involve a random number of failed trials, and one success at the end. The failed trial may end with a router notification or a timeout. This process can be modeled as a negative multinomial distribution [19]. The analytic model can be found in a longer version of this paper [67].

We can numerically compute the distribution of the connection set up latency I. This latency includes the NRLS query latency, the failure detection and retry latency, and finally the latency to send a connection setup packet such as a TCP SYN. Fig. 12 shows the cumulative distribution and the complementary cumulative distribution ($P\{I > x\}$) of the connection set up latency, with 1% route failure probability, 80% NRLS cache hit rate, 100-ms round trip delay, 3-s timeout value, and 3-level of NRLS name hierarchy. We vary the fraction that a failure is detected by a router notification from 90%, to 50%, and to 10%. The rest of failures are detected via timeout. The NRLS cache hit rate is set according to the study that shows the DNS cache hit rate exceeds 80% with a time-to-live field as short as 15 minutes [34]. The failure rate is set according to the studies that show the route failure rate in the present Internet is less than 1% [18], [38]. With a 1% route failure rate, the connection setup time is mostly affected by the NRLS cache hit rate. Nearly 80% of the connections that have a NRLS cache hit can send their packets successfully within a round trip time, and nearly 99% of them can send their first packets successfully within three round trip time.

VI. Policy Exceptions

Our design is optimized for the common case that a domain-level route is valley-free, but it also works when there are nonvalley-free transit policies. Non-valley-free routes are represented by a sequence of addresses. A nonvalley-free route can be broken into shorter route segments, with each route segment being valley-free. For instance, in Fig. 2, if the domain R_2 is willing to provide transit service between its provider B_1 and its peer R_3, then the route between Bob and Alice $N_1 \rightarrow R_1 \rightarrow B_1 \rightarrow R_2 \rightarrow R_3 \rightarrow N_3$ is a policy-allowed nonvalley-free route. It can be broken into two valley-free segments: $N_1 \rightarrow R_1 \rightarrow B_1 \rightarrow R_2$ and $R_2 \rightarrow R_3 \rightarrow N_3$. Each valley-free route segment can be represented by two addresses. The nonvalley-free route can be represented by concatenating the encodings of the two valley-free segments. For instance, the route segment from Bob to R_2 $N_1 \rightarrow R_1 \rightarrow B_1 \rightarrow R_2$ can be represented by 1:1:1::1000 and 1:2::/32; the route segment from R_2 to Alice $R_2 \rightarrow R_3 \rightarrow N_3$ can be represented by FFFF:1::/32 and FFFF:2::1::2000. The nonvalley-free route $N_1 \rightarrow R_1 \rightarrow B_1 \rightarrow R_2 \rightarrow R_3 \rightarrow N_3$ can be represented by a routing header with a total of four addresses: 1:1:1::1000, 1:2::/32, FFFF:1::/32, and FFFF:2::1::2000. The first two addresses will be placed in the source and the destination address field. When a packet with the routing header arrives at R_2, the next two addresses will be shifted to the source and the destination address field.

The advantage of this representation scheme is that only domains that have special nonvalley-free transit policies need to turn on the source routing option. Routers in other domains can ignore the option, and forward packets using only the source and the destination address in a packet header. More details on the nonvalley-free route representation and forwarding schemes can be found in [67].

We do not provide separate mechanisms for users to discover nonvalley-free routes. Domains such as R_2 may use TIPP to propagate its special transit policies to its neighbors. Users may also store nonvalley-free routes at their NRLS servers. We note that TIPP and NRLS do not guarantee that users can discover all possible routes, because TIPP messages do not propagate globally. However, our design isolates route discovery as an individual module, and allows users to use general mechanisms for route discovery. For instance, a provider that offers nonvalley-free transit service may advertise its service on its web page. It is out of the scope of this paper to study those general mechanisms.

VII. Related Work

At a high level, related work falls into three categories: scalable routing schemes, routing architecture proposals, and current route selection technologies.

A. Scalable Routing

Scalable routing schemes aim to reduce the amount of routing state a router keeps. The well-known schemes include the cluster-based hierarchical routing [36], the landmark hierarchical routing system [60], geographical routing [20], and hybrid routing [58]. However, the goals of these routing systems are fundamentally different from us. We aim to provide a feasible and scalable approach to support user-controlled routes, while they aim to reduce the size of routing tables, or the number of routing updates, and do not necessarily support user-selected routes.

