The University of Kansas

Technical Report

Postmodern Internetwork Architecture

Bobby Bhattacharjee, Ken Calvert, Jim Griffioen,
Neil Spring, and James P.G. Sterbenz

ITTC-FY2006-TR-45030-01

February 2006

Project Sponsor:
National Science Foundation

Postmodern Internetwork Architecture

Bobby Bhattacharjee[1], Ken Calvert[2], Jim Griffioen[2], Neil Spring [1], and James Sterbenz[3]

February, 2006

[1]Computer Science, University of Maryland, College Park, MD
[2]Computer Science, University of Kentucky, Lexington, KY
[3]Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS

Abstract

Network-layer innovation has proven surprisingly difficult, in part because internetworking protocols ignore competing economic interests and because a few protocols dominate, enabling layer violations that entrench technologies. Many shortcomings of today's internetwork layer result from its inflexibility with respect to the policies of the stakeholders: users and service providers. The consequences of these failings are well-known: various hacks, layering violations, and overloadings are introduced to enforce policies and attempt to get the upper hand in various "tussles". The result is a network that is increasingly brittle, hostile to innovation, vulnerable to attack, and insensitive to concerns about accountability and privacy.

Our project aims to design, implement, and evaluate through daily use a minimalist internetwork layer and auxiliary functionality that anticipates tussles and allows them to be played out in policy space, as opposed to in the packet-forwarding path. We call our approach *postmodern internetwork architecture*, because it is a reaction against many established network layer design concepts. The overall goal of the project is to make a larger portion of the network design space accessible without sacrificing the economy of scale offered by the unified Internet.

We will use the postmodern architecture to explore basic architectural questions. These include:

- What mechanisms should be supported by the network such that any foreseeable policy requirement can be explicitly addressed?
- To what extent can routing and forwarding be isolated from each other while maintaining an efficient and usable network?
- What forms of identity should be visible within the network, and what forms of accountability do different identities enable?
- What mechanisms are needed to enable efficient access to cross-layer information and mechanisms such that lower layers can express their characteristics and upper layers can exert control downward?

We plan to build and evaluate a complete end-to-end networking layer to help us understand feasible solutions to these questions.

The Internet has fulfilled the potential of a complete generation of networking research by producing a global platform for innovation, commerce, and democracy. Unfortunately, the Internet also amply demonstrates the complexity and architectural ugliness that ensue when competing interests vie for benefits beyond those envisioned in the original design. This project is about redesigning the waist of the architectural hourglass to foster innovation, enhance security and accountability, and accomodate competing interests.

1 Postmodern Internetwork Architecture

Network-layer innovation has proven surprisingly difficult, in part because a few protocols dominate, which enables layer violations that entrench technologies. These layer violations support the policies that were not explicitly designed for within the existing architecture. For example, few policy goals, such as privacy, accountability, and even performance, are supported by explicit mechanisms. Disruptions such as denial of service attacks, spoofing, and routing outages expose imperfections in the architecture every day. The research community has identified many problems in the Internet architecture and proposed solutions [31, 108, 32, 109, 43], most of which are constrained because they aim for some backward-compatibility.

Many shortcomings of today's internetwork layer result from its inflexibility with respect to the policies of the stakeholders: users and service providers. The consequences of these failings are well-known: various hacks, layering violations, and overloadings are introduced to enforce policies and attempt to get the upper hand in various "tussles" [34]. The result is an Internet that is increasingly brittle, hostile to innovation, vulnerable to attack, and insensitive to concerns about accountability and privacy.

We propose to *design, implement, and evaluate through daily use a minimalist internetwork layer and auxiliary functionality that anticipates tussles and allows them to be played out in policy space, as opposed to in the packet-forwarding path.* We term this design a *postmodern internetwork architecture* because it is a reaction against many established network layer design concepts. The overarching goal of the project, and the motivation for choosing a *minimal* networking layer, is to make a larger portion of the network design space accessible without sacrificing the economy of scale offered by agreeing on a common set of protocols. The primary tenets of our postmodern design are (i) strict separation of concerns, and (ii) inclusion of explicit mechanisms in support of all foreseeable policies influencing network-layer behavior.

These principles lead to a design with the following novel characteristics:

- We *separate path determination from forwarding* to allow users greater control over the paths followed by packets through the network. At the same time, providers may, if they desire, retain control over transit paths through their networks for traffic engineering purposes. We also enable users to amortize the cost of determining a path from a source to a destination over multiple packets (if they choose to do so) via a *caching* mechanism.

- We *eschew hierarchical identifiers* (i.e., anything like IP addresses); components in the topology can thus be added and rearranged without concern for allocation of scarce resources (topology-based addresses) and without consulting a global authority.

- We *capture an unforgeable record of the path traversed by each packet* to provide the accountability that would curtail denial of service, spam, and other forms of abuse. At the same time, we attempt to provide complete control to providers over the "opacity" of their network topologies and policies. The portion of the recorded path inside their network can be used to reconstruct internal topology only with the provider's cooperation and as controlled by the provider's policy. We thus isolate the concern of path selection from that of topology disclosure.

- On the other hand (and to some extent in compensation for this isolation), our architecture *requires transparency of the top-level interconnection of realms.* We believe the desire for providers to hide peering relationships is an artifact of existing protocols (BGP) and the fact that realm-level customer-provider relationships are largely constrained by topology today.

- We *separate the customer-provider relationship from topology* by providing an explicit mechanism for expressing *why* each router should forward a packet. A user might be a customer of several transit providers; explicitly motivating routers to forward packets enables customer-directed routing that would choose among them.

- We support *information flow from the network* to the user (for example, about traffic conditions or path availability), and *policy flow from the user* to the elements of the network (for example, preferential drop policies) via cross-layer "knobs and dials."

Our research team brings to the project a wealth of experience in several critical areas, including design and implementation of novel network architectures [39, 53, 52, 15, 16, 17]; understanding and inferring service provider policies [95, 97, 70]; distributed identities and PKIs [73]; and performance optimization using cross-layer hints [64, 98]. In addition, the PIs have a strong record of building real systems and demonstrating end-to-end benefit of architectural decisions and in-network mechanisms. The team has a history of working together and blends complementary

skills in protocol architectures (Calvert/Griffioen/Sterbenz), protocol implementations (all PIs), novel routing schemes (Calvert/Griffioen), high-speed networks (Sterbenz), security (Bhattacharjee/Spring), topology discovery, routing policy and network management (Spring). These domain-specific skills will provide invaluable experience in the design and implementation of the architecture. The team will meet in person semi-annually and hold bi-weekly teleconferences for planning and accountability.

The rest of the proposal is organized as follows. In the next section, we discuss the motivating policy problems. Section 3 gives an overview of our postmodern approach. Section 4 describes the design of our minimal internetwork layer in more detail. Section 5 describes how we will carry out the research and evaluate our designs. Section 6 relates our proposal to the state of the art and prior work.

2 The Role of Mechanism and Policy

To arrive at a minimal internetwork architecture, we first ask, what *must* a network layer support? End-to-end addressing is the answer of textbooks, but not the answer in practice: address translation, impeded reachability with firewalls, and isolated virtual private networks are useful and common violations of end-to-end addressing and reachability. The new answer, after decades of experience and frustration, is that the network layer must support a diversity of mechanisms to achieve a diversity of policies. Accountability, anonymity, authentication, authorization, censorship, confidentiality, copyright enforcement, protocol filtering, spam filtering, etc., are all reasonable, sometimes incompatible, policy goals, none of which is well-supported by the Internet.

How might we enable a network architecture that achieves these policy goals? We first remove addressing, and with it, routing and at least some aspects of forwarding. By decoupling these functions from the base network layer, we open the possibility of isolating legitimate from malicious users, constructing policy-compliant source routes to deliver end-to-end performance guarantees, and providing accountability where needed.

But policies complicate networking. Protocols specify behaviors that enable users to cooperate to achieve some mutually-beneficial objective; a well-designed protocol admits a wide range of behaviors that achieve the objective. The purpose of policy, on the other hand, is to constrain local actions in order to elevate the interests of one stakeholder over another. Examples of policy in today's network layer include preferring certain packets over others when resources are scarce [18, 40], preferring to send traffic via some providers over others [94, 103], and which packets to forward across a trust boundary [30]. In our thin network layer, policy also encompasses who can specify forwarding behavior, select partial paths, and access information about the topology of a provider network.

Because explicit enforcement mechanisms are missing from IP, header bits are overloaded as *implicit* enforcement mechanisms. The canonical example is the use of port numbers to determine whether traffic is associated with benign or nefarious purposes. This overloading puts the user, who has no alternative mechanism for directing the traffic, at a disadvantage. The result is inevitably an arms race, as the use of HTTP as a vehicle to get anything and everything through firewalls attests.

