Unmanaged Internet Protocol
Taming the Edge Network Management Crisis

Bryan Ford
Massachusetts Institute of Technology

Abstract

Though appropriate for core Internet infrastructure, the Internet Protocol is unsuited to routing within and between emerging ad-hoc edge networks due to its dependence on hierarchical, administratively assigned addresses. Existing ad-hoc routing protocols address the management problem but do not scale to Internet-wide networks. The promise of ubiquitous network computing cannot be fulfilled until we develop an *Unmanaged Internet Protocol* (UIP), a scalable routing protocol that manages itself automatically. UIP must route within and between constantly changing edge networks potentially containing millions or billions of nodes, and must still function within edge networks disconnected from the main Internet, all without imposing the administrative burden of hierarchical address assignment. Such a protocol appears challenging but feasible. We propose an architecture based on self-certifying, cryptographic node identities and a routing algorithm adapted from distributed hash tables.

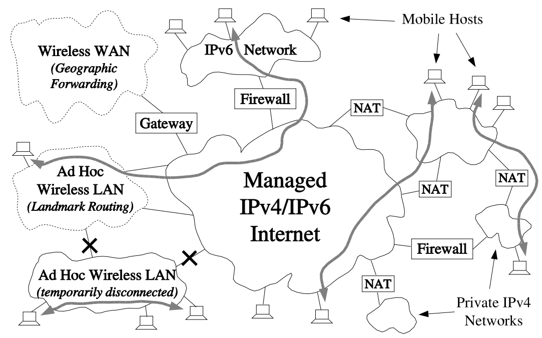

Figure 1: Today's Internetworking Challenges

1 Introduction

The promise of ubiquitous computing is that people will soon routinely own many "smart" networked devices, some mobile, others perhaps built into their homes and offices, and all of which they can access and control from any location so long as appropriate security precautions are taken. Before we can expect ordinary, non-technical people to adopt this vision, however, the ad-hoc *edge networks* in which these devices live must be able to manage themselves. Each device must be able to find and communicate with its peers—whether connected directly, indirectly over a local-area network, or remotely across a long distance via the Internet—with no special configuration or other technical effort on the part of the user.

The current Internet Protocol is unsuited to this purpose. IPv4 and IPv6, with their accompanying routing, naming, and management protocols, have evolved around the requirements of *core* network infrastructure: corporate, academic, and government networks deployed and managed by skilled network administrators. IP's hierarchical address architecture in particular is fundamentally dependent on skilled network management. Current ad-hoc networking protocols by themselves are not sufficient either, because they are only scalable to local-area network sizes of a few hundreds or thousands of nodes.

To achieve ubiquitous network computing, we need an *Unmanaged Internet Protocol*, or UIP, that combines the self-management of ad-hoc networks with the scalability of IP. As illustrated in Figure 1, achieving this goal in today's chaotic mix of networking technologies also means routing traffic automatically and securely through NATs, and transparently bridging IPv4, IPv6, and other address domains. We propose an architecture based on *scalable identity-based routing*, or routing based on topology-independent node identifiers. While more difficult than routing over topology-dependent addresses such as IP addresses, there is evidence that scalable identity-based routing is possible and practical.

This position paper is organized as follows. Section 2 lays out the motivation for UIP and the inadequacies of current solutions. Section 3 proposes and outlines an identity-based UIP routing architecture, and Section 4 describes implementation status and deployment. Section 5 summarizes related work, and Section 6 concludes.

2 Motivation for UIP

The original ARPAnet vision was to enable computer users to communicate and share resources with users of any other connected computer [18, 28]. This vision has evolved into the modern Internet Protocol, whose purpose is to implement any-to-any connectivity between hosts, whether connected directly or indirectly via paths crossing many administrative domains. While physical and link-layer technologies such as Ethernet provide low-level building blocks for communication, and higher-level protocols enable applications and users to take advantage of the network, *interoperable end-to-end connectivity* via IP remains the Internet's central focus.