Provider-rooted hierarchical addressing has long been proposed to scale the Internet routing [14], [21], [22], [61]. Our design builds on the idea of hierarchical addressing, but we developed a complete routing system that supports user route choice.

B. Routing Architecture Proposals

Nimrod [9] proposes to use a map-distribution mechanism for a source to discover network topology and to use virtual circuit to set up routes. However, Nimrod does not address how to fit its design into a policy-rich inter-domain routing environment. In contrast, our design is optimized to fit into the Internet routing environment, and preserves the packet-switched feature of the Internet.

The inter-domain policy routing (IDPR) protocol [55] proposes to use a domain-level link state routing protocol to distribute the domain-level topology to route servers of each domain. A source sends a route request to a route server to obtain a domain-level route. Our work does not need a global link state routing protocol for a user to discover routes, and does not require the presence of a per-domain route server.

The scalable inter-domain routing architecture [16] proposes to use a BGP router's routing information base and to flood a route request from a destination to a source to assist a user to discover multiple routes from the destination to the source. We design a new protocol TIPP to assist route discovery. TIPP has less overhead than the flooding-based query approach, and allows a user to discover more routes than what is present in a router's BGP table.

TRIAD [11], [26] is an Internet architecture that provides explicit support for content routing. Packets could be routed by names, rather than by IP addresses. NIRA is designed to handle IP layer routing and addressing issues. The design goal is fundamentally different. TRIAD includes a wide-area relaying protocol (WRAP) [47] that provides extended addressing and improved loose-source routing. NIRA follows the good lead of WRAP to use the path-based addressing scheme.

Feedback-based routing [68] proposes to use a domain-level link-state routing protocol for edge routers to learn domain-level topology. An edge router selects a route to reach a destination, monitors the route condition, and switches to a different routes if the route fails. In contrast, our work does not require global link-state routing. We can leverage the route monitoring algorithm described in this proposal for rapid route fail-over.

Platypus [46] is designed for the case that users have contractual agreements with many ISPs. A user attaches a cryptographic hash value in his packets as a proof that he has purchased service from an ISP. NIRA assumes the bilateral contracts and treats valley-free routes as the common case.

The HLP proposal [58] intends to improve the scalability of BGP. HLP uses a mixture of a link-state routing protocol and a path vector routing protocol to provide fast routing convergence. The goal of NIRA is essentially different. NIRA aims to allow users to choose provider-level routes, and includes a protocol TIPP that distributes routes and topology information to users for them to choose routes.

We note that NIRA and the IPv6 site multihoming proposal [32], [41] share some similarity. The IPv6 proposal also gives multiple addresses to a multihomed site. A key difference between NIRA and the IPv6 proposal is that NIRA considers the address selection problem as a path selection problem. It provides necessary topology information for users to select a path. The protocol TIPP propagates topology information associated with addresses to users. This allows a user to map an address to a provider-level route, and to choose an initial source address that is failure-free. Moreover, NIRA uses both the source and the destination address to forward a packet. This allows a user to control the domain-level route, including both the part of the route in the sender's access network, and the part of the route in the destination's access network. In contrast, the IPv6 proposal does not change the routing paradigm of the Internet. Routes are chosen by routers, and forwarding is destination-based. As a result, the source address of a packet influences the return path a packet takes, but does not determine the outgoing path the packet follows. An end host finds a working path by exploring address pairs [4], [6] without knowing the providers the addresses map to. Further, NIRA allows a user to choose beyond the first hop provider.

C. Current Route Control Technologies

Both commercial route control products [33], [50] and overlay networks [3], [51], [56], [59] offer route selection service to some extent. Route control products are limited to selecting the next hop provider for outbound traffic, and cannot choose beyond the first hop provider. Moreover, they are generally not affordable by individual users or small sites. An overlay network has a limited scope. Only nodes on an overlay network can control their paths by tunnelling traffic through other nodes on the overlay network. Our work aims at providing a long term solution to support user route selection. We introduce changes at the network layer, and once deployed, all Internet users are able to benefit from our design.

Consumers today can manually select providers in the cellular phone and telephone markets. In contrast, NIRA provides protocols to inform a user of the available routes so that provider selections can be done by software at a fine granularity, such as at the granularity of per connection or per application.