Routing and Policy The interaction of *routing* and policy in the Internet has received much attention, because the Border Gateway Protocol [81] allows providers to express policies regarding route preference, and conflicting policies can result in routing instability. *Users* have no corresponding method of expressing routing policy. Conventional wisdom holds that source routing—one alternative for enhanced user control—has been disabled because it circumvents ISP routing policy. That is, if source routing were in widespread use, traffic engineering would not be effective at relieving hotspots, and anarchy would ensue. One can imagine that source routing might also circumvent tenuously placed filters in the middle of the network, allowing traffic that should otherwise not traverse an interface to do so. Conventional wisdom, however, applies only to conventional source routing. No explicit policy language expresses the means for routing through the network, and as a result, no intermediate router can decide whether a given source route is compliant. Simply, even the inter-domain routing invariants described by Gao [44] cannot easily be verified on-line.

Our objective with respect to supporting greater routing flexibility is *not* to solve the BGP problems of today's Internet, but to recognize the complexities of prioritizing routes and of traffic engineering in a cooperative network as first-class problems that deserve explicit design in the routing architecture. These goals need not rely on complex policy expressions, but rather, on designed protocols that explicitly consider business rules and assume common-sense business rules are universal.

Imagine a distance-vector-type routing protocol that includes not only the "cost" of each path, but a specification of the type of traffic supported along it. We might then build a network that routes only solicited traffic: only traffic that matches an expression can be delivered to networks and to hosts. In today's terms, that expression might include

only connection-setup requests (SYN packets) that match a few ports, as well as the packets of existing connections authorized by both endpoints.

Although annotations about which traffic can be supported along a path might wildly increase the amount of information that propagates through the routing protocol, they might be kept only as optimizations: recall that source routing allows some packets to traverse good paths or, perhaps, to include more motivation for forwarding that bypasses (some types of) filters for solicited traffic.

The ability to place a filter like this into the routing protocol (or into another routing protocol atop the usual), however, becomes yet another matter of policy: because state is consumed for every filter and preference, the allocation of that state may have to be guarded. How to permit the manipulation of per-destination queuing and filtering is an open question we intend to address.

Applications and Policy As noted above, a common goal of network policy is to enable, prohibit, or charge for certain *applications*. Some applications exchange traffic that is performance-sensitive, causing a push for quality of service in the network that would ostensibly enable such applications. Other applications are more or less suitable for different networks: corporate networks are less likely to want to support file sharing or personal Web servers. Wireless providers sell "Web" for one rate, "corporate email" for another, and unfiltered access for a higher price. Unfortunately, the middle of today's network is a poor environment for understanding applications—filtering on transport-layer ports and payload inspection is increasingly difficult, particularly as the amount of encrypted traffic in the network grows.

This highlights another important tussle between users and network providers—one where users have the upper hand. That is, users can tunnel and encrypt to hide information from providers and other users. Providers, meanwhile, must turn to deep packet inspection, despite the fact that they have essentially no desire to see the content of the packets. Instead, providers just need to know *where* to forward, *why* they should forward the packet (which is a more general notion of whether applications are benign, whether a peering relationship exists, etc.) and *how* the user wants the packet treated. In fact, given this information, intermediate nodes can directly apply policy to decide whether and what kind of service to provide the packet.

3 A New Approach: Postmodern Networks

To enable the tussle [34] between users wanting greater control over how their packets are delivered and providers wanting control over how their networks are used, we propose a postmodern internetwork architecture. The postmodern internetwork architecture attempts to give providers the hooks they need to enforce policy while giving users the ability to provide enough instructions and documentation so that providers can and will forward packets as users request.

In view of the policy requirements outlined in the previous section, the postmodern internetwork layer consists of six functional blocks; each is provided space in the header (Figure 2).

A forwarding directive ("where") that instructs intermediate nodes how to direct or duplicate the packet. The directive might be as short as an address or as long as a source route.

Motivation ("why") that compels intermediate nodes to forward packets, which may consist of no information at all or a lightweight proof that the sender or receiver is a paying customer.

Accountability tokens ("who") that allow each conversation to be audited, in some networks to the precise authorized endpoints, in more anonymous networks, only to the level of large aggregates by following a chain of references. A "flow id" that indicates packets belong to the same conversation can be included in the accountability token.

Knobs ("how") that express cross-layer hints to lower layers about forwarding intentions, for example that packet would be better dropped than retransmitted, that it might expire after some time, or that it should be encoded with forward error correction. These also provide access to the queuing and scheduling mechanisms needed to implement end-to-end QoS.

Dials ("what") that recover information from lower layers, collecting, for example, the maximum observed loss rate along a path, the cumulative queuing delay, etc., that is needed to make informed decisions about congestion control, end-system path selection, session-level encoding method, etc.

Each block, as an element of the network layer, may be modified by intermediate nodes. For example, "motivation" may be added by providers on behalf of their customers. Forwarding directives may increase in detail through provider

networks, but remain coarse outside them. Accountability tokens may be replaced or transformed as packets cross trust boundaries, so that presenting a legitimate query (with probable cause in the form of evidence of abuse) to each operator is required to discover the original source. However, to preserve end-to-end semantics, each block may also have portion that the user can designate to remain unmodified end-to-end, which cannot be modified without detection.

3.1 Setting

The network consists of *links* interconnected by *nodes*. Links must support packet transfer, but may do so in any way using any underlying technology. We expect links to be symmetric or that they can be made to appear symmetric. Nodes may include single hardware devices or entire sub-networks; a node can be defined recursively as a set of links interconnected by nodes. We illustrate this recursive definition in Figure 1.

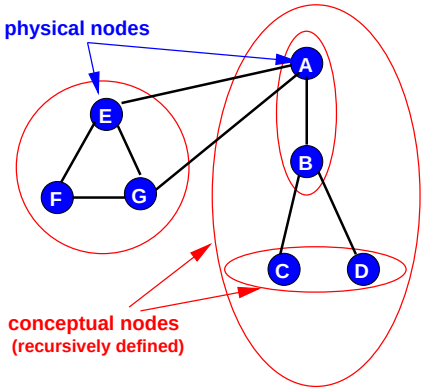

Figure 1: Network Abstraction: *Nodes* connect *links* and may define trust or realm boundaries.

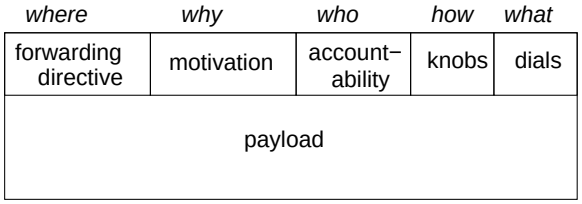

Figure 2: Postmodern header

The network is a hierarchy of nodes structured along the boundaries between organizations (departments, divisions, companies, consortia, etc.). Trust boundaries divide the network and may block the flow of control information and data packets. Trust boundaries typically align with node boundaries, but are not required to do so. *Providers* operate networks of links to provide a service (e.g., an ISP network, a wireless network, a university backbone, a sensor net, etc). We use *realm* to refer to these (virtual) network nodes that often define trust boundaries as well.[1] Because a realm is just a (virtual) node connecting links, the realm "as a whole" must support the functional components required by our postmodern internetwork architecture. Typically, this functionality will be supported at the edge nodes. A realm may, to connect links to other realms, use any underlying network technology to carry traffic: realms may, for example, be MPLS-like [83]. Nodes inside a realm that wish to be an endpoint of communication must understand and support the functional components needed for inter-realm communication, though they need not understand details of remote realms. The network infrastructure (in particular, links) is typically stable over time-scales of hundreds to thousands of seconds. However, our architecture is specifically designed to also accommodate links where this is not the case (e.g., links between mobile wireless nodes). In such cases, we need to provide for local mechanisms to "fix up" the problem without end-node action.

In addition to this conceptual hierarchy of realms and their interconnection, we make two assumptions about future network characteristics that will enable our design. First, packet header bits are not and should not be precious. Bandwidth is sufficiently cheap that the network should favor carrying more information in the header rather than sacrifice functionality. Second, network hardware is powerful enough to provide the flexibility needed to achieve the performance goals, provided some attention is paid to promoting parallelism and other performance requirements in the design.

3.2 Design Goals

Within this setting, we have overarching design goals that encompass both our desire for a minimal networking layer and for the behaviors of the mechanisms within it. Broadly, the goals follow from and elaborate upon our key principles: separation of concerns and explicit mechanisms to support policies.

[1] Terms for network aggregates are overloaded; realms here are not identical to NAT realms, Kerberos realms, etc.

- Support for any foreseeable policy requirement via explicit mechanism. This forces tussles to play out explicitly, under the control of the mechanism, as opposed to "under the covers."
- Complete isolation of routing and forwarding. Routing (path selection) may be in-band, out-of-band, or some of both. This provides maximum flexibility with respect to the amount and type of resources devoted to path selection.
- User control over inter-realm paths, within the constraints imposed by provider-specified policies. This allows for users to select paths to achieve quality or other policy objectives.
- Isolation of the basic forwarding mechanism from any kind of endpoint identifier. This makes it possible to use different forms of endpoint ID without modifying the fast forwarding path.
- Complete independence from hierarchical, topology-constrained identifiers (addresses). This obviates global numbering authorities, and lowers the cost of modifying the network topology (e.g., in mobile networks).
- Support late binding of provider policies to define which packets are forwardable, and how they are forwarded. This enables dynamic negotiation of compensation between users and providers, for example based on current conditions.
- Support single-packet exchange in one round trip.