Technical, social, and economic pressures have hindered the achievement of this vision, however. The protocols underlying the Internet were designed by technically savvy individuals who understand how networks work but often do not understand how *non-technical users* work. As a result many aspects of network operation still require careful and skilled management. We desire and increasingly expect that *everyone* should be able to use the Internet and deploy networked devices, and strong economic incentives exist for businesses to sell Internet-enabled hardware and Internet-based services to technically unsophisticated users. Since businesses seek lowest-cost paths to profitable solutions, the commercial Internet has evolved—via a chaotic series of hacks and extensions—into a system geared toward particular usage patterns that facilitate business opportunities, often at the expense of interoperability and general end-to-end connectivity.

2.1 The Edge Network Management Crisis

An unsophisticated user can now buy a computer, connect it to the Internet, and use it for browsing the Web, reading E-mail, and shopping on-line. Users who are a bit more adventurous but still relatively non-technical may set up a small home network and surf the Web from several computers at once. But consider the following scenario:

1. Joe User is working at home on his laptop. He has remote shell and database access sessions open, through his WiFi home network, to his desktop PC and to a machine at his workplace.

2. Joe's friend Jim calls and invites him over. Joe puts his laptop into sleep mode and hops into his car.

3. Joe stops for a bite to eat on the way to Jim's, and scribbles some notes on his PDA in the restaurant.

4. Upon returning to his car, Joe tries to synchronize his PDA with his laptop, but discovers they won't talk to each other even though they're both WiFi-enabled and are at most a foot apart. Being unfamiliar with the technical details of IP networks, he doesn't realize that this is because (a) the WiFi adapters are in infrastructure rather than ad-hoc mode, and (b) even if they could communicate at the link layer, neither machine would be able to get an IP address because there's no DHCP [7] server nearby.

5. Joe arrives at Jim's place, and the two brainstorm about their project at work. Joe takes out his laptop and wakes it up. Since Jim also has an Internet-connected WiFi home network, Joe hopes to use Jim's Internet access and resume his existing application sessions to his desktop PCs at home and work. Again Joe is disappointed. After figuring out that he has to remove and re-insert his laptop's WiFi card in order to get it to recognize Jim's network at all, Joe's application sessions are gone. He does not realize that moving to a new attachment point changed his IP address, breaking his existing TCP connections.

6. Joe tries to re-start his application sessions, but finds that he cannot even locate let alone connect with his desktop PC at home. He doesn't realize that this is because (a) his ISP did not give him a permanent IP address useable for connecting remotely to his home network, and (b) even with a permanent IP address, his desktop PC would still be inaccessible because it is behind a network address translator [32].

Joe's naïve expectations of his networked devices are not fundamentally unreasonable, and all of the problems above are solvable with current technology. Joe could in theory: (a) configure his home NAT to assign fixed site-local IP addresses to his desktop and laptop PCs at home; (b) configure his NAT to open the appropriate external ports for remote access and forward incoming connections on those ports to his desktop PC; (c) register for a global DNS host name with a Dynamic DNS [36] service provider; (d) set up his desktop PC to update this DNS name periodically with the dynamic IP address his ISP assigns to his home NAT; (e) set up Mobile IP [25] so that his desktop PC at home will intercept packets destined for his laptop's "home" IP address, and tunnel them to his laptop at its actual connection point while connected elsewhere; (f) run daemons on his laptop and PDA that detect when no infrastructure-mode WiFi access point or DHCP service is available, and automatically switch into ad-hoc mode using a routing protocol such as AODV [24].

Only the most dedicated, desperate, or geeky will go to this trouble, however. To most users, having a "working" network means being able to get to Google, CNN, and Amazon.com. Any "ubiquitous" connectivity outside this commercial client/server straitjacket is fickle, unreliable, and management-intensive if available at all.

2.2 IP Networks Require Management

The scalability and efficiency of the current Internet Protocol relies on Classless Inter-Domain Routing (CIDR) [26], in which network nodes are assigned addresses whose hierarchical structure reflects the routing topology. BGP routers take advantage of the hierarchical structure of IP addresses, aggregating information about distant nodes and networks sharing a common address prefix into a single routing table entry [27].