VIII. Conclusion

Giving a user the ability to choose domain-level routes has the potential of fostering ISP competition to offer enhanced service and improving end-to-end performance and reliability. We present the design of NIRA, an inter-domain routing system that practically supports user choice. The design of NIRA addresses a broad range of issues, including ISP compensation, scalable route discovery, efficient route representation, fast route fail-over, and security. NIRA supports user route choice without running a global link-state routing protocol. Our design splits an end-to-end route into a sender part and a receiver part, and uses an address to represent each part. A user can choose routes by choosing addresses. As both the source address and the destination address are used for forwarding, packets with arbitrary spoofed source addresses will be dropped, and will not be forwarded to their destinations. NIRA includes a protocol TIPP that propagates to a user his addresses and the topology information associated with his addresses. Our evaluation suggests that NIRA is practical. It supports user route choice with low overhead.

Acknowledgment

The authors wish to thank the anonymous reviewers and Editor O. Bonaventure for their useful comments, which greatly helped them in revising this paper.

References

[1] S. Agarwal, Domain relationship inferrence data, Univ. California, Berkeley, 2004 [Online]. Available: http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/data/

[2] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman, "A measurement-based analysis of multihoming," in *Proc. ACM SIGCOMM*, 2003, pp. 353–364.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. ACM Symp. Operating Syst. Principles*, 2001, pp. 131–145.

[4] J. Arkko and I. Beijnum, *Failure detection and locator pair exploration protocol for IPv6 multihoming*, IETF Internet draft, draft-ietf-shim6-failure-detection-05.txt, 2006.

[5] AT&T, SBC and AT&T Merger News, AT&T, San Antonio, TX, 2005 [Online]. Available: http://www.att.com/merger/

[6] M. Bagnulo, *Default locator-pair selection algorithm for the SHIM6 protocol*, IETF Internet draft, draft-ietf-shim6-locator-pair-selection-00.txt, 2006.

[7] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and I. S. Walfish, "A layered naming architecture for the internet," in *Proc. ACM SIGCOMM*, 2004, pp. 343–352.

[8] Y. M. Braunstein, "Broadband industry structure: Policy, pricing and penetration," presented at the Pacific Telecommun. Conf., 2003 [Online]. Available: htp:/www.ischool.berkeley.edu/~bigyale/ptc2003_braunstein.pdf

[9] I. Castineyra, N. Chiappa, and M. Steenstrup, *The Nimrod Routing Architecture,* IETF Informational, RFC 1992, 1996.

[10] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger, "The origin of power laws in Internet topologies revisited," in *Proc. IEEE INFOCOM*, 2002, vol. 2, pp. 608–617.

[11] D. R. Cheriton and M. Gritter, TRIAD: A new next-generation Internet architecture. Stanford Univ., Stanford, CA, Tech. Rep., 2000.

[12] D. Clark, J. Wroclawski, K. Sollins, and R. Braden, "Tussle in cyberspace: Defining tomorrow's internet," in *Proc. ACM SIGCOMM*, 2002, pp. 462–475.

[13] R. Cox, A. Muthitacharoen, and R. Morris, "Serving DNS using chord," presented at the IPTPS'02, 2002 [Online]. Available: http://www.cs.rice.edu/Conferences/IPTPS02/

[14] S. Deering and R. Hinden, *Internet protocol version 6 (IPv6) addressing architecture*, IETF Proposed Standard, RFC 3513, 2003.

[15] N. Economides, The Telecommunications Act of 1996 and its Impact, NYU Center for Law and Business, New York, 1998.

[16] D. Estrin, Y. Rekhter, and S. Hotz, "Scalable inter-domain routing architecture," in *Proc. ACM SIGCOMM*, 1992, pp. 40–52.

[17] P. Faratin, J. Wroclawski, G. Lee, and S. Parsons, "Social agents for dynamic access to wireless networks," in *Proc. AAAI Spring Symp. Human Interaction with Autonomous Systems in Complex Environments*, Stanford, CA, Mar. 2003.

[18] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek, "Measuring the effects of internet path faults on reactive routing," in *Proc. ACM SIGMETRICS*, 2003, pp. 126–137.

[19] W. Feller, "The binomial and Poisson distributions," in *An Introduction to Probability Theory and Its Applications*. New York: Wiley, 1968, ch. VI.

