

# Transparent Mobility with Minimal Infrastructure

Praveen Yalagandula, Amit Garg, Mike Dahlin, Lorenzo Alvisi, Harrick Vin  
ypraveen, amitji, dahlin, lorenzo, vin@cs.utexas.edu  
University of Texas at Austin  
**IBM Contact:** Ray Young, Linux Technology Group

## 1 Synopsis

The COPE (Consistent 0-administration Personal Environment) project at the University of Texas at Austin seeks to address the problems that arise when individual users have dozens of machines sharing their data and services. This paper addresses one key problem in such environment: each machine's IP addresses may often change due to mobility, due to switching between different network connections (e.g., Ethernet, 802.11, and infrared), and due to dynamic IP assignment (e.g., DHCP). Because higher level protocols and applications often implicitly assume that IP addresses correspond to specific, unchanging hosts, changes in the physical IP address used to reach a host can cause failures in these protocols and applications.

To address this problem, we introduce VIP, a virtual IP layer, that applies the principle of virtual addressing to Internet naming. VIP presents an abstraction in which machines refer to one another by name and in which physical IP addresses are hidden from higher-level protocols.

Two key goals of VIP's design are incremental deployability and minimal configuration.

VIP achieves these goals by following two design principles (1) **transparent mobility**: the system virtualizes the IP level of the protocol stack—the “neck of the protocol hourglass”—to avoid modifying higher-level network protocols and applications, and (2) **minimal infrastructure**: the system takes advantage of and minimizes changes to existing network infrastructure.

MobileIP [10, 12] also attempts to support transparent mobility, but despite its early development and standardization, MobileIP is not widely used. We believe this is due to its infrastructure requirements: to use MobileIP a user must acquire *static, globally unique IP addresses* for each device and a user must deploy a *home*

*agent* machine that forwards packets routed to these addresses to the user's mobile devices.

Although these design trade-offs made sense in the early 1990's, recent trends in network deployment increase the significance of these requirements as barriers to deployments. First, the 32-bit IPv4 address space makes static IP addresses expensive. For example, an ISP account that includes multiple static IP addresses may be much more expensive than a dynamic-IP account, making it costly for an individual user to deploy MobileIP for personal use. In addition, the widespread use of firewalls to protect intranets and even home networks makes it hard to deploy a globally-reachable MobileIP home agent; often a dedicated machine must be deployed to the public side of the firewall, where machine deployment may be tightly controlled.

VIP is a simple system that addresses these problems. VIP uniquely identifies machines by their fully-qualified domain names (FQDNs) (e.g., *example.acm.org*) rather than their IP addresses (e.g., *199.222.69.43*), and uses Secure Dynamic DNS (DDNS) [5] to update and distribute name-to-address mappings as machines move. To maintain backward compatibility, each pair of communicating nodes negotiates virtual IP addresses, which are opaque 32-bit tokens that correspond to their FQDNs. Applications and other layers above VIP use these 32-bit virtual IP addresses rather than physical IP addresses. When machines communicate, the VIP layer translates between virtual and physical IP addresses.

This approach supports transparent mobility without requiring deployment of new infrastructure. First, applications and other network layers above the VIP layer never see physical IP addresses, so the VIP layer is free to change the mapping from the the virtual IP address token used to identify a machine to the physical address used to route packets. Second, routers and other infrastructure below the VIP layer never see virtual IP

addresses, so each pair of communicating nodes is free to negotiate lightweight VIP-to-FQDN mappings rather than relying on globally unique and unchanging static IP addresses. Third, by using DDNS for mapping FQDNs to IP addresses, the system takes advantage of existing translation infrastructure rather than requiring deployment of new translation infrastructure. Furthermore, free third-party DNS servers such as no-ip.com and dyn-dns.org allow individual users to use the system without even having to manage their own name servers.