While this hierarchical address assignment scheme makes the core Internet infrastructure efficient and scalable, it is precisely this address assignment scheme that makes edge networks brittle and difficult to manage. Whenever a node moves or its surrounding network is renumbered, the node's IP address must change. Statically configuring and maintaining the IP addresses of many nodes is challenging even for technically competent network administrators, leading to organizational resistance against IP address renumbering [5]. Dynamic address assignment transfers administrative responsibility from edge nodes to DHCP servers, at the expense of making edge nodes unable to communicate at all without access to a DHCP server. Workarounds in which nodes choose their own local IP addresses after failing to contact a DHCP server [6] are slow, unreliable, and at best allow nodes to communicate only with immediate link-neighbors while disconnected from the main Internet. These issues will persist even into a future IPv6 world in which there are "enough" IP addresses for everyone and network address translators do not exist, because the basic address architecture remains the same in IPv6.

2.3 Ad-Hoc Networks Do Not Scale

Classic distance-vector [15, 11] and link-state [22] routing protocols, as well as ad-hoc routing variants such as DSR [14] and AODV [24], require every node to store and regularly exchange information about *every* other node in the network. This linear per-node storage and/or bandwidth overhead limits the scalability of these protocols to a few hundreds or thousands of nodes. While ad-hoc protocols can be used to route within a particular IP subnet, this subnet must be centrally allocated and managed in order to be globally routable on the Internet, and all participating nodes must be assigned to that subnet. Configuring each node statically is tedious and inconvenient, while using DHCP again makes the nodes unable to communicate with each other while out of range of a DHCP server.

To fulfill the promise of ubiquitous networking, an edge network routing protocol must be self-managing not only on a *local* scale, but also on a *global* scale. We need an ad-hoc routing protocol that can seamlessly route packets throughout an *Internet-wide federation* of ad-hoc edge networks, consisting of potentially millions or billions of edge nodes that frequently hop from one edge network to another. This protocol must still provide reliable ad-hoc routing within edge networks that are temporarily or permanently disconnected from the Internet. This is the purpose of *Unmanaged Internet Protocol*, or UIP.

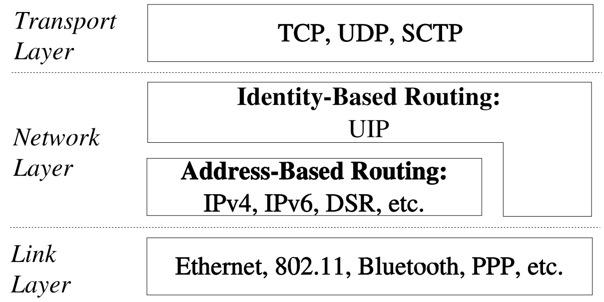

Figure 2: UIP in the Internet Protocol Architecture

3 Proposed UIP Architecture

Since IP does an excellent job of routing packets efficiently through the managed core Internet infrastructure, we intend UIP not to replace IP but to run on top of it, as a new network layer component (Figure 2). In our proposed architecture, UIP appears to upper-level transport and application protocols as a new address/protocol family, much like IPv6 does now.

3.1 Node Identities

To refer to other UIP nodes, applications use self-certifying cryptographic *identifiers* that are stable over time and independent of network topology. All connections between UIP nodes are privacy- and integrity-protected by default, as in IPSEC [16]. A node's UIP identifier is a hash of the node's public key, making identifiers *self-certifying*, like Moskowitz's host identities [21] or SFS pathnames [20]. Cryptographic identities provide several properties crucial to achieving robust connectivity in future ubiquitous networking environments:

- Any node can create a globally unique UIP identifier at any time without reference to central authorities. The identifier's uniqueness depends only on the strength of the cryptographic hash used to create it.
- A node's identifier remains valid as long as desired. Security practice may limit the lifetimes of long-term keys and node identities to a few years, but since this is the useful lifetime of most PCs, many nodes may never have to change identifiers.

- Since a node identifier contains no topology information, the node can retain its identity when it moves or the surrounding network topology changes.

- A node can cryptographically prove ownership of an identifier using the associated private key, preventing an attacker from stealing its identity.

- A node can have multiple identities simultaneously, representing distinct services or "virtual hosts" on one physical machine for example.

