Spread–Identity mechanisms for DOS resilience and Security.

Dhananjay S. Phatak
Computer Science and Electrical Engineering Department
University of Maryland Baltimore County (UMBC)
phatak@umbc.edu

Abstract

*The explosive growth in wireless (and wired) networking technologies and services indicates that multiple means of network connectivity will become available in the near future. For example, stationary and mobile hosts currently support Internet access via wired LANs, Wireless LANs/PANs (e.g., 802.11x, 802.15) or wide area wireless cellular phone and data networks (like GSM). In essence, heterogeneous multi-homing is now a necessity for all hosts (mobile or non-mobile). In order to tap the full potential of such heterogeneous multi-homing, we introduce the novel "***Spread Identity (SI)***" communications paradigm. Therein, the concept of multi-homing is extended to allow each interface to simultaneously assume multiple addresses and dynamically acquire and release them as needed which is tantamount to "Spreading Identity" at the network(IP) level and has fundamental implications for security.*

In this paper we show how the spread Identity mechanisms can effectively (1) Mitigate DDOS attacks by rate-limiting the number of name-resolution responses. (2) Quickly detect and neutralize resource-overload type DDOS attacks that cannot be prevented by rate-limiting (3) Enable surviving the remaining types of DDOS attacks by quenching destination addresses they target (in essence by changing the Identity) (4) and preventing future attack flows by returning NULL addresses, and re-directing the attackers against one-another.

We demonstrate that Spread Identity mechanisms can also be leveraged to bolster the security of single source-to-destination flows. SI mechanisms can attain the same level of security as that of a single link with Strong Security Infrastructure (SSI) at a lower cost (in terms of the infrastructure required and the encryption effort needed).

The fundamental concept of Spreading-Identity revealed herein is more general and potentially applicable to other scenarios beyond Internet/Electronic communications.

1 Introduction

As wireless and wired network services continue their explosive growth, it is clear that multiple network transport conduits and mechanisms will be available to hosts. For instance, stationary (non-mobile) hosts in an organization could have access to the Internet through dedicated lines (such as T1), cable modems, DSL connections, Wireless LANs (within the organization), wide area GPRS/CDMA services and lastly via cell-phone modems. Typical academic scenario allows stationary hosts at least two types of connections. wired-Ethernet and 802.1x wireless LANs. Business organizations dealing with Information Technology tend to have more than one independent connections to the Internet for reliability purposes. Likewise, a mobile host (ex: a laptop) can have access to the Internet via multiple networking technologies such as wired LANs (when at a hotel, conference, etc.), wireless LANs (for instance Bluetooth, 802.11, Airport LANs, GPRS, WCDMA, etc.), and cellular phone modem(s), where each technology has a corresponding service provider.
It is then a dynamic optimization problem to select which combination of available network services to use at any given time, which is tantamount to a dynamic selection of *data transport service(s)* (DTS or dynamic transport selection). Factors such as bandwidth/throughput, latency, jitter, quality of service (QoS) and security requirements, cost, power consumption, residual battery life, interference, and traffic patterns should be taken into account with relative priorities determined by an application or user.

In essence, heterogeneous multi-homing is on the rise and is likely to become a necessity for all hosts (mobile or non-mobile). In order to fully exploit the true potential of multi-homing, we've introduced the novel "**Spread Identity (SI)**" communications paradigm [56]. Therein, the concept of multi-homing is extended to allow each interface to simultaneously assume multiple addresses spanning different subnets and *dynamically* acquire and release them as needed. The ability to dynamically change addresses has fundamental implications for security. For instance, snoop-

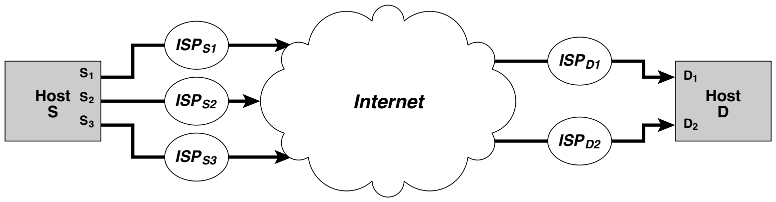


Figure 1. Communicating multihomed hosts.