These objectives, each of which (except the last) is new relative to today's Internet, would offer significant improvements in the elegance of design, in the ability to enable tussles to play out between competing interests, and in enabling new applications. They focus the following discussion of how we plan to build example mechanisms within the postmodern internetwork architecture for forwarding, accountability, motivation, and cross-layer knobs and dials.

4 Research Plan

In this section, we describe the three major project thrusts: the design and implementation of a forwarding mechanism that is independent of routing and does not require hierarchical addresses; the development of mechanisms to support policies governing accountability and forwarding motivation; and cross-layer mechanisms for performance and manageability.

In what follows, the term *user* denotes the entity that originates or receives packets; when a user has a role to play in choosing paths or as a principal in the network, we expect an application or kernel stack to make decisions on the user's behalf without explicit involvement. Every user has a provider, or realm. Typically this relationship will be established out-of-band, though on-the-fly relationships may be supported by some, for example, wireless, providers.

4.1 An Internetwork Forwarding Service

We propose a mechanism based on loose source routing as one of many possible means of providing forwarding directives within the postmodern internetwork architecture. Our approach *maximizes the separation between forwarding and routing*, so that tradeoffs are exposed and optimizations in each dimension can be exploited. IP, in contrast, entangles forwarding and routing: every node computes its own mapping from locator (IP address) to path, and that mapping is applied at every hop. IP "pre-pays" the cost of finding paths to *all* destinations from *all* nodes and then amortizes this cost over *all* packets. Thus, the marginal cost of path determination is the same for all packets. The IP approach works reasonably well in a homogeneous, relatively static network. But when paths in network change rapidly (e.g., mobile nodes), when packet transmission is expensive for some (battery-powered) nodes and nearly free for others, or when applications require highly constrained paths—more flexibility in the "amortization schedule" is needed.

A second advantage of our approach is that forwarding does not require hierarchical identifiers linked to topology. Consequently, concerns about scalability and global administration of namespaces are confined to various resolution and bootstrapping mechanisms, and do not interact with the "fast path."

4.1.1 Forwarding Mechanism

The forwarding mechanism is based on the following postulates:

- Every *link* (channel between forwarding elements) has a globally unique linkID that need not be tied to topology.
- Each forwarding element knows the linkIDs of directly attached links.
- The forwarding directive in each packet specifies the links the packet should traverse. For unicast, the forwarding directive is a sequence of linkIDs; for multicast, it is a tree of linkIDs.

Each forwarding element (FE), upon the arrival of a packet, parses the header to (i) verify that link on which the packet arrived is consistent with the header, and (ii) determine from the forwarding directive the link(s) over which the packet should be forwarded. If the forwarding directive is complete, the next link(s) are directly attached to the FE and the packet directed to each output link specified. Some linkIDs in the forwarding directive are virtual links that tell how to *demultiplex* the packet to an application or transport protocol at the destination. Before transmission, the header is updated to record the link traversal. If any specified next link is *not* directly attached to the FE, a *forwarding fault* occurs and the packet is diverted to a fault-handler, which fills in the "gap" in the path, using routing information. We describe forwarding fault handling in Section 4.1.3.

This mechanism achieves our goal of separating routing and addressing mechanisms. It also allows forwarding directive information to be *cached*, thus supporting our goal of allowing the cost of path determination to be amortized over multiple packets. Such caching might be accomplished in several ways; we propose to explore two. The first is simply to record the actual path traversed by the packet in the header (as in the IP record route option). A second option is to cache the forwarding directive information at each FE. Each FE would store a mapping between the incoming and outgoing link(s) in *ephemeral state* [25], where this mapping would persist for a short, fixed duration then disappear. This would essentially implement tag switching, but with bindings that disappear from the switch after a short time and are not refreshable. The advantage of this approach (over soft-state) is that it allows the lookup mechanism to be designed to meet a worst-case throughput requirement. Setting the lifetime to maximize the expected number of times a path can be re-used before it expires is an engineering decision.

Both caching approaches allow the user to trade off the risk of the path breaking against the cost of re-determining the path. Explicit caching in the packet gives the user control over how much of the path is re-used. Paths consisting of links with stable performance might be re-used for a long time. Paths that include mobile links, on the other hand, might not be re-used at all. Meanwhile, the "dials" mechanism (Section 4.3) provides the information needed to evaluate this tradeoff quickly: path performance and stability information that does not depend upon the experience of many packets.

Bidirectional paths would allow knowledge accumulated in forwarding a packet through the network to be used by any reply. To make this possible, it is sufficient to postulate that all links in the network are bidirectional, then rely on forwarding faults to patch routes that cannot be made symmetric. Some kinds of links that we want to support, such as satellite and wireless links, are not inherently bidirectional, but asymmetries may be patched by forwarding elements transparently to the end systems. Whether this approach is reasonable for all of the interesting situations is one of the research issues to be investigated.

The remaining questions are: how forwarding directives are obtained and constructed, and how forwarding faults are handled. To answer those questions, we first consider the bigger picture.

4.1.2 Model of Communication

The general communication process involves the following parts and steps (Figure 3):

1. The *objective* of the communication (an informal notion representing what the user wants to accomplish) is resolved into a *destination specification*, which is identified by a token that is independent of location in the network. (In today's network, a typical destination specification is a URL, an email address, or a DNS name; the "resolution" is often handled out-of-band, but may be performed interactively, for example, through Google.)
2. The destination specification is resolved to a *locator* , which is determined by the destination's location in the network. (In today's network, the locator is the IP address and this resolution step is handled by the DNS.)
3. The locator is resolved to a *path* through the network. The path depends on the locator, the source's location, and the policies of the providers whose networks are used by the source and destination. (In today's Internet, this resolution is performed hop-by-hop as the packet travels through the network. Each node determines the next hop in the path according to its own policies and the destination IP address.)

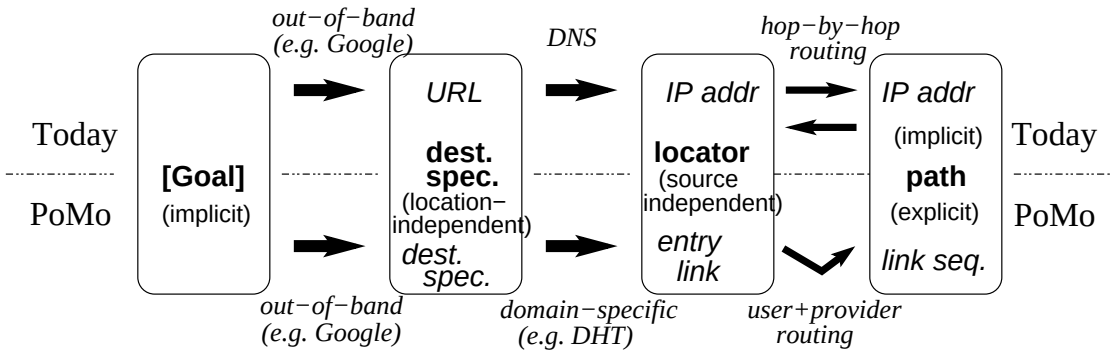

Figure 3: Communication steps

In our architecture, the first step would remain more or less unchanged; as in the current architecture, we expect to support various forms of destination specification, possibly with their own resolution infrastructures. The locator produced in the second step would be the linkID of an entry link into the realm containing the destination. It is determined by the topological location of the destination in the network, but need not be identified by a hierarchical address. The resolution mechanism for this step provides a mapping from destination specifications to (realm, ingress link) pairs. This mapping might be accomplished using a hierarchical system like the DNS, but more likely would use a global DHT operated by cooperating providers.

In the third step, as in today's network, the task of determining a path to the specified location in the network is divided between the originating user and the providers of realms through which paths between source and destination pass. Our paths can be considered to consist of a sequence of three segments. (We consider linear paths here; extending to trees is straightforward.) The *exit segment* includes all links from the originator to the egress link of the originating provider. The *entry segment* consists of the link that enters the destination provider and all subsequent links. The *transit segment* consists of everything between the exit segment and the entry segment.

The transit segment will be at most partly specified when the packet leaves the originating realm; it starts out as a specification of the realms to be traversed by the packet en route to the destination realm; each realm is specified by an entry and exit link. This sequence of realms may be determined by the user through some out-of-band means, or the user may choose to allow the originating provider to select the transit path. As the packet enters a realm, a forwarding fault occurs and a path to the designated exit link is determined according to the traffic engineering goals of the provider network. The resulting path segment is added to the forwarding directive. Each provider can use a mechanism of its choice to manage transit paths within its network.