We have implemented the VIP system in Linux. Our experiments show that the system provides efficient remapping both when one node moves and when both parties in a connection simultaneously change their addresses. A key optimization in the system is peer-to-peer hints, which can greatly improve the latency of this remapping. We use IPsec [8] to provide end-to-end security as VIP  $\leftrightarrow$  IP address mappings change; to keep infrastructure requirements low, VIP implements the option of a simple peer-to-peer anonymous key exchange protocol for IPsec that is similar to the one used by SSH [9]. Overall, we find that new infrastructure and practices that have become widespread since the standardization of MobileIP (e.g., DHCP for dynamic IP assignment [4], Dynamic Secure DNS for updating name-to-IP mappings [5], and IPsec for secure communication [8]) make it relatively simple to support transparent mobility without requiring other infrastructure changes.

The main limitation of this approach is our decision to modify end-stations rather than relying on external infrastructure. There are two issues. First, although our system is backwards compatible in that it allows communications with unmodified machines, our system does not support migration of connections unless both participating machines implement our extension. Second, although this approach simplifies many aspects of deployment, modifying end-stations does involve barriers of its own. However, as others have argued [14], although the MobileIP design assumed that it was easier to modify routers than end stations, in practice the reverse seems to be true. Furthermore, modifying end stations solves the “chicken and egg” problem faced by infrastructure-based approaches: in an end-station based approach, an individual can take advantage of the optimization by upgrading her machines; whereas in an infrastructure-based approach, there is little incentive to deploy new infrastructure until a large user base emerges and little incentive to become part of that user

base until infrastructure is deployed.

## 2 Related Work

The problem of supporting host mobility on the Internet has been extensively studied. MobileIP, like VIP, is designed to achieve transparent mobility by virtualizing the IP layer of the protocol stack. Other approaches have been proposed at the IP, transport, and application layers of the protocol stack or have used names rather than physical addresses for routing.

### 2.1 MobileIP

MobileIP [10, 12] has many similarities with VIP. Both systems share the goal of providing transparent mobility. In both systems, this is achieved by decoupling the routing and naming roles that coexist in conventional IP addresses; in both systems a host receives two distinct identifiers - a permanent name, that does not change when a host moves, and a variable address, that changes to reflect the host’s current point of attachment to the Internet.

There are two fundamental differences between MobileIP and VIP.

The first is the nature of a mobile host’s permanent name: in Mobile IP the home address is a valid IP address to which packets can be routed, but in VIP the virtual address is a node’s fully-qualified domain name (FQDN). For backwards compatibility with layers above VIP, VIP introduces 32-bit tokens that act as synonyms with FQDNs, but these VIP addresses have no semantic meaning within IP. This clean distinction between virtual and physical IP addresses makes deployment easier and cheaper for a user. Whereas VIP nodes automatically generate lightweight VIP address tokens, a user wishing to deploy a device using MobileIP must acquire a static IP address from an ISP.

The second fundamental difference is the mechanism used to map between home (or virtual) addresses and correspondent (or physical) addresses. MobileIP uses a home agent machine that receives packets addressed to a mobile node’s home address and that tunnels packets to the mobile node’s correspondent address. Instead, VIP uses DNS to map machine names to physical IP addresses. By using DNS for translation, VIP takes advantage of existing infrastructure rather than requiring

deployment of new translation infrastructure. Furthermore, free third-party DNS servers such as no-ip.com and dyndns.org allow individual users to use VIP without even having to manage their own name servers.

Note that although route optimizations to some implementations of MobileIP modify both communicating end-stations to eliminate need for separate foreign agents [10] and to reduce triangle routing [11], these implementations still require home agents to establish and update the mappings cached at end stations. But also note that MobileIP’s approach of using routable addresses and home agents provides one advantage over VIP: if a mobile node that has been modified to support MobileIP communicates with a fixed node that does not support MobileIP, MobileIP’s home agent can forward packets from the fixed node as the mobile node moves. Conversely, although VIP retains backwards compatibility in that VIP and non-VIP nodes can communicate, VIP does not support mobility in such a scenario. Unfortunately, it appears that this advantage of MobileIP fundamentally requires that virtual addresses be routable, which we believe raises too high a barrier to deployment.

IPv6 [3] supports mobility using MobileIP techniques. However, it requires deployment of large amounts of new infrastructure.