ing/compromising a connection is harder if the source and destination dynamically change their addresses during the session. At the network layer a host is *identified* by it's IP address. Hence dynamically changing addresses during communication is tantamount to "Spread Identity" (SI) communications. In this paper we show that SI mechanisms in conjunction with other schemes make it possible to deploy radically new, simpler and effective ways to enhance Distributed Denial of Service (DDOS) resilience (this is explained in detail in Section 2). Spread Identity mechanisms can also be leveraged to enhance the security of single source-to-destination flows (henceforth abbreviated S-SD flows) (Section 3).

The "Spread Identity" paradigm is analogous to (and inspired by) "Spread Spectrum Communications" (hence the title substring "Spread Identity"). Spread-Spectrum communications were invented during World War II and have been widely deployed since then (CDMA is one type of Spread Spectrum communication). However, spread-spectrum techniques have so far been restricted only to the physical layer. This paper explores the usage of similar mechanisms at higher layers (network and application layers).

Moreover, the fundamental concept of Spreading-Identity revealed herein is more general and potentially applicable to other scenarios beyond Internet/Electronic communications (for example SI could be applicable to databases, storage and archival systems).

1.1 Background and Related Work

Figure 1 illustrates the case when two multi-homed hosts, S and D, wish to communicate via the Internet. For instance, host S might be a laptop with network access via an 802.11 LAN (say interface S1), Wired Ethernet (interface S2), and a Wide area GPRS/CDMA data service (S3). In general, the IP addresses assigned to interfaces S1, S2, and S3 will be controlled by separate Internet Service Providers (ISPs) who might in turn implement filtering to various degrees. For example most routers will egress-filter packets with spoofed addresses: if host S were to send a packet with the IP address of interface 1 on say interface 2, the ISP2's router will (legitimately) consider it as source address spoofing and drop the packet. We denote source-node by S, destination node by D and a man-in-the-middle attacker by M. The attacker M could snoop/modify the traffic between S and D or simply hijack the connection leaving the source or the destination with the illusion that it is talking to the intended peer. Throughout the rest of the paper *n* denotes the number of interfaces available to a node (so that, in Figure 1, for node S, $n = 3$).

Since the availability of *affordable* multiple interfaces and network services is a recent development the work published in the literature so far has focused on bandwidth aggregation and mobility support.
Bandwidth aggregation across multiple interfaces has been considered at various level levels of the networking stack: Link layer (a sampling can be found in [43, 67, 77, 78, 60]); Network Layer [9, 68, 57, 58]; Transport Layer [41, 49, 30, 79, 26]; and Application Layer [66, 12, 29].

In recent prior work [26], we have demonstrated that the same attributes that are needed to support efficient data striping across multiple interfaces also enable end-to-end transport layer support for host mobility. There [26] a unified transport layer framework for data striping and host mobility was presented, analyzed and shown to effectively solve two seemingly independent and important problems (data striping and host mobility). So far the focus in the literature has been on bandwidth aggregation, jitter control etc.
To the best of our knowledge, security aspects of multi-homing and multiple-interfaces have not yet been adequately addressed.

This paper addresses this important issue. The next section describes our novel SI mechanisms for DOS/DDOS resilience. Therein we first describe the enabling mechanisms. Then we present our DOS resilient architecture and illustrate how we **leverage the name-resolution process as an implicit token-granting mechanism which enables rate-limiting as well as rapid detection of abnormal traffic**. We then classify the types of DOS attacks based on what they target and describe in detail how each type of attack can be effectively countered. The next subsection discusses the practical considerations and demonstrates that the proposed scheme can be easily implemented. The next subsection weights the advantages of the proposed mechanisms over other related methods proposed in the literature and discusses its potential drawbacks. Section 3 investigates deploying multiple interfaces for security of individual peer-to-peer flows. The final section presents our concluding remarks.