The entry segment is determined by the destination provider once the packet enters the destination realm. We envision two ways for this to happen. One is for the receiving user to provide the originating user (through some out-of-band means) with a *path token* that specifies the path from the entry link to the destination. The other is for the destination provider to perform the resolution in-band, when the packet enters the destination domain. For this to be possible, the forwarding directive must contain the destination specification.

Finally, the exit segment is obtained by resolving the originating domain's egress linkID, either using routing state at the originating node, or the provider's mapping system.

4.1.3 Forwarding Faults: Offloading Path Resolution

Forwarding faults occur for one of two reasons: (1) path resolution and refinement is (implicitly) requested by the source, or (2) the specified path contains an error. Because realms are defined recursively, the path resolution step can be applied recursively, distributing the work of determining the complete end-to-end path to the sequence of realms on the way to the destination. This does *not* require hierarchical addresses; it suffices, for example, for nodes to know the topology (including linkIDs) within only their realm.

Faults resulting from a path error can support useful features such as mobility. Example path errors include dead links, malformed paths that include non-existent links, paths that do not comply with network policy, packets headers with insufficient motivation tokens to use the requested link, or the forwarding behavior requested by the knobs is unavailable. One response to a path error is to return an error message to the sender so that it can select a new path

and update its routing state. A friendlier alternative is to have the network find a route that will get the packet to its destination. Because FEs must handle "normal" forwarding faults, there is little harm in reusing the mechanism for error cases. When transmitting packets, sources specify which of these behaviors they desire, akin to the "don't fragment" bit in IP.

Packets that experience forwarding faults are diverted from the "fast path" for handling; one of three cases applies: (1) the node knows the realm-specific routing information needed to get the packet to its next hop and modifies the forwarding directive appropriately, or (2) the node knows a node to contact that can map the destination specification (carried in the forwarding directive) to an intra-realm path (e.g, a realm-specific DSR service) that will reach the destination, or (3) the node forwards the packet to a node that has more knowledge and can assume the responsibility of forwarding the packet toward its destination.

Another feature of this type of incremental address resolution is that it permits dynamic route adaption to enable features such as mobility, load balancing, server selection, and subcasting—all of which can be controlled by the receiver's realm. For example, if a path fails because a mobile receiver moved, the faulting node can use either option (2) or (3) to automatically discover a new route to the destination.

4.1.4 Routing Protocols

The goal of a routing service is to map a destination location to a path for the forwarding layer. Because a key design goal is separation of concerns, we implement routing as an independent service. Because the network is organized into hierarchical realms, conventional hierarchical routing approaches similar to those found in the Internet can be applied, but because our forwarding mechanism uses linkIDs, the routing protocol design differs significantly. In particular, our routing protocols are concerned exclusively with collecting and disseminating *topology* information.

- An inter-realm protocol, exchanging sufficient path information to construct at least policy-compliant paths to entry segments. Policy enforcement is coupled with forwarding, not routing, because any end node can construct a route. The simplest implementation of an inter-realm protocol would start from BGP, substituting LinkIDs for autonomous system numbers.
- An intra-realm protocol fills "gaps" left in source routes computed from inter-realm routing information. Because realms can be arranged hierarchically, the routing protocol may need to act both as an intra-realm protocol and an inter-realm protocol. Depending on the size of the realm, conventional link-state methods may be used, optionally with traffic-engineering overrides to fractionally route some traffic away from the shortest paths should links become over-utilized. A realm may chose to keep its internal paths private, open them to the public, or encrypt their routes before making them visible outside the realm. In the latter case, the route only becomes meaningful when it is being resolved inside the realm.

Each realm has discretion over whether the intra-realm routes are erased from the forwarding directive as it leaves the realm (thus preventing re-use for packets going the opposite direction). Although this decreases efficiency, the secrecy involved may be valued. It may also be valuable in situations where underlying network makes maintaining virtual symmetry too expensive.

4.1.5 Research Tasks in Forwarding

Our work has three major components: First, we will implement the forwarding engine on various platforms, including the linux kernel. Second, we will build the necessary fault-handling and topology-exchange mechanisms for both inter- and intra-realm use, as described above. Finally, we will experiment with a DHT-based service for resolving destination specifications from an unstructured space to entry-link locations. The implementation of a DHT using our link-based forwarding service, without the global connectivity provided by IP, will be an interesting research challenge.

4.2 Motivation and Accountability

In the thin network layer, packets contain explicit *motivation* and *accountability* headers that describe *why* the packet should be forwarded and *who* is responsible for that packet. Both of these statements require principals, identities, and certification to convincingly demonstrate that a packet was or should be handled by a particular entity. Part of our proposed work is to understand what forms of identities *must* be exposed to the network. Through our research, we expect to better understand the following fundamental questions:

- Should network-visible identities represent processes, users, hosts, networks, or some other aggregate? What identities must be carried with every packet, and which can be amortized across streams of packets? What are the tradeoffs in performance, security, and flexibility?

- What new forms of end-to-end accountability are enabled by having strong identities in packets? Can actions on behalf of packets be accounted for across realms?

In the postmodern network, a node will forward a packet only if the packet contains explicit "motivation" bits, but the packet may not need explicit source or destination addresses. This is the opposite of the current Internet, in which forwarding motivation is implicit but an explicit "identity" of an IP address is carried in each packet. Because this identity is trivial to spoof, many network-layer security violations are possible, including DoS attacks.

Identity makes it possible for senders and networks to populate the motivation fields and the accountability fields with unforgeable tokens. To provide these identities, however, requires addressing the following three questions: Who issues the identities? What principal is identified? And, how are principals identified?

Consider an extreme design point in which each packet carries a cryptographic identity that would enable network providers and destinations to unambiguously identify "senders" and the "network path" taken by the packet. We quote both senders and network path because both can be specified at different granularities, from individual host processes to whole networks at the sender and from network interfaces to realms along the path. Different granularities may be easier, for example, allowing a host to offload motivation to its network provider, or may be in keeping with provider concerns of keeping a secret, autonomous network where only high-level information about network paths can escape.

Identities could conceivably be provided hop-by-hop with each pair of adjoining realms setting up a symmetric key. Such a solution would of limited use since it either would require each realm to share a key with all other realms, which does not scale, or identities would only be valid in an adjoining realm. Such scoped identities provide some benefit—it can capture customer-provider and peer relationships—but is ultimately unsatisfactory because end-hosts cannot reliably identify "who" originated a packet beyond asserting which of their upstream ISPs the packet came from.

Hence, we claim that any useful packet identification scheme must rely on asymmetric keys. To be useful, asymmetric keys require some form of certification authority and PKI (or a key generation authority if Identity-based Signatures are used [20, 46]). We consider these different approaches next.

Identity Authorities for a Global Network Assume that each entity (hosts or realms) generates the public key it uses to sign packets. A single global certification authority (CA), possibly an analog to IANA, could be used to certify public keys, and these certificates could be carried along with signatures on packets. Such a monolithic solution is likely to be unsatisfactory because all realms would have to trust a single global authority. Such an authority would have immense power to decide who may access the network and at what price. The authenticity of every identity in the network would depend on the single authority's correct and honest behavior. In a CA-based solution, trust in an identity is by fiat: identities are trusted since the authority is trusted. Such a solution is intrinsically fragile, since the compromise of a single site or key can devastate the entire identity system, leading us to consider decentralized authorities.

Decentralized Authorities A completely decentralized identity authority will have no single point of failure and will not require trust by fiat. One possible solution is to use a system in which authority is inferred using trust and reputation, akin to a global deployment of PGP [110, 22]. Reputation systems and trust metrics are an active research topic [59, 69, 21, 71, 37, 50], and research prototype PKIs based on trust have been built [27, 73]. Part of our work will be to understand if and how this research can be pushed down to the network layer, where concerns of latency and scalability pose far greater problems.

We will also investigate hybrid authority systems in which a set of well-known principals (e.g., tier-1 ISPs or some other entities) establish a clique which then certifies all realm identities using threshold signatures. This scheme requires that users (or other principals for whom keys are generated) trust a few of many possible clique members. As long as the thresholding scheme is constructed so that no certificate can be generated without involving at least one trusted clique member, all users can trust all keys, without necessarily trusting all of the signing clique. Finally, we will also experiment with thresholded hierarchical identity based signatures [2] since this would allow network entities to verify signatures without requiring globally trusted entities or a PKI.

Authenticating Packets Next we assume that identities are certified, using either a centralized or decentralized authority, and focus on questions of which entities are identified and the overhead of carrying packet signatures.

The first question to address is that of which principal should be identified in every packet? In the network layer, is it sufficient to identify realms, or should higher layer entities be visible at the network layer? Finer-grained identities are useful for actions that might require higher resolution, for example, blocking individual hosts that are part of a botnet instead of blocking entire realms.

We will investigate to what extent identities from *higher layers* help with accountability within the network. For example, consider network architectures, such as TVA [109] and Off-by-Default [4], that require applications to acquire credentials before forwarding packets. Application-layer directives (e.g., whether a particular connection should be allowed or not) are mapped onto objects (credentials) that are interpreted inside the network, and these credentials are used to stop unwanted traffic close to the source. In general, adding higher-layer identities will enable the network layer to parse and act upon higher-layer directives. Although such an architecture may not conform to the usual norms of layering, providing flexible mechanisms for stopping unwanted traffic close to the source is sufficient justification for rethinking network visible identities.