- The network layer does not depend on centralized public key infrastructure (PKI). Higher layers may use PKI to map convenient names to node identifiers, but given a node identifier, finding and connecting securely to that node is fully decentralized.

3.2 UIP Routing

UIP's primary technical challenge is to forward traffic from any node to any other in an Internet-scale network, without the help of hierarchically structured node addresses. Since UIP node identifiers are unrelated to network topology, they have no locality properties routers can use to aggregate routing information about distant nodes. Requiring every node to store and propagate routing information about every other node in an Internet-scale network may arguably be viable for a desktop PC with a high-speed Internet connection, but is definitely impractical for small, low-power devices such as PDAs.

3.2.1 Approaches to Identity-Based Routing

Bellman-Ford and similar routing algorithms find *optimal* routes, based on either hop count or some per-link cost metric. We do not need optimal routing, however: in practice it suffices to find *reasonably efficient* routes. The routes that BGP finds are probably less than optimal already, due to the difficulty of supporting site multihoming in the Internet's hierarchical address model [1], and the lack of incentive for ISPs to reveal all of their peering relationships in their BGP advertisements.

Route efficiency is usually measured in terms of *stretch*: the length of the route discovered by the protocol over the length of the best possible route. Algorithms now exist that can route through an N-node network with arbitrary node labels, using $\tilde{O}(\sqrt{N})$ bits of routing table state per node, and a small constant maximum stretch [4].

In practice, we do not even necessarily demand that *all* nodes have sublinear storage requirements. We might accept a routing protocol that has sublinear overhead on *most* nodes, but requires *a few* nodes present in the network to have $\Omega(N)$ storage and/or bandwidth. It is critical to our ubiquitous networking goals, however, that these "supernodes" do not need to be hard-wired as such. *Any* node must be able to take on that role dynamically whenever supernodes are needed and the surrounding network is small enough. While Joe's laptop and PDA are connected to the Internet, for example, their ability to route to distant edge nodes might depend on a massive central server somewhere that continuously maintains a complete map of the Internet. If Joe takes several network devices with him into the mountains where there is no Internet access, however, each device must be able to take on supernode responsibilities as necessary to direct traffic within any smaller ad-hoc network he may form. Joe's laptop and other small devices never need to map the entire Internet, but only the smaller edge networks Joe may participate in while disconnected from the Internet.

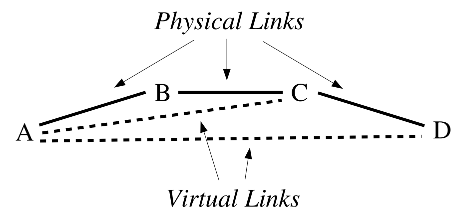

Figure 3: Forwarding via Virtual Links

3.2.2 Converting DHTs into Routing Algorithms

We are experimenting with a scalable routing protocol for UIP derived from the Kademlia distributed hash table (DHT) [19]. This protocol, detailed elsewhere [9], empirically achieves $O(log\ N)$ storage and maintenance overhead per node and an average stretch of 2 on simulated networks. We have not yet formalized the algorithm or derived theoretical performance properties, however.

DHTs normally implement only indexing and lookup, relying on underlying protocols to provide connectivity between all participants. Our protocol extends Kademlia to function over any topology through the use of recursive source routes, or *virtual links*. Once any two nodes have connected and established a "neighbor" relationship, ether node can use that relationship to build further neighbor relationships recursively, covering longer topological distances. In Figure 3, for example, node A uses physical links AB and BC to build a virtual link AC, thereby establishing a neighbor relationship with C. Node A then uses virtual link AC to build a recursive virtual link to C's physical neighbor D. Node D can now communicate with node A via its physical link to C and C's virtual link to A, without having to know the details of the path between C and A. Node D might not know about the existence of node B at all. In this way the routing protocol abstracts the details of routing at different levels, achieving an effect analogous to IP address aggregation without actually depending on hierarchically assigned addresses.

A node n sorts its neighbors n_i into *buckets*, according to the longest common prefix length in bits between n's and n_i's identifier. To join the network, n needs a physical link to some existing node n_1. Node n searches n_1's neighbor table for another node n_2 with a longer common prefix, and builds a virtual link through n_1 to n_2. This process continues until n finds the node with the closest identifier to its own. If a suitable connectivity invariant is maintained, every node in the resulting structure can find and build a forwarding path to any other node on demand.