2 Spread Identity mechanisms for DOS Resilience

The entity which is seeking DOS resilience is assumed to be a server "$\mathcal{S}$" within some organization X (as opposed to an individual's laptop). Accordingly, a small amount of extra network infrastructure and computing hardware overhead is assumed to be affordable. We present the analysis assuming that $\mathcal{S}$ has "n" interfaces and multiple (say k) addresses assigned to each interface at any given time. However, we would like to emphasize that the DOS resilience scheme illustrated in this section works even when $n = 1$, i.e., with a single interface and multiple IP addresses. First we describe the fundamental mechanisms needed for effective "spreading" of Identity.

2.1 Fundamental Mechanisms

2.1.1 Dynamic Extended Multi-homing

Multi homing refers to a scenario where a node has Multiple IP Interfaces each with a distinct IP address. We extend the conventional multi-homing concept to allow each interface to have multiple IP addresses (this is known as IP-aliasing in the Linux world) and dynamically acquire and release them. Ideally, in order to complicate the adversary's task, the IP addresses that are doled out should not be contiguous chunks and if possible, they should span different subnets. This extension is not new, rather the way this mechanism gets exploited to dump unwanted flows onto specific address and then terminate them by quenching the addresses is a novel scheme (see Section 2.5.7 for further details.)
Note that acquiring and releasing multiple address is feasible without any changes to the existing DHCP infrastructure. Multiple addresses can be acquired via multiple requests. If the DHCP server allocates only one IP address per MAC address, then the DHCP client could trivially fake multiple MAC addresses.

The scarcity of IP addresses in IPv4 suggests that this scheme could be difficult to implement. However, IPv6, the next generation of IP protocol makes abundant number of IP addresses available. Even in IPv4, it is conceivable that not all IP addresses available to a DHCP server of a large ISP provider like Comcast Cable are in use. This is likely to be true because not all people leave their PCs on all the time.

2.1.2 Spreading Identity by controlling Name Resolution

Simply acquiring multiple IP addresses is no good unless the entity/node that acquires those addresses can itself (selectively) reveal addresses as it pleases. In other words, in order to truly "spread" identity, two things are necessary:
(1) Dynamic access to multiple addresses
(2) Control over how the addresses are revealed, i.e., over the name-resolution process.

Control over name-resolution is easy to achieve: any organization/entity (say X.org) that registers its domain name with IANA can specify the machines or entities that will act as the primary/authoritative DNS server (henceforth abbreviated ANS or Authoritative Name Server) for X.org. In our scheme, entity X itself controls it's ANS with one simple modification: the ANS simply relays a request for resolution of the name "X.org" back to a some node/entity within X itself; if this "self resolution" capability and desire is indicated by simply setting a flag. Now X can resolve its name to any of it's current addresses. The response from X is relayed back to the source through the ANS. Note that if the flag is not set, the ANS responds exactly like a normal DNS server (i.e., it looks up a default record for X.org and sends the information).

Note that the name-resolution query syntax is untouched which ensures complete compatibility with current DNS system. The only minor change is restricted to the "backend", i.e., how the query is handled by the ANS. The frontend is the same (all the world can send the exact same resolution requests as before).

2.1.3 Address Resolution Strategies

In general, network-level Identity Spreading can be accomplished by controlling name-resolution:

$$\begin{aligned}\text{IP-addr-returned-in-name-resolution} &= \\ \mathcal{M}[\text{source-address of the query, state}] &\qquad (1)\end{aligned}$$

where ideally the "state" information is embodied in small number of parameters and $\mathcal{M}$ is a simple (lightweight) mapping. The state could null, i.e., the IP address returned could depend only on the source address of the request.