Generating and carrying public key signatures and associated certificates will incur significant (possibly untenable) overhead. Different mechanisms may reduce this overhead: instead of signatures per packet, only control packets or connection-initiation packets could contain signatures. Alternatively, signatures could be "spread" over multiple packets, such that an end-host (or a logging router) would have to gather a few packets before it could verify a complete signature. Both these alternatives have been studied: TVA [109] signs control packets to generate credentials, while EMSS [77] amortizes signature overhead by periodically generating packets that contain the signed hash of a few prior packets (which are sent without signatures). Both solutions require per-entity state inside the network and additional mechanisms such as fair queuing to guard against DoS attacks against the verification process.

We propose to investigate efficient mechanisms for carrying packet signatures, including schemes that spread signature over time (sign only a few select packets) or over space (partial signatures carried in all packets). Current hardware and router economics prohibit carrying strong signatures in every packet. However, we expect our research to lead both to a better understanding of the benefits that accrue from having strong identities in packets, and to new mechanisms for efficiently proving identities in packets.

Identity and Anonymity There is a fear that demanding strong identities in each packet will preclude anonymous communication. However, this is only the case if the underlying protocol requires source spoofing (which no reasonable protocol does). It is important to understand that in anonymous communication protocols, users who wish to be anonymous choose a pseudonym (typically a public key). The adversary is assumed to know the set of pseudonyms and the set of hosts participating in the anonymity protocol. The primary goal of the protocol is to keep the mapping between host and pseudonym secret. Other properties that the protocol maintains include keeping the source of a message secret from the receiver (sender-anonymity), the recipient of a message secret from the sender (receiver-anonymity), and the fact that two parties are communicating secret from an observer (sender-receiver anonymity). However, in no case is the set of hosts participating the protocol kept secret, and thus, a properly designed anonymous communication protocol will maintain its security properties even if every packet is marked with a network visible user signature.

A slightly different form of anonymous communication enables users to access public services without disclosing their identities. These protocols use trusted third-party proxies (or sequences of proxies) to access the service and return content to the anonymous user. Once again, these protocols can be used without modification and without compromising their security properties.

Accountability Network-wide identities enable strong end-to-end accountability. Assume that each network-level entity (host/router) has a two-part signature: the first part identifies its realm and the second part identifies the host/router within the realm. Such signatures could be added to the accountability headers in the postmodern network to to generate a *path signature*, a record of the path a packet traversed by a packet. Such a path has two properties: (1) it unambiguously identifies each realm traversed by the packet, and (2) ISPs/network providers still maintain complete control over how much of their topology is divulged. The first property holds since a rogue ISP cannot forge a signature that does not belong to its realm.[2] The second property follows since ISPs can (presumably) forge any signature *within* their domain. We next describe use of path signatures (network-wide whitelists and blacklists) that we propose to implement and conduct experiment with.

Filtering and Credential-based Forwarding Path signatures described above can be used to push filters (blacklists) into the network [76], and can also be used to implement credential-based access to protected resources. We explain

[2] Rogue ISPs may not put a signature on the packet, but this can be detected at the edge of the rogue ISP's network.

with an example: suppose host V (the victim) receives packets from host A (the attacker). The packet contains the path signature which V can use to determine the nodes (not necessarily particular routers, since ISPs may choose to aggregate parts of their topology) traversed by A's packets. V can then request its ISP that all packets that match the path signature be dropped. V's ISP, with whom V has a financial contract, will honor this request, and may choose to propagate the filter further. Since the path signature cannot be forged, the offending packet stream and associated upstream providers are identified unambiguously. The upstream providers may recursively choose to block this traffic, since it reduces transit traffic on their network. Interestingly, A's provider may choose to *not* propagate the filter request; doing so allows the provider to charge for packets sent by A. In this simple case, the victim (V) needed to block packets from only one distinguished host (A); however, if multiple hosts in the victim domain are compromised, the attacker can generate packets with many path signatures that share the same suffix. V may choose to filter based on a suffix that blocks hosts from the entire offending realm unless they include a credential carried in the motivation header. The motivation header provides a "dual" to the accountability header, and these two functional blocks allow us to seamlessly implement both default-on and default-off networks with fine-grained control. In general, the motivation header can be used to subsume (and implement) credential-based forwarding schemes directly at the network layer including TVA [109], SOS [62], Mayday [1], and SIFF [107].

4.3 Knobs and Dials

Although layered architectures have the benefits of modularity and preserve the independence of design choices, opaque layer boundaries are now thought to hide too much, resulting in implicit mechanisms that make the wrong assumptions, preventing straightforward performance optimizations and complicating diagnosis and management. Replacing opaque boundaries with *translucent* boundaries can enable inter-layer control loops that dramatically improve performance, as well as enhanced monitoring, control, and management capabilities [33, 99, 65].

The canonical example of implicit mechanisms is embodied in TCP, which, because the underlying links layer characteristics are completely hidden, assumes that loss events are caused only by congestion and never by corruption [56]; when it runs over wireless lossy (but uncongested) links, performance suffers because the response (backoff vs. retransmit) cannot explicitly matched to the stimulus (congestion vs. corruption). There has been significant research activity concentrating on TCP over wireless links [64, 3], with mixed results at best and negative results at worst (e.g. [60]). Our previous work [64] has shown that an order-of-magnitude improvement bound is theoretically possible if congestion losses can be discriminated from corruption losses so that TCP takes the proper action: back off only if congested.[3]

Cross-layer optimizations allow lower layers to express their characteristics via *dials* to upper layers, which in turn can exert control downward on lower layers through *knobs*. The key aspects of our postmodern internetwork architecture are separation of concerns and explicit mechanisms. Knobs and dials allow control information to be *explicitly* conveyed vertically across layers, without loading assumptions and implicit behavior into the protocols.

The vertical control require *horizontal* signaling within a layer. For example, an end-to-end service may wish to discover a characteristic of a path, for example, available capacity [102, 57]. At each forwarding hop, the forwarding element transforms the dial field to lower the recorded available capacity when a bottleneck is reached. Such signaling can assist network monitoring and management, and also allow applications to monitor service [75].

Dials allows cross-layer optimizations for performance enhancement: the dial field can convey channel conditions and errors at the desired granularity (per packet or per flow) so that end-to-end protocols can use the proper error and transmission rate control functions. The application can report its reliability requirements (policy) via knobs, which can be used by the network to adjust the error control mechanisms applied.

Cross-layer signaling may be in-band, in which a data packet accumulates information about the path it traverses, or out-of-band, in which probes recover the characteristics of network paths. Because we support knobs and dials in every packet, our approach supports both in-band operation and out-of-band: probes are simply packets with empty payloads. This unifies data transport and signaling into a single, simple mechanism. The potential benefits of knobs and dials are an improvement in performance of the application and manageability of the network service. But there are also *costs* measured in the complexity of signaling and state managed, and the possibility of reduction in *overall* performance due to side-effects. This has been one of the problems with current ad hoc tweaks to existing protocols that lack the architectural framework to implement knobs-and-dials in an understandable manner.

We plan to understand and quantify these tradeoffs by exploring and implementing points in a layer vs. mechanism design matrix[98], and verifying current simulation of this framework. The two key horizontal protocol functions are

[3] Our work [64] also showed that this theoretical bound is difficult to achieve in practice when constrained by the backward-compatible TCP/IP semantics specified in RFC 1122.

error control and transmission rate control. At a given layer, these functions can be implemented as closed-loop feedback (e.g. ARQ), open-loop (e.g. FEC), or not at all. The question them becomes: *Given a protocol function, at what layer(s) should it be implemented* (end-to-end, hop-by-hop, or both), *using what mechanism* (open vs. closed-loop), *given network characteristics* (e.g. path loss characteristics), *and constrained by application policy* (e.g. reliability requirements)?

We will implement an end-to-end protocol (initially, an augmented UDP with postmodern shim headers and configurable error and transmission control) that supports the postmodern internetwork architecture header in general, and knobs and dials in particular. We will use a variety of applications to test user-level performance, including Web browsing and media streaming. Hard problems that we plan to tackle within this framework include how to model and analyze the multivariate complexity of these mechanisms, how to measure and understand bad side-effects and (feature) interactions of local optimizations, and what are the *minimal but necessary* set of knobs and dials. Further, by implementing and evaluating cross-layer optimizations within our postmodern internetwork architecture implementation, we hope to understand the policy issues in enabling knobs and dials across realms.

5 Research Method

Our research philosophy puts great weight on empirical evaluation: we will build implementations, "live" in the network we construct, and learn about the limitations of our designs with experience. To "live" in the postmodern internetwork architecture, we must use it for everyday tasks: porting existing applications and gatewaying them to the rest of the Internet where needed. This is our key goal and benchmark for success.