## 2.2 Other approaches

Below, we discuss other approaches to mobility, organized by the layer of the protocol stack at which virtualization is introduced: the IP layer, transport layer, or application layer. Finally, we discuss several other “name-centric” routing architectures.

Gupta and Reddy [7] propose a IP-level redirection mechanism that is similar to MobileIP with route optimization. The focus of this work is on anycast, but the technique can be applied to mobility as well.

Snoeren and Balakrishnan [14] propose an architecture for supporting mobility in TCP that, like VIP, is designed to minimize dependence on new infrastructure. This architecture relies on a peer-to-peer protocol to update address information when a node moves. Our approach differs in two ways. First, we implement mobility at the IP level rather than the transport level. Conceptually, we believe abstracting IP addresses directly above the IP layer is a simpler approach, and this approach has the practical advantage of allowing one im-

plementation of virtualization to support a wide range of higher-level protocols, including TCP, RTP, ICMP, and UDP, and it supports straightforward integration with IPsec. Second, because we focus on supporting users with dozens of devices, we regard simultaneous movement of both ends of a connection as an important case—encountered, for instance, when a user carrying several devices exits a building—and we engineer our protocol to support it.

The idea of exposing names to applications and invisibly translating from name to physical address or route is a core idea in the Intentional Naming system [1] and TRIAD [2]. These systems, however, are more ambitious efforts to re-engineer the protocol stack. Intentional naming forgoes backwards compatibility, and both introduce translation to the routing infrastructure.

## 3 VIP architecture

The basic idea of VIP is simple; the VIP framework identifies a machine by its unique fully-qualified domain name (FQDN, e.g., example.acm.org), and the VIP layer on each machine maintains a mapping from FQDN to the physical IP addresses of peer machines so that it can direct messages addressed to an FQDN to that machine’s current location – its current physical IP address. As IP addresses change due to migration, VIP updates this FQDN↔IP mapping using secure dynamic DNS. But, because FQDNs do not change, communication transparently continues across physical IP address changes.

Unfortunately, current applications use IP address as the basis for communication and the FQDN merely as a means of obtaining it. We maintain backwards compatibility by virtualizing IP through a layer of indirection. Thus each FQDN is mapped to a 32-bit token, which we call a virtual IP address, that in turn maps to the physical IP address. We refer to the former as the VIP address or virtual IP address and the latter as the IP address or physical IP address.

The VIP address is integrated into the system by a VIP layer that resides immediately above the IP layer. Layers above VIP see and work with virtual IP addresses, which are merely backwards-compatible synonyms for FQDNs, and layers below VIP see the physical IP addresses required to route packets to their intended destination. A separate user-

level daemon maintains these FQDN $\leftrightarrow$ VIP address and VIP address $\leftrightarrow$ IP address mappings and updates the latter using dynamic DNS.

To simplify reasoning about security, the system uses IPSec to encrypt and authenticate all VIP communication. Following our goal of minimal infrastructure, this security scheme uses a simple peer-to-peer “anonymous” key exchange protocol similar to the one used in SSH [9].

The key design issues for the VIP architecture lie in the creation and maintenance of the two mappings, namely FQDN $\leftrightarrow$ VIP address and VIP address $\leftrightarrow$ IP address. The former deals with how we choose and associate virtual tokens with unique names and avoid collisions in these mappings. The latter resolves connection migration. In what follows we describe the design of these two mappings.

### 3.1 FQDN $\leftrightarrow$ VIP address negotiation

The assignment of VIP tokens to FQDNs should satisfy the following four properties.

- *Unchanging and unique.* A VIP is a synonym for an FQDN that has these properties. Furthermore, the mapping should be collision-free with respect to real IP addresses because legacy machines will not include a protocol for resolving collisions.
- *Symmetric.* Communicating machines must agree on what they call one another. Several higher layer protocols, including TCP, assume this property.
- *Scalable.* A mobile device will communicate with dozens or hundreds of servers and other mobile devices.
- *Lightweight.* This makes it easy for a user to deploy the system.