4 Implementation and Deployment

A UIP prototype is under development, which we look forward to using and evaluating shortly. For portability, the prototype runs as an application-level daemon, communicating with other nodes primarily over UDP. The daemon can also directly utilize the link layers of some systems if it has appropriate privileges. Multiple local applications can share a single UIP daemon, interacting through a proxy library that exports a standard sockets-based interface. To simplify initial deployment of UIP, applications without special privileges can be bundled with their own UIP daemon, which the application starts automatically if a systemwide daemon is not available.

An immediate benefit of UIP is that it allows applications to establish secure, peer-to-peer connections through NAT and firewall barriers without special effort, as long as there are some widely-accessible UIP nodes on the Internet through which the daemon can forward traffic if necessary. The UIP daemon implements UDP hole punching [8] as an application-transparent optimization, utilizing widely-accessible UIP nodes as "introducers" to establish direct IP-based peer-to-peer communication paths across many NATs and firewalls. The daemon falls back to explicit forwarding whenever hole punching fails, ensuring maximum robustness.

5 Related Work

UIP is a fusion of ideas from many projects. Like Resilient Overlay Networks [3], UIP introduces a routing layer above IP that can route around discontinuities and failures in the Internet, but UIP seeks to be scalable and self-managing as well as resilient. Ad-hoc routing protocols such as DSR [14] address the management problem at the local level but are not scalable to Internet-wide ad-hoc networks. Landmark [34] and AODV [24] offer scalability under localized traffic patterns, but not under the global traffic patterns of the Internet.

UIP node identities are similar to those of Moskowitz's proposed Host Identity Protocol [21], but UIP uses identities for routing as well as authentication. The Internet Indirection Infrastructure ($i3$) [33] provides location-independent host identities, multicast and anycast communication, and NAT traversal, but does not implement a general-purpose routing protocol that can function independently from the Internet.

Many Internet host mobility solutions have been proposed. Higher-level naming systems can provide applications independence from their host's IP address [36, 2, 30], at the cost of tying applications to a particular naming scheme and making it difficult to maintain connections as the host moves [31]. Mobile IP [25] allows a mobile host to roam without breaking outstanding TCP connections or UDP bindings, but requires each mobile host to have a stable "home" IP address through which packets are tunneled. A similar illusion of a static IP address can be achieved with IP multicast [23, 12].

Work on peer-to-peer connectivity through firewalls and NATs [13] has led to various special-purpose protocols [17, 29, 35]. UDP hole punching [8] allows peer-to-peer connectivity through many NATs, but not all, without the use of explicit proxy protocols. Name-based routing [10] offers more general bridging of IP address domains, but its ties to the management-heavy domain name system make it unsuitable for ad-hoc networks.

6 Conclusion

Ubiquitous network computing will require an ad-hoc routing protocol that can not only route autonomously within small edge networks of hundreds or thousands of nodes, but can seamlessly route among a large Internet-connected *federation* of edge networks. The traditional solution to scalability, hierarchical address assignment, is unsuited to edge networks due to its management costs. Scalable identity-based routing protocols appear feasible and represent a promising research direction, though many practical technical problems remain unsolved. Besides providing a key building block for ubiquitous network computing, scalable identity-based routing may also help address the more immediate problems of Internet host mobility, NAT traversal, and bridging between IPv4, IPv6, and other address domains.

Acknowledgments

I wish to thank my advisor Frans Kaashoek, my colleagues Dave Andersen and Chris Lesniewski-Laas, Prof. David Karger, and the HotNets reviewers for many helpful comments and suggestions.

References

[1] J. Abley, B. Black, and V. Gill. Goals for IPv6 site-multihoming architectures, August 2003. RFC 3582.

[2] William Adjie-Winoto et al. The design and implementation of an intentional naming system. In *17th ACM Symposium on Operating System Principles*, Kiawah Island, SC, December 1999.