Using an implementation of the new architecture provides several benefits not achievable through other means. Our ability to separate routing, forwarding, and addressing can be compellingly demonstrated only by implementing different mechanisms. Ideas for how to support new applications by exploiting new capabilities in network-layer knobs, dials, and accountability will come only from experience. Finally, by migrating existing applications to the new network layer and running all atop a conventional network, we hope to show the architecture's ability to encompass these and future technologies.

5.1 Implementation and Evaluation

We now summarize our evaluation plan, which consists of Emulab-based [106] kernel implementation development and testing, PlanetLab-based [78] wide area deployment, and interfacing with the network to live within it. We expect that the GENI [45] environment would contribute more realistic reporting of faults and control over virtual links for our knobs and dials components that, when running as an overlay, may have limited flexibility. For example, overlay links may be able to probabilistically achieve a specific loss rate using forward error correction, but are clearly unable to guarantee any high level of service. Such limits demand imagination to infer what would happen given more control, but there is much experience to be had even using virtual overlay links.

Emulab Experiments

Emulab is a network testbed environment developed by the University of Utah that supports experimentation with kernel modifications, especially those that affect network stack behavior. The Emulab facility located at the University of Kentucky is managed and operated by the PIs of this proposal and will serve as an excellent resource for the initial phase of our development and testing. In particular, we will use Emulab to test a kernel-level implementation of our thin network layer. We expect to supplant the IP stack of the kernel by tying our own TCP- and UDP-like transport protocols directly to the thin network layer. The kernel must include the forwarding engine for performance; fault-handling may be provided by a user-level process that exchanges routing information on-demand to manipulate ephemeral forwarding state within the kernel. The controlled environment offered by Emulab allows us to reproduce experiments to isolate problems and unexpected behavior.

PlanetLab Experiments

A second, partially concurrent, phase of our evaluation will target our thin network layer implementation to user-level processes on PlanetLab. PlanetLab offers real, unexpected behaviors [96, 82] such as failures, resource limits, arbitrary processing delay, memory corruption, contention with real users, and misbehavior, making it a challenge platform for performance-sensitive distributed services.

PlanetLab will help us experiment with knobs and dials across virtual network links between nodes, inter-domain policy restrictions such as not using Abilene (Internet2) for commercial endpoints, accountability tokens that trace any perceived-to-be-abusive traffic back to an ingress point and user, inter-realm routing in which different PlanetLab sites (or groups of universities attached to the same aggregate network like CalREN or the Pacific Northwest GigaPop) act as realms.

A particularly powerful component of the thin network layer will be the domain-specific endpoint resolution layer that maps a destination specification to an entry link or links. We call this the *destination specification resolution (DSR)* service. Various destination specifications could be implemented, each with different addressing abstractions. For example, we plan to implement a Speccast[80] DSR based on abstract-able predicates, a geospatial DSR service based on location[55] (e.g., GPS coordinates), a hierarchical name-based DSR service (e.g., similar to DNS), and an IP-based DSR service. The IP-based service will allow us to easily port network applications written to the sockets interface with IPv4 addressing, likely without even recompiling the application. An IP-based DSR need not guarantee that nearby IP addresses are nearby in topology: a flat routing protocol is possible. The facilitates more flexible IP address to host assignments and can also support mobility efficiently.

Experimentation with PlanetLab will doubtless yield valuable experience in itself, but it is a means to our true benchmark of success, living within the network.

Living with the Network

Living in the network is a challenge. The architecture, to support our demanding use patterns must support our applications, interface sanely with the outside world, support mobility, and be extensible in the applications and services that can be discovered. To "live in our network" is a strong statement that it supports the needs of today, with the security, manageability, and flexibility we have designed in for future evolution.

One approach to integrating the old and new networks will create a new realm representing the current Internet that has multiple links into our thin network layer network (i.e., PlanetLab nodes). Unlike IPNL [43], the legacy Internet bears no special role aside from providing service to legacy content and users.

While connecting from a customized network to the outside requires very little application work—simple address translation at edges suffices—application-level gateways, such as mail servers and Web proxies, will be needed to provide access from the legacy Internet to services provided by machines in the thin network layer network.

By connecting our thin network layer network to conventional Internet services, we will be able to "live in our network"—to carry out everyday activities such as sending and receiving email, surfing the Web, and serving up Web pages. The IP-based DSR service will, given an IP address, return a nearby PlanetLab node, either to the source (least likely to choose a bad path) or to the destination (providing the greatest flexibility); that node's link into the IP realm will be the entry link in the forwarding directive. Similarly, traffic headed into the thin network layer from the IP network can be directed, via DNS tricks or application-level gateways, to enter the thin network layer at chosen entry points, although we do not expect to innovate in ways to trick the legacy IP network into performing well. By forcing real-world traffic the IP realm to enter and then traverse a sizable portion of the thin network layer network before it reaches its intended destination, we create an environment in which policies can be deployed to prevent, redirect, or give preference to certain packets—using real-world traffic. Servers located within the thin network layer can use network layer information to embed detailed accountability information in mail messages to trace and black-hole spam, can use network layer knobs and dials to block connections from misbehaving subnetworks, can use detour routes to avoid performance bottlenecks, etc.

5.2 Success Metrics

Our thin network layer architecture posits several novel mechanisms that run contrary to conventional wisdom. Consequently, a key metric will be whether they in fact allow tussles to be played out in policy space. **First**, we hope to show that both user and provider policies can be accomodated by composing and resolving forwarding directives created by opaque routing services and filled in by intermediate realms. Routing and forwarding without explicit node addressing is a powerful, but still unverified, primitive, central to our assertion that addressing does not belong in a network layer. Consequently, we need to show that it can be done and then measure the overhead/costs that result. **Second**, we hope to demonstrate the ability to dial-in choices along tussle-spaces, including defining configurable levels of accountability, privacy, routing control, quality of service levels, filtering, etc. Although we feel the proposed mechanisms offer the functionality needed to play out these tussles in policy space, achieving "configurable" control may require new or different sets of mechanisms, not simply a different parameter setting. Our benchmark is to demonstrate support for

representative classes of applications including financial (high accountability), medical (high privacy), mobility (low routing overhead), and filtering (customized network policy). **Third**, we hope to show how to exploit knobs and dials in policy-compliant ways to make forwarding behavior suitable for demanding interactive applications under dynamic conditions. The question is not whether the network can offer a particular level of service, but rather whether it can provide applications with sufficient information to enable the application-level decisions and adaptations to achieve desired performance. **Finally**, the ultimate metric will be our own contentment. Our implementation-based "eat our own dog food" evaluation philosophy will act as a primary, and final, metric of success.

6 Related Work

This section deals with prior research generally related to network layer architecture; work related to specific topics (such as identity) has been presented in earlier sections.

Nimrod [28] was an early re-envisioning of Internetwork architecture. It has several aspects in common with our general approach, including optional out-of-band destination resolution (via separate query-response protocol) before forwarding a packet. However, it did not specify any mechanism for dealing with inter-domain policy constraints.

Numerous attempts at network layer enhancement, extension or replacement incorporate loose source routing. Among these were various proposals for IPng, including Pip [41], SIP [36], and SIPP [42]. IPNL [43], an extension of IPv4 to deal cleanly with Network Address Translation (NAT), uses fully-qualified domain names to designate intermediate hops. Triad [29], an extension aimed at content-based routing, builds on the separate Wide-area Relay Addressing Protocol (WRAP), which adds a layer on top of IP that provides source routing via a "potentially opaque" token, which could be implemented as a sequence of IP addresses. WRAP relies on IP as an underlying forwarding mechanism.

The NewArch project [31] has produced several excellent architectural analyses and proposals. Our approach resembles NIRA, the New Internet Routing Architecture [108], in both goals and components. Its notion of provider compensation is similar to our motivation (Why) block, and it permits the use of caching. However, like most other proposals, it relies on hierarchical provider-rooted addressing. Our proposal is similar to FARA [32] in that we also propose a general framework, within which many different schemes for identifying destinations can be instantiated, and we separate forwarding from routing.

Our own work on Speccast proposes a network layer supporting a more powerful mechanism for specifying destinations. Our initial work [80], does not separate forwarding from addressing and routing. However, our more recent work (as yet unpublished) is based on a path-determination mechanism that does not require hierarchical addresses. (We intend to implement a Speccast-like destination specification capability for our postmodern internetwork architecture.)

A good deal of work has been done on different models of routing in the context of sensor and ad-hoc networks [51]. Many of these systems [54] target specific problems and service models. Many have in common with our approach that the cost of the (expensive) initial communication is amortized over subsequent packets by the establishment of forwarding state.

A variety of overlay-based network-layer services have been proposed in recent years. Nearly all of them assume an underlying IP-like service, i.e., direct connectivity among all nodes in the underlying layer, and therefore inherit the problems of the current architecture. *Indirection* has been identified as a powerful primitive for developing services based on a shared infrastructure. The seminal Internet Indirection Infrastructure (i3) [100] can be used to implement several conventional services using only unicast underneath. It achieves its power by routing over a distributed hash table network (such as Chord [101]) to a rendezvous point.