Unfortunately, it is difficult to satisfy these properties simultaneously with 32-bit tokens. We resolve this dilemma by relaxing the first requirement. In particular, VIP includes a negotiation phase that provides limited-duration VIP addresses that are pair-wise unique rather than permanent VIP addresses that are globally unique. Details of the negotiation protocol may be found in the technical report [15].

This negotiation typically adds one round trip time to connection set-up with VIP-enabled destinations. It also adds one additional DNS query to each host lookup.

The purpose of this query is to allow systems to efficiently detect legacy, non-VIP clients; it thus eliminates the need for the VIP daemon to send a probe to the remote host to verify its VIP capability. Note that, the additional DNS query is pipelined with the standard DNS query to minimize latency.

Because the FQDN $\leftrightarrow$ VIP address mappings are no longer *unchanging and unique*, VIP nodes must carefully control garbage collection of these mappings to maintain the abstraction that VIP addresses are synonyms for FQDNs. We discuss garbage collection policies in the technical report [15].

### 3.2 VIP $\leftrightarrow$ IP mapping

The set of VIP address $\leftrightarrow$ IP address mappings a machine stores can be thought of as a cache of mappings for the targets with which the machine communicates. This cache must be kept consistent with the true IP addresses of those targets. We use an invalidation protocol with leases [6] to accomplish this.

When a machine  $X$ 's IP address changes, it sends invalidation hints ( $VIP_X$ ,  $newIP_X$ ) to the VIP daemons on its *Active Partner List*, the set of machines with which  $X$  has communicated during the previous  $T$  seconds. Conceptually, the invalidation hints signal the receiver to query DNS for the new mapping the next time it sends a message to  $X$ , but since  $X$  sends the hints via IPSec and includes the updated values, the receiver can trust them and update its mappings immediately.

There are two cases when a machine  $Y$  will not receive an invalidation when a machine it had been talking to,  $X$ , moves: (a) the lease has expired or (b)  $Y$ 's IP address also changed. Therefore, any time a node  $Y$  sends a message to a node  $X$  from which  $Y$  has not received a packet from during the last  $T$  seconds,  $Y$  queries DNS to renew the mapping. In addition, if a machine does not receive an acknowledgment to an invalidation, it queries DNS to re-validate the mapping and then resends the invalidation hint. These procedures ensure that the mappings get updated quickly. When a connection is in active use, the corresponding mapping is updated through an invalidation hint otherwise the mapping is updated only when needed (before sending the first packet) through DNS lookup. Thus this scheme efficiently maintains the consistency of mappings.

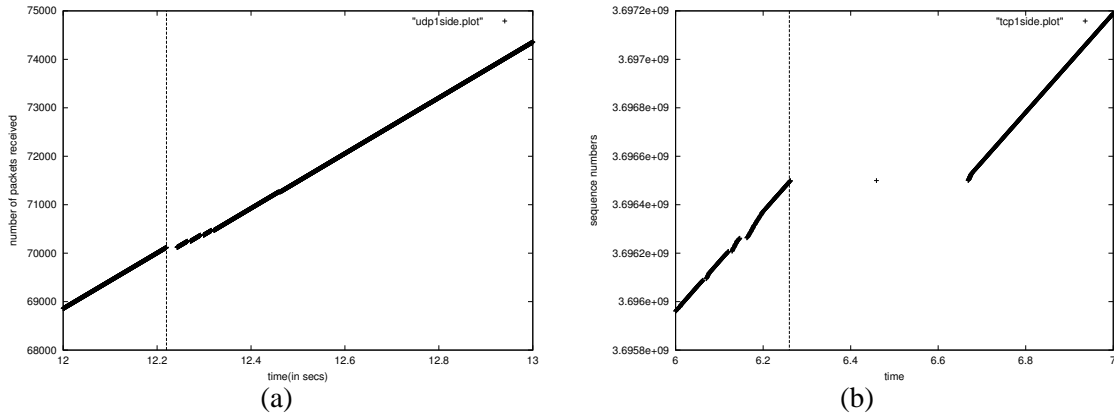


Figure 2: One-sided switch for (a) UDP and (b) TCP connection.