[3] David G. Andersen et al. Resilient overlay networks. In *18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[4] Marta Arias et al. Compact routing with name independence. In *15th ACM Symposium on Parallelism in Algorithms and Architectures*, San Diego, CA, June 2003.

[5] B. Carpenter, J. Crowcroft, and Y. Rekhter. IPv4 address behaviour today, February 1997. RFC 2101.

[6] Microsoft Corporation. Plug and play networking with Microsoft automatic private IP addressing, March 1998.

[7] R. Droms. Dynamic host configuration protocol, March 1997. RFC 2131.

[8] Bryan Ford. Peer-to-peer (P2P) communication across middleboxes, October 2003. Internet Draft `draft-ford-midcom-p2p-01.txt` (Work in Progress).

[9] Bryan Ford. Scalable Internet routing on topology-independent node identities. Technical Report MIT-LCS-TR-926, MIT Laboratory for Computer Science, 2003. Forthcoming.

[10] M. Gritter and D. R. Cheriton. An architecture for content routing support in the Internet. In *Usenix Symposium on Internet Technologies and Systems*, March 2001.

[11] C. Hedrick. Routing information protocol, June 1988. RFC 1058.

[12] Ahmed Helmy. A multicast-based protocol for IP mobility support. In *Networked Group Communication*, pages 49–58, 2000.

[13] M. Holdrege and P. Srisuresh. Protocol complications with the IP network address translator, January 2001. RFC 3027.

[14] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[15] L. R. Ford Jr. and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton N.J., 1962.

[16] S. Kent and R. Atkinson. Security architecture for the Internet Protocol, November 1998. RFC 2401.

[17] M. Leech et al. SOCKS protocol version 5, March 1996. RFC 1928.

[18] J.C.R. Licklider. The computer as a communication device. *Science and Technology*, April 1968.

[19] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *1st International Workshop on Peer-to-Peer Systems*, March 2002.

[20] David Mazières, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. Separating key management from file system security. In *17th ACM Symposium on Operating Systems Principles*, Kiawah Island, South Carolina, December 1999.

[21] R. Moskowitz and P. Nikander. Host identity protocol architecture, April 2003. Internet-Draft (Work in Progress).

[22] J. Moy. OSPF version 2, July 1991. RFC 1247.

[23] Jayanth Mysore and Vaduvur Bharghavan. A new multicasting-based architecture for Internet host mobility. In *Third ACM/IEEE International Conference on Mobile Computing and Networking*, pages 161–172, 1997.

[24] Charles E. Perkins and Elizabeth M. Belding-Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

[25] C. Perkins, Editor. IP mobility support for IPv4, August 2002. RFC 3344.

[26] Y. Rekhter and T. Li (editors). An architecture for IP address allocation with CIDR, September 1993. RFC 1518.

[27] Y. Rekhter and T. Li (editors). A border gateway protocol 4 (BGP-4), March 1995. RFC 1771.

[28] Lawrence G. Roberts and Barry D. Wessler. Computer network development to achieve resource sharing. In *Spring Joint Computer Conference*, Atlantic City, New Jersey, May 1970.

[29] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs), March 2003. RFC 3489.

[30] Alex C. Snoeren and Hari Balakrishnan. An end-to-end approach to host mobility. In *Sixth ACM/IEEE International Conference on Mobile Computing and Networking*, August 2000.

[31] Alex C. Snoeren, Hari Balakrishnan, and M. Frans Kaashoek. Reconsidering Internet mobility. In *8th Workshop on Hot Topics in Operating Systems*, May 2001.

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

[33] Ion Stoica et al. Internet indirection infrastructure. In *ACM SIGCOMM*, 2002.

[34] Paul Francis Tsuchiya. The Landmark hierarchy: A new hierarchy for routing in very large networks. In *ACM SIGCOMM*, pages 35–42, Stanford, CA, August 1988.

[35] UPnP Forum. Internet gateway device (IGD) standardized device control protocol V 1.0, November 2001. `http://www.upnp.org/`.

[36] P. Vixie, Editor, S. Thomson, Y. Rekhter, and J. Bound. Dynamic updates in the domain name system (DNS UPDATE), April 1997. RFC 2136.