[20] G. Finn, Routing and addressing problems in large metropolitan-scale Internetworks, University of Southern California, Los Angeles, ISI Tech. Rep. ISI/RR-87-180, 1987.

[21] P. Francis, "A near-term architecture for deploying Pip," *IEEE Networking*, vol. 7, no. 3, pp. 30–37, May 1993.

[22] P. Francis, "Comparison of geographical and provider-rooted internet addressing," *Comput. Netw. ISDN Syst.*, vol. 27, no. 3, pp. 437–448, 1994.

[23] P. Francis and R. Gummadi, "IPNL: A NAT-extended internet architecture," in *Proc. ACM SIGCOMM*, 2001, pp. 69–80.

[24] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, Dec. 2001.

[25] D. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang, "Optimizing cost and performance for internet multihoming," in *Proc. ACM SIGCOMM*, 2004, pp. 79–92.

[26] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet," in *Proc. USENIX Symp. Internet Technol. Syst.*, 2001, pp. 37–48.

[27] K. P. Gummadi, H. Madhyastha, S. D. Gribble, H. M. Levy, and D. J. Wetherall, "Improving the reliability of internet paths with one-hop source routing," in *Proc. OSDI*, 2004, pp. 183–198.

[28] M. Handley and A. Greenhalgh, "The case for pushing DNS," presented at the Hotnets-IV, 2005 [Online]. Available: http://www.sigcomm.org/HotNets-IV.

[29] R. Harrison, *Lightweight directory access protocol (LDAP): Authentication methods and security mechanisms,* IETF Proposed Standard, RFC 4513, 2006.

[30] G. Huston, "Interconnection, peering and settlements – Part I," *Internet Protocol J.*, vol. 2, no. 1, pp. 2–16, Mar. 1999.

[31] G. Huston, "Interconnection, peering and settlements – Part II," *Internet Protocol J.*, vol. 2, no. 2, pp. 2–23, Jun. 1999.

[32] G. Huston, *Architectural approaches to multi-homing for IPv6,* IETF Informational, RFC 4177, 2005.

[33] Internap Products and Services, Internap, Atlanta, GA, 2005 [Online]. Available: http://www.internap.com/products/route-optimization.htm

[34] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 589–603, Oct. 2002.

[35] R. Keralapura, C.-N. Chuah, N. Taft, and G. Iannaccone, "Can co-existing overlays inadvertently step on each other," in *Proc. IEEE ICNP*, 2005, pp. 211–214.

[36] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks: Performance evaluation and optimization," *Comput. Netw.*, vol. 1, no. 3, pp. 155–174, Jan. 1977.

[37] G. Lee, P. Faratin, S. Bauer, and J. Wroclawski, "A user-guided cognitive agent for network service selection in pervasive computing environments," in *Proc. IEEE PerCom*, Orlando, FL, Mar. 2004.

[38] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *Proc. IEEE INFOCOM*, Mar. 7–11, 2004, vol. 4, pp. 2307–2317.

[39] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," in *Proc. ACM SIGCOMM*, 1988, pp. 123–133.

[40] J. Moy, *OSPF Version 2,* IETF Standard, RFC 2328, 1998.

[41] E. Nordmark and M. Bagnulo, *Level 3 multihoming shim protocol*, IETF Internet draft, draft-ietf-shim6-proto-05.txt, 2006.

[42] The Network Simulator–ns-2 2004 [Online]. Available: http://www.isi.edu/nsnam/ns/

[43] D. Oran, *OSI IS-IS Intra-Domain Routing Protocol*, IETF Informational, RFC 1142, 1990.

[44] R. Perlman*, Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*. Reading, MA: Addison-Wesley, 2000.

[45] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On selfish routing in internet-like environments," in *Proc. ACM SIGCOMM*, 2003, pp. 151–162.

[46] B. Raghavan and A. C. Snoeren, "A system for authenticated policy-compliant routing," in *Proc. ACM SIGCOMM*, 2004, pp. 167–178.

[47] C. Rai and D. Cheriton, Wide-area relay addressing protocol (WRAP): Packet relay in TRIAD, Stanford Univ., Stanford, CA [Online]. Available: http://www-dsg.stanford.edu/triad/wrap_spec.txt

[48] Y. Rekhter, T. Li, and S. Hares, *A border gateway protocol 4 (BGP-4)*, IETF Draft Standard, RFC 4271, 2006.