References

[1] David G. Andersen. Mayday: Distributed filtering for internet services. In *USENIX Symposium on Internet Technologies and Systems*, 2003.

[2] Joonsang Baek and Yuliang Zheng. Identity-Based Threshold Signature Scheme from the Bilinear Pairings. In *International Conference on Information Technology: Coding and Computing (ITCC'04)*, 2004.

[3] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997.

[4] Hitesh Ballani, Yatin Chawathe, Sylvia Ratnasamy, Timothy Roscoe, and Scott Shenker. Off by Default! In *Proceedings of the Fourth Workshop on Hot Topics in Networking (HotNets)*, 2005.

[5] S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Efficient peer location on the Internet. *Submitted to Journal on Selected Areas of Communications (JSAC) Special Issue on Recent Advances in Service Overlay Networks*, 2002.

[6] S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Scalable peer finding on the Internet. In *Proceedings of Global Internet Symposium, Globecom*, November 2002.

[7] Suman Banerjee and Bobby Bhattacharjee. Scalable Secure Group Communication over IP Multicast. *JSAC Special Issue on Network Support for Group Communication*, 20(8), Oct 2002.

[8] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommreddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM*, 2002.

[9] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. *Proceedings of INFOCOM*, 2003.

[10] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. 'omni: An efficient overlay multicast infrastructure for real-time applications. *Accepted for publication in* Special Issue of Computer Networks on Overlay Distribution Structures and their Applications., 2005.

[11] Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, and Aravind Srinivasan. Resilient multicast using overlays. In *Proceedings of ACM SIGMETRICS*, 2003.

[12] Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, and Aravind Srinivasan. Resilient multicast using overlays. *ACM Transactions on Networking (to appear)*, 2006.

[13] Suman Banerjee, Seungjoon Lee, Ryan Braud, Bobby Bhattacharjee, and Aravind Srinivasan. Scalable resilient media streaming. In *NOSSDAV*, pages 4–9, 2004.

[14] Bobby Bhattacharjee, Sudarshan Chawathe, Vijay Gopalakrishnan, Pete Keleher, and Bujor Silaghi. Efficient peer-to-peer searches using result-caching. In *The 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, February 2003.

[15] Samrat Bhattacharjee. *Active Networks:Architectures, Composition, and Applications*. PhD thesis, Georgia Institute of Technology, July 1999.

[16] Samrat Bhattacharjee, Ken Calvert, and Ellen Zegura. Active Networking and the End-to-End Argument. In *Proceedings of ICNP'97*, 1997.

[17] Samrat Bhattacharjee, Kenneth L. Calvert, and Ellen W. Zegura. Self-Organizing Wide-Area Network Caches. In *Proceedings of IEEE Infocom'98*, 1998.

[18] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, 1998. RFC 2475.

[19] M. Bond, C. S. Dhillon, J. Griffioen, and K. Calvert. Designing Service-Specific Execution Environments. In *Proceedings of the 2002 International Working Conference on Active Networks (IWAN 2002)*, December 2002.

[20] D. Boneh, H. Shacham, and B. Lynn. Short Signatures from the Weil Pairing. In *Asiacrypt*, 2001.

[21] Raouf Boutaba and Alan Marshall. Management in peer-to-peer systems: Trust, reputation and security. *Computer Networks*, 50(4):469–471, 2006.

[22] J. Callas, L. Donnerhacke, H. Finney, and R. Thaye. OpenPGP message format, May 1996. RFC 1951.

[23] K. Calvert, J. Griffioen, B. Mullins, S. Natarajan, L. Poutievski, A. Sehgal, and S. Wen. Leveraging Emerging Network Services To Scale Multimedia Applications. *Software Practice and Experience*, 33:1377–1397, 2003.

[24] K. Calvert, J. Griffioen, B. Mullins, L. Poutievski, and A. Sehgal. Secure Customizable Many-to-One Communication. In *Proceedings of the Sixth International Working Conference on Active Networks (IWAN 2004)*, October 2004.

[25] K. Calvert, J. Griffioen, and S. Wen. Lightweight network support for scalable end-to-end services. In *ACM SIGCOMM 2002*, pages 265–278, August 2002.

[26] K. Calvert, S. Venkatraman, and J. Griffioen. Authenticated Access to Reserved Network Resources. *International Journal on Network Security*, 2006. (to appear).

[27] Srdjan Capkun, Levente Buttyán, and Jean-Pierre Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Trans. Mob. Comput.*, 2(1):52–64, 2003.

[28] Isidro Castineyra, Noel Chiappa, and Martha Steenstrup. The nimrod routing architecture. Internet Engineering Task Force Request for Comments RFC-1992, August 1996.

[29] D. Cheriton and M. Gritter. TRIAD: A new next generation internet architecture, March 2000.

[30] W. Cheswick, S. Bellovin, and A. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker (Second Edition)*. Addison Wesley, 2002.

[31] D. Clark et al. Newarch project final technical report, 2004.

[32] David Clark, Robert Braden, Aaron Falk, and Venkata Pingali. FARA: reorganizing the addressing architecture. In *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pages 313–321, Karlsruhe, Germany, 2003.

[33] David D. Clark. Protocol design and performance. INFOCOM'95 tutorial notes, 1995.

[34] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in cyberspace: defining tomorrow's Internet. In *Proceedings of ACM SIGCOMM*, pages 347–356, Pittsburgh, Pennsylvania, USA, 2002.

[35] Bobby Bhattacharjee Dave Levin, Rob Sherwood. Fair File Swarming with FOX. In *Proceedings of IPTPS*, 2006.

[36] S. Deering. Simple internet protocol, September 1992. Work in progress.

[37] Zoran Despotovic and Karl Aberer. P2P reputation management: Probabilistic estimation vs. social networks. *Computer Networks*, 50(4):485–500, 2006.

[38] C. Dhillon, M. Bond, J. Griffioen, and K. Calvert. Building Layered Active Services. *Computer Networks*, 2006. (to appear).

[39] K. L. Calvert (editor). Architectural Framework for Active Networks. DARPA Active Networks Working Group Draft, December 2001.

[40] S. Floyd and V. Jacobson. Link-sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.

[41] Paul Francis. A near-term architecture for deploying pip. *IEEE Network*, 7(3):30–37, May 1993.

[42] Paul Francis and Ramesh Govindan. Flexible routing and addressing for a next generation IP. In *ACM SIGCOMM '94*, pages 116–125, 1994.

[43] Paul Francis and Ramakrishna Gummadi. IPNL: A NAT-extended Internet architecture. In *Proceedings of ACM SIGCOMM*, pages 69–80, August 2002.

[44] Lixin Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, December 2001.

[45] Geni: Global environment for network innovations. `http://www.geni.net/`.

[46] C. Gentry and A. Silverberg. Hierarchical ID-based Cryptography. In *Asiacrypt*, 2002.

[47] Vijay Gopalakrishnan, Bobby Bhattacharjee, and Peter Keleher. Distributing google. In *2nd IEEE International Workshop on Networking Meets Databases (NetDB' 06)*, Atlanta, GA, April 2006.

[48] Vijay Gopalakrishnan, Ruggero Morselli, Bobby Bhattacharjee, Peter Keleher, and Aravind Srinivasan. Ranking search results in peer-to-peer systems. Technical Report CS-TR-4779, University of Maryland, College Park, MD, January 2006.

[49] Vijay Gopalakrishnan, Bujor Silaghi, Bobby Bhattacharjee, and Pete Keleher. Adaptive replication in peer-to-peer systems. In *The 24th International Conference on Distributed Computing Systems*, March 2004.

[50] Minaxi Gupta, Mostafa H. Ammar, and Mustaque Ahamad. Trade-offs between reliability and overheads in peer-to-peer reputation tracking. *Computer Networks*, 50(4):501–522, 2006.

[51] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the Symposium on Operating Systems Principles*, pages 146–159, October 2001.

[52] Gísli Hjálmtýsson and Samrat Bhattacharjee. Control on Demand. In *Proceedings of the First International Working Conference on Active Networks (IWAN)*, June 1999.

[53] Gísli Hjálmtýsson and Samrat Bhattacharjee. Control on Demand: An efficient approach to router programmability. *IEEE Journal on Selected Areas in Communications (JSAC): Special Issue on Service Enabling Platforms for Networked Multimedia Systems*, August 1999. Guest Editors–D. Hutchison, G. Pacifici, B. Plattner, R. Stadler and J. Sventek.

[54] Tomasz Imielinski and Samir Goel. DataSpace – querying and monitoring deeply networked collection in physical space. In *Proceedings of International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'99)*, August 1999.

[55] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the MobiCom 2000 Conference*, pages 56–67, 2000.

[56] Van Jacobson and Michael J. Karels. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, November 1988.

[57] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. In *Proceedings of ACM SIGCOMM*, pages 295–308, 2002.