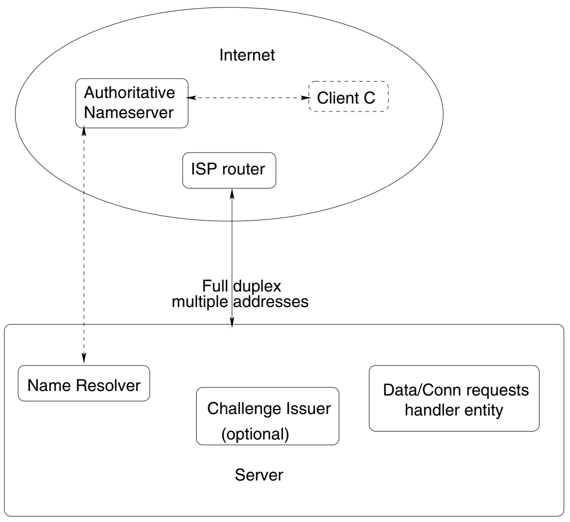
To illustrate the main idea consider the following example: Suppose the main data server $\mathcal{S}$ in X.org has 3 connections/interfaces. Then all name resolution queries for X.org from source nodes in the US get doled out addresses from a subset S1 of the addresses assigned to link-1. Likewise all name resolution requests from nodes in say China get doled out addresses in another subset S2, and all queries from say Brazil get returned addresses from a third subset S3. In essence, S returns different IP addresses to different sources. Now if under DDOS attack, S could selectively turn off all traffic flows say from Brazil (by using ARP updates or NULL routes and relinquishing the addresses that were revealed to hosts in Brazil).

Note that several other address revealing strategies are feasible. The simplest one is a round-robin issue of all addresses in response to successive requests. Other possibilities include address revealing based on time-of-day sensitivity, load balancing considerations, etc.

Also note that *n* physically distinct interfaces are more effective but are not necessary. A single interface with IP addresses from *n* distinct (preferably non-contiguous) IP networks assigned to it would suffice for the purpose of Identity Spreading.

2.2 The proposed DOS resilient architecture

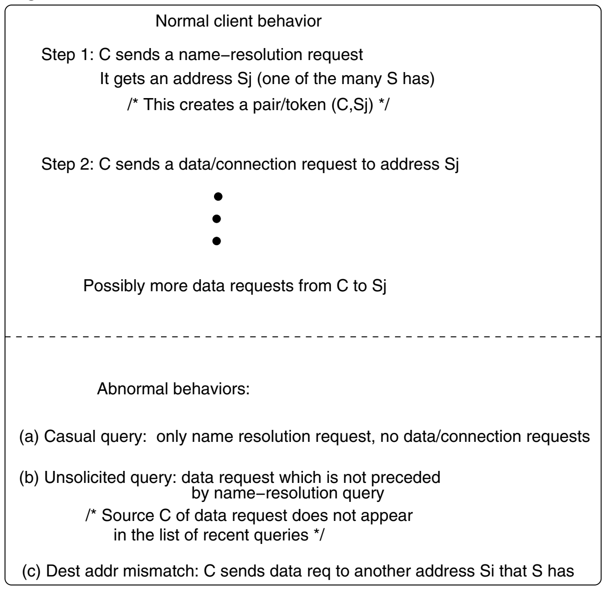
The overall architecture is illustrated in Figure 3. The DOS resilient server S conceptually includes 3 entities
(1) Name resolver
(2) Data/Connection request handler
(3) an optional challenge issuer.
These entities could be processes within the same node or nodes within an organization's premises that are internally connected by a LAN. The name resolver entity uses one fixed IP address to communicate with the Authoritative Name Server (ANS) responsible for server S. This address is NEVER published to the outside world. All other addresses are part of the dynamic Identity Spreading (i.e., these addresses are revealed to the outside world and are periodically refreshed.)
As seen in Figure 4, a typical/normal session between some client *C* the server $\mathcal{S}$ includes the following steps:
(1) C sends a name-resolution query for $\mathcal{S}$
(2) The ANS for $\mathcal{S}$ forwards the query to $\mathcal{S}$
(3) depending upon the current state and the source address of C, $\mathcal{S}$ responds to the request. If the current load is heavy, $\mathcal{S}$ could simply delay the response or if under extremely heavy load $\mathcal{S}$ could return a NULL address. If $\mathcal{S}$ estimates that it is under a DOS attack, it can again delay the response, return a NULL address or take more radical actions like return one attack-bot's address in response to a query from another (suspected) bot.
(4) If a valid address S_i (one of the multiple addresses $\mathcal{S}$ currently has) was returned, then the pair (C,S_i) is added to the list of recently-resolved-queries (abbreviated the "RRQ-list").
(5) The response from $\mathcal{S}$ is relayed back to C by the ANS.
(6) C then sends one or more data requests. that are processed by $\mathcal{S}$ depending upon the outcome of a simple "token matching" test and the current state of the server.