[49] T. Roughgarden*, Selfish Routing and the Price of Anarchy*. Cambridge, MA: The MIT Press, 2005.

[50] Routescience's pathcontrol, Networkworld, Southborough, MA, 2002 [Online]. Available: http://www.networkworld.com/reviews/2002/0415rev.html

[51] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: Informed Internet routing and transport," *IEEE Micro*, vol. 19, no. 1, pp. 50–59, Jan. 1999.

[52] M. Shand and S. Bryant, *IP Fast Reroute Framework*, IETF Internet draft, draft-ietf-rtgwg-ipfrr-framework-05.tx, 2006.

[53] J. Spinelli and R. Gallager, "Event driven topology broadcast without sequence numbers," *IEEE Trans. Commun.*, vol. 37, no. 5, pp. 468–474, May 1989.

[54] P. Srisuresh and K. Egevang, *Traditional IP network address translator (Traditional NAT)*, IETF RFC 3022, 2001.

[55] M. Steenstrup, *An Architecture for Inter-Domain Policy Routing*, IETF Proposed Standard, RFC 1478, 1993.

[56] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proc. ACM SIGCOMM*, 2002, pp. 73–86.

[57] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the internet hierarchy from multiple vantage points," in *Proc. IEEE INFOCOM*, 2002, vol. 2, pp. 618–627.

[58] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica, "HLP: A next-generation interdomain routing protocol," in *Proc. ACM SIGCOMM*, 2005, pp. 13–24.

[59] J. Touch and S. Hotz, "The X-bone," *IEEE Global Internet*, 1998.

[60] P. F. Tsuchiya, "Landmark Routing: Architecture, Algorithms, and Issues," The MITRE Corporation, Bedford, MA, Tech. Rep. MTR-87W00174, 1988.

[61] P. F. Tsuchiya, "Efficient and robust policy routing using multiple hierarchical addresses," in *Proc. ACM SIGCOMM*, 1991, pp. 53–65.

[62] J.-P. Vasseur, M. Pickavet, and P. Demeester*, Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. New York: Morgan Kaufmann, 2004.

[63] Verizon and MCI Introduce Verizon Business, Verizon, Basking Ridge, NJ [Online]. Available: http://www22.verizon.com/merger/

[64] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, *Dynamic updates in the domain name system (DNS UPDATE)*, Proposed Standard RFC 2136, 1997.

[65] B. Wellington, *Secure domain name system (DNS) dynamic update*, IETF Proposed Standard RFC 3007, 2000.

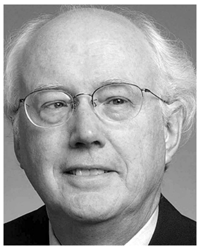
[66] X. Yang, "NIRA: A new internet routing architecture," in *Proc. ACM SIGCOMM FDNA Workshop*, 2003, pp. 301–312.

[67] X. Yang, "The design and evaluation of a new Internet routing architecture (NIRA)," Ph.D. dissertation, M.I.T., Cambridge, MA, 2004.

[68] D. Zhu, M. Gritter, and D. R. Cheriton, "Feedback based routing," *ACM SIGCOMM CCR*, vol. 33, no. 1, pp. 71–76, 2003.

Xiaowei Yang (S'99–M'05) received the Ph.D. degree in computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 2004.

Currently, she is an Assistant Professor in the Department of Computer Science at the University of California, Irvine. Her research interests include congestion control, quality of service, Internet routing architecture, and network security.



David Clark (M'66–F'98) received the Ph.D. degree from the Massachusetts Institute of Technology (MIT), Cambridge, in 1973.

He has been a Research Scientist at the Computer Science and Artificial Intelligence Laboratory, MIT , since 1973. Since 1975, he has been involved in the design of the Internet. He has worked on Internet architecture, QoS, economic and policy issues, broadband access, and security.

Arthur W. Berger received the Ph.D. degree in applied mathematics from Harvard University, Cambridge, MA, in 1983.

He then worked at Bell Labs (and subsequently AT&T Labs). Currently he is a Senior Research Scientist at Akamai Technologies, Cambridge, MA, and a Research Associate at the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (MIT), Cambridge. His research interests include: network architecture, overlay routing, and measurement, modeling and prediction of Internet performance.