[58] C. Jaynes, B. Seales, K. Calvert, Z. Fei, and J. Griffioen. The Metaverse - A Collection of Inexpensive, Self-configuring, Immersive Environments. In *Proceedings of The 7th International Workshop on Immersive Projection Technology/Eurographics Workshop on Virtual Environments*, May 2003.

[59] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the Proc. 12th Intl. Conf. on the World Wide Web*, 2003.

[60] Vikas Kawadia and P.R. Kumar. A cautionary perspective on cross-layer design. *IEEE Wireless Communications*, 12(1), 2005.

[61] Pete Keleher, Samrat Bhattacharjee, and Bujor Silaghi. Are virtualized overlay networks too much of a good thing? In *The 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

[62] A. Keromytis, V. Misra, and D. Rubenstein. Sos: An architecture for mitigating DDoS attacks. *IEEE Journal on Selected Areas of Communications (JSAC)*, 2003.

[63] C. Kommareddy, S. Banerjee, and B. Bhattacharjee. Finding close friends on the Internet. *Submitted to Transactions on Networking*, 2002.

[64] Rajesh Krishnan, James P.G. Sterbenz, Wesley M. Eddy, Craig Partridge, and Mark Allman. Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks. *Computer Networks*, 46(3), 2004.

[65] T. LaPorta and ed. M.Gerla. Nsf wirless networking workshop: Final report. KU technical report ITTC-FY2004-TR-32950-01.

[66] S. Lee, S. Banerjee, and B. Bhattacharjee. The Case for Multi-hop Wireless Local Area Network. In *Proceedings of IEEE Infocom*, March 2004.

[67] S. Lee, B. Bhattacharjee, A. Srinivasan, and S Khuller. Efficient and Resilient Backbones for Multihop Wireless Networks. Technical report, CS-TR 4726, University of Maryland, College Park. Available at http://www.cs.umd.edu/users/slee/pubs/tr4726.pdf, June 2005.

[68] Seungjoon Lee, Bobby Bhattacharjee, and Suman Banerjee. Efficient Geographic Routing in Multihop Wireless Networks. In *Proceedings of ACM MobiHoc*, May 2005.

[69] Seungjoon Lee, Rob Sherwood, and Bobby Bhattacharjee. Cooperative peer groups in NICE. *Proceedings of INFOCOM*, 2003.

[70] Ratul Mahajan, Neil Spring, David Wetherall, and Thomas Anderson. Inferring link weights using end-to-end measurements. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, pages 231–236, 2002.

[71] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[72] Ruggero Morselli, Bobby Bhattacharjee, Jonathan Katz, and Pete Keleher. Trust-preserving set operations. *Proceedings of INFOCOM*, 2004.

[73] Ruggero Morselli, Bobby Bhattacharjee, Jonathan Katz, and Michael Marsh. Keychains: A Decentralized Public-Key Infrastructure. Technical Report CS-TR-4788, Department of Computer Science, University of Maryland, March 2006.

[74] Ruggero Morselli, Bobby Bhattacharjee, Aravind Srinivasan, and Michael A. Marsh. Efficient lookup on unstructured topologies. In *PODC*, pages 77–86, 2005.

[75] T. S. Eugene Ng, Yang-hua Chu, Sanjay G. Rao, Kunwadee Sripanidkulchai, and Hui Zhang. Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems. In *Proceedings of the IEEE Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 2199–2209, 2003.

[76] Kihong Park and Heejo Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proceedings of ACM SIGCOMM*, pages 15–26, 2001.

[77] Adrian Perrig, J. D. Tygar, Dawn Song, and Ran Canetti. Efficient authentication and signing of multicast streams over lossy channels. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 56, Washington, DC, USA, 2000. IEEE Computer Society.

[78] Larry Peterson, Thomas Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the Internet. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, pages 59–64, 2002.

[79] L. Poutievski, K. Calvert, and J. Griffioen. Routing without Addresses. In *The Poster Proceedings of ICNP 2004*, October 2004.

[80] L. Poutievski, K. Calvert, and J. Griffioen. Speccast. In *Proceedings of the INFOCOM 2004 Conference*, March 2004.

[81] Yakov Rekhter and Tony Li. A border gateway protocol 4 (BGP-4). Internet Engineering Task Force Request for Comments RFC-1771, March 1995.

[82] Sean Rhea, Byung-Gon Chun, John Kubiatowicz, and Scott Shenker. Fixing the embarrassing slowness of OpenDHT on PlanetLab. In *Proceedings of the Second USENIX Workshop on Real, Large Distributed Systems*, pages 25–30, December 2005.

[83] Eric C. Rosen, Arun Viswanathan, and Ross Callon. Multiprotocol label switching architecture. Internet Engineering Task Force Request for Comments RFC-3031, January 2001.

[84] Narendar Shankar, Christopher Komareddy, and Bobby Bhattacharjee. Finding Close Friends over the Internet. In *Proceedings of International Conference on Network Protocols*, November 2001.

[85] Rob Sherwood, Bobby Bhattacharjee, and Ryan Braud. Misbehaving TCP Receivers Can Cause Internet-Wide Congestion Collapse. In *Proceedings of Computer and Communications Security (CCS)*, 2005.

[86] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: A Protocol for Scalable Anonymous Communication. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[87] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: A Protocol for Scalable Anonymous Communication. *Journal of Computer Security (To appear)*, 2006.

[88] Rob Sherwood, Ryan Braud, and Bobby Bhattacharjee. Slurpie: A cooperative bulk data transfer protocol. In *Proceedings of IEEE INFOCOM*, 2004. To appear.

[89] Rob Sherwood, Seungjoon Lee, and Bobby Bhattacharjee. Cooperative peer groups in nice. *Computer Networks*, 50(4):523–544, 2006.

[90] S. Shi, L. Wang, K. Calvert, and J. Griffioen. A Multi-path Routing Service for Immersive Environments. In *Proceedings of the Workshop on Grids and Advanced Networks 2004 (GAN'04) in conjunction with 4th IEEE/ACM International Symposium on Cluster Computing and the Grid*, April 2004.

[91] Bujor Silaghi, Bobby Bhattacharjee, and Pete Keleher. Routing in the TerraDir Directory Service. In *Proceedings of SPIE ITCOM*, 2002.

[92] Bujor Silaghi, Vijay Gopalakrishnan, Bobby Bhattacharjee, and Pete Keleher. Hierarchical routing with soft-state replicas in TerraDir. In *The 18th International Parallel and Distributed Processing Symposium*, April 2004.

[93] Bujor Silaghi, Pete Keleher, and Bobby Bhattacharjee. Multi-dimensional quorum sets for read-few write-many replica control protocols. In *Fourth International Workshop on Global and Peer-to-Peer Computing*, April 2004.

[94] Neil Spring, Ratul Mahajan, and Thomas Anderson. Quantifying the causes of path inflation. In *Proceedings of ACM SIGCOMM*, pages 113–124, 2003.

[95] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM*, pages 133–146, 2002.

[96] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using PlanetLab for network research: Myths, realities, and best practices. *ACM SIGOPS Operating Systems Review*, 40(1):17–24, January 2006.

[97] Neil Spring, David Wetherall, and Thomas Anderson. Reverse-engineering the Internet. In *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, pages 3–8, Cambridge, MA, November 2003.

[98] James P.G. Sterbenz. Towards a framework for cross-layer optimisation in support of survivable and resilient autonomic networking. Dagstuhl Perspectives Workshop on Autonomic Networking,, 2006. available from http://www.ittc.ku.edu/survivability/papers/dagstuhl2006.pdf.

[99] James P.G. Sterbenz and Joseph D. Touch. *High Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication*. John Wiley Networking Council, 2001.

[100] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. In *ACM SIGCOMM*, pages 73–88, August 2002.

[101] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *ACM SIGCOMM*, pages 149–160, San Diego, California, August 2001.

[102] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 39–44, 2003.

[103] Feng Wang and Lixin Gao. Inferring and characterizing Internet routing policies. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 15–26, 2003.

[104] L. Wang, J. Griffioen, and K. Calvert. Estimating Achievable Throughput. In *The Poster Proceedings of SIGCOMM 2005*, August 2005.

[105] L Wang, J. Griffioen, K. Calvert, and S. Shi. Passive Inference of Path Correlation. In *Proceedings of the NOSSDAV '04 Conference*, June 2004.

[106] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guguprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and network. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, December 2002.

[107] Abraham Yaar, Adrian Perrig, and Dawn Xiaodong Song. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In *IEEE Symposium on Security and Privacy*, 2004.

[108] Xiaowei Yang. NIRA: a new Internet routing architecture. In *ACM Workshop on Future Directions In Network Architecture*, pages 301–312, Karlsruhe, Germany, 2003. ACM Press.

[109] Xiaowei Yang, David Wetherall, and Thomas Anderson. A DoS-limiting network architecture. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 241–252, Philadelphia, Pennsylvania, USA, 2005. ACM Press.

[110] P. Zimmermann. Pretty good privacy user's guide. Distributed with PGP software, June 1993.