Internet
Authoritative Nameserver
Client C
ISP router
Full duplex multiple addresses
Name Resolver
Challenge Issuer (optional)
Data/Conn requests handler entity
Server

DDOS resilient server architecture

Conceptually the server entity includes the name−resolver entity,
the data/connection req handler entity and an optional challenge issuer.
The entities could be processes within a host or nodes within an
organization's premises connected by a LAN

Figure 3: DOS resilient architecture.



Normal client behavior

Step 1: C sends a name−resolution request
It gets an address Sj (one of the many S has)
/* This creates a pair/token (C,Sj) */

Step 2: C sends a data/connection request to address Sj

•
•
•

Possibly more data requests from C to Sj

Abnormal behaviors:

(a) Casual query: only name resolution request, no data/connection requests

(b) Unsolicited query: data request which is not preceded by name−resolution query
/* Source C of data request does not appear in the list of recent queries */

(c) Dest addr mismatch: C sends data req to another address Si that S has

Figure 4: Token matching quickly identifies abnormal behavior

2.3 Leveraging name-resolution as an implicit Token-granting process

This is one of the main novel ideas. As mentioned above, after each resolution request is processed, the pair (C,S_i) = (query-source-address , resolved-address) gets added to the RRQ-list. Under normal conditions, the (source,destination) address pair in every incoming packet should match some (entry) in the RRQ-list. **Consequently, each pair (C,S_i) in the RRQ-list can be effectively used as a "token" at the IP level as follows:**

1. After a name resolution step, future communications from C are expected to be directed to destination address S_i. If C sends a packet to some other address S_k then it is not normal and could happen because of any of the following reasons:
   (i) C could have an old cached address, or
   (ii) C is an attack bot trying to flood all addresses.
2. If the source address (say H) in an incoming packet does not appear in the RRQ-list, then it is an "unsolicited" query and is not normal. Once again, this could happen either because H got old cached address (cached by the application itself or by some secondary DNS servers), or because H is an attack bot that learned the addresses of S from other bots without making a name resolution query.

Once an abnormality is identified, further action can be taken depending on the state of $\mathcal{S}$ (ex: it could re-direct a request to the challenge-server in order to give genuine clients a chance to redeem themselves. If under extremely heavy load, $\mathcal{S}$ could simply filter off abnormal packets).

This mechanisms raises several questions including
(1) How long should the list be and how often it is refreshed (this is addressed in Section 2.6, 2nd item)
(2) How does application and DNS caching affect this method of outliers detection (this is also addressed in Section 2.6, 4th item).
(3) How to distinguish a genuine client ? This is simple to do: an abnormal (token mismatched) data request could be re-directed to a challenge-response server to screen for a human in the loop.

2.4 Classification of DDOS attacks based on what they target

DDOS attacks can
(1) clog the bandwidth by sending a lot of packets (we call this input bandwidth overload, abbreviated BO)
(2) overwhelm resources (memory/CPU, etc, by employing mechanisms like a SYN floods that consumes buffers, or repeated invocation of CGI scripts. these are abbreviated as RO);
(3) A truly successful DDOS attack is indistinguishable from extremely heavy load: too many bots send requests (indistinguishable from genuine client requests) at once. This could overload the output bandwidth (too many clients requesting huge downloads) or server resources. This type of attack is abbreviated as "GO" (disguised as Genuine Overload).

The third type attack is the worst because it might not get detected as an attack. The next highest vulnerability is to type-2 attacks (resource overload attacks). Input Bandwidth clogging is a brute-force attack requiring adversary to generate a lot of traffic. Our scheme can offer very effective protection against all types of attacks. Especially those of types (3) and (2), the most dangerous/damaging attacks.

2.5 Comprehensive DDOS mitigation

2.5.1 Prevention of overload by Rate Limiting the number of Name Resolution Responses

A fundamental strategy to mitigate DDOS attacks is to limit the number of name-resolution query responses returned in any given interval (i.e., limit the rate at which responses are sent). In essence, the server $\mathcal{S}$ knows it's capacity and accepts no more potential connector nodes than it can handle.