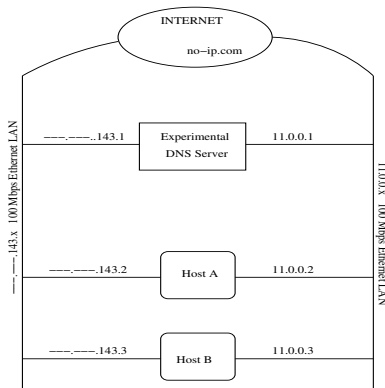


Figure 1: The evaluation testbed.

## 4 Prototype and Evaluation

Our prototype uses the Linux 2.2.17 kernel and modifies the IP/IP Encapsulation layer to implement VIP. We use FreeS/WAN 1.8 for IPsec. The modified name daemon, VIPD daemon, is currently implemented in Perl and runs at user level.

Our test bed is shown in Figure 1. Hosts A and B are 933 MHz Pentium III machines with 256MB of RAM. Each one of these machines has two 100Mbps Ethernet cards, which are connected to different LANs. The RTT measured by PING requests has an average  $250\mu s$  between the hosts when they are communicating on same LAN and  $450\mu s$  when communicating through interfaces on different LANs. The mobility of the hosts is emulated by deactivating of the interfaces and activating the other interface through the *ifconfig* command.

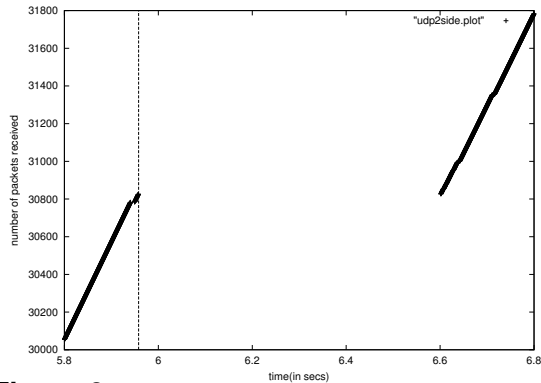
A third machine of similar hardware configuration is used for running *named*, the domain name server, a part of bind 9.1.1 from [www.isc.org](http://www.isc.org). This version supports signed dynamic updates [5]. We create a domain

*vipip.net* and assign each host a name from this domain. Each host also shares a secret key with the DNS server for dynamic update requests. This node also acts as a router between the two testbed LANs.

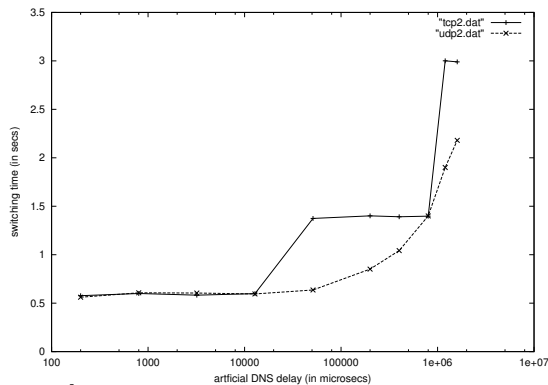
We have successfully tested our system for numerous applications running on both hosts that communicate using various protocols. These include a telnet session over TCP, RealPlayer video streaming over RTP and UDP, and ping over ICMP. Communication between application on two hosts continues successfully for both the cases of switching interfaces on one host (one-sided switch) and simultaneously switching interfaces on both hosts (two-sided switch). We have also successfully tested our system using *no-ip.com* as the name server by assigning our hosts names from the *no-ip.com* domain and configuring the hosts to send the dynamic DNS update requests to *no-ip.com*'s DNS server.

For measuring switching times, we implement a simple application with A continuously sending dummy messages to B using either UDP or TCP. We collect the communication traces using *tcpdump*. For TCP, we plot the sequence number of each packet against the time it was received. For UDP, we plot the number of UDP packets received versus time.