Note that the most dangerous attacks where all bots behave like "normal clients" will be effectively mitigated by this rate limiting strategy. To appear a like genuine client a bot B must make a name-resolution query (otherwise B's source address will not appear in the RRQ-list and its traffic could be filtered off at the IP level thereby foiling the attack). Forcing a name-resolution step in turn implies that the bots are subjecting themselves to rate limiting mechanisms and will not be able to bring the server down.

2.5.2 Early detection of abnormal behavior

Type-3 attacks are mitigated by the rate-limiting strategy above. The next most damaging attacks are type-2 (resource-overload or RO) attacks where the bots don't try to masquerade as genuine clients (i.e., don't make name-resolution queries). For instance, a few bots could learn all the addresses $\mathcal{S}$ has and distribute them to a large number of other bots. All the bots then send traffic. Note that such "unsolicited requests" (the communications that are not preceded by name-resolution requests) will token-mismatch and get flagged as abnormal right away.

2.5.3 Implicit token matching admission control to prevent server resource overload attacks

The big advantage of the token matching process is that a mismatch can be inferred at the IP level. The server could simply filter out unsolicited traffic at the IP level, so that it never consumes server memory/CPU and other resources (for instance, for TCP SYN flood to be effective the packets should at least reach TCP level. Our scheme makes it feasible to filter them off at the IP level. The same holds for CGI request floods).

In other words RO attacks can be neutralized by filtering unwanted flows at the IP level. Our implicit token-matching process greatly facilitates the identification/tagging of flows to be filtered.

2.5.4 Ability to screen automated traffic

Note that when under attack or heavy load, the server can simply reject token-mismatched requests by IP level filtering. However this could cause problems in cases where the mismatched was caused by caching of addresses (caching obviates the need to make repeated name server queries).

To mitigate this problem our architecture includes a challenge-response server. All requests that deviate from expected behavior are re-directed to the challenge-server entity. It accepts a connection/data request and issues a simple challenge that is understandable only by a human. Those requests that do pass the challenge are re-directed to one of the real addresses of the data server. Those that fail the challenge can be terminated or re-directed depending on the state the server finds itself in.

2.5.5 Stopping undesired future flows right at the source

If the data server is under attack or under heavy-load the name- resolver can return a (i) NULL address or (ii) a non-routable address (127.0.0.1 so that the attack traffic never leaves the host or 192.168.x.x type address) (iii) the source address in the query (reflection).

Likewise, if the client fails the challenge test the challenge server entity can re-direct (like a web redirection) the data request to NULL, non-routable or the source's self address.
The data server entity can also handle with unsolicited requests in the same way when under attack.

Note that this defensive mechanism would stop the traffic right at the source: if name resolution fails or the address returned is non-routable, or is the address of the source itself (reflection) the attack packets never leave the source node.

2.5.6 Take the fight back to the attackers: re-direct the bots to attack each other

The most interesting possibility is that source is returned the address of another current attacker (or its data request is re- directed to the address of a current attacker), so that bots clobber each other.

Note that the name resolver itself could direct an incoming request against a current attacker by returning the address of an attacker. However, such a redirection is a drastic step and we would like to be sure that a genuine customer is not being sent off to attack another bot. Hence such a mutual clobbering redirection is done only if the challenge-response fails AND the server is currently under attack.

There is yet another possibility: the server organization could pay some ISP to provide packet sinks (an opportunity for yet another value added service). The server node could then direct undesired flows to the packet-sink address.

The main point is that control over resolving one's own addresses opens up a number of flexible and powerful options.

2.5.7 SI: the last line of defense against on-going attacks

Unsolicited RO floods can be prevented from reaching above the IP level, However the floods can still consume bandwidth and choke the traffic. This is where another subset of SI mechanisms is the last line of defense.

Continuing with the example above, suppose node S has 3 interfaces and addresses from interface1 get exposed to sources in the US, interface-2 to China and interface-3 to Brazil.Now if a Distributed Denial