In Figure 2, we plot the traces for the one-sided switch case for TCP and UDP. For UDP, the observed switching time is about 23ms, including the time to send an update to the peer ( $\sim 20ms$ ) and acknowledgment time ( $\sim 3ms$ ). For TCP, switching time is about 400ms. This is longer than UDP because of a TCP retransmission timeout (200ms) and a *delayed ack* timeout (200ms) [13]. Figure 3 plots the behavior of UDP transmission for the two-sided mobility case. The switching



**Figure 3:** Two-sided switch on UDP connection.



**Figure 4:** Effect of DNS lookup times on TCP and UDP two-sided switch case

time is about 644ms which includes a 500ms timeout by the VIP daemon waiting for the ack of the hint it sends to the peer's VIP daemon. After the timeout, the VIP daemon does a DNS lookup and then sends the update hint to the other host at the correct physical address.

In Figure 4, we look at the dependence of switching time on DNS lookup latency for two-sided switching in both TCP and UDP modes of communication. The delays were simulated by inserting an artificial *usleep* into the local VIP daemon. As expected, UDP switching time increases almost linearly with latency. However, TCP exhibits a stepwise exponential curve corresponding to an exponential back off in retransmission.

Overall, we find that performance is good. For the two-sided switch case, fail-over time is dominated by the hint timeout, the round trip time to the DNS server, and the round trip time to the peer. Peer-to-peer updates appear to be a useful optimization. For the one-sided case, time is dominated by the round trip time between the peers and, for nearby peers, the overhead of our Perl-based name daemon.

## 5 Conclusion

VIP applies the principle of virtualization to IP addresses because exposing physical IP addresses to applications thwarts mobility and dynamic IP assignment, factors which will continue to grow in importance as mobility becomes more common, as users own increasing numbers of devices, and as the limited IPv4 address space is consumed. From the point of view of applications on VIP-enabled hosts, DNS names are used to address network traffic, and physical IP addresses are hidden.

A key benefit of VIP's implementation of virtualization is that its design emphasizes incremental deployability. Powerful building blocks for virtualization now exist so that virtualization can be done almost entirely by relying on existing infrastructure. As a result, VIP is simple enough to deploy that, for example, a Linux hobbyist could easily deploy and benefit from the system.

## References

- [1] W. Adje-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Symposium on Operating Systems Principles*, pages 186–201, 1999.
- [2] D. Cheriton and M. Gritter. TRIAD: A new next generation Internet architecture, 2000.
- [3] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6). Request for Comments 1883, Network Working Group, December 1995.
- [4] R. Droms. *Dynamic Host Configuration Protocol*. IETF, October 1983. RFC 1531.
- [5] D. Eastlake. Secure Domain Name System Dynamic Update. Technical Report RFC-2137, Internet Engineering Task Force, Apr 1997.
- [6] C. Gray and D. Cheriton. Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency. In *Proceedings of the Twelfth ACM Symposium on Operating Systems Principles*, pages 202–210, 1989.
- [7] S. Gupta and N. Reddy. A Client Oriented IP Level Redirection Mechanism. In *Proceedings of INFOCOM 99*. IEEE, March 1999.
- [8] S. Kent and R. Atkinson. "Security Architecture for the Internet Protocol". Technical Report RFC-2401, Internet Engineering Task Force, Nov 1998.
- [9] OpenSSH. <http://www.openssh.com>.
- [10] C. Perkins. IP Mobility Support. RFC 2002, IETF, October 1996.
- [11] C. Perkins, A. Myles, and D. Johnson. The Internet Mobile Host Protocol (IMHP). In *Proceedings of INET*, June 1994.

- [12] C. E. Perkins and A. Myles. Mobile IP. *Proceedings of International Telecommunications Symposium*, pages 415–419, 1994.
- [13] Ed. R. Braden. *Requirements for Internet Hosts – Communication Layers*. IETF, October 1989. RFC 1122.
- [14] A. C. Snoeren and H. Balakrishnan. An End-to-End Approach to Host Mobility. In *Proc. 6th International Conference on Mobile Computing and Networking (MobiCom)*, 2000.
- [15] P. Yalagandula, A. Garg, M. Dahlin, L. Alvisi, and H. Vin. Transparent Mobility with Minimal Infrastructure. Technical Report TR-01-30, Department of Computer Science, The University of Texas at Austin, July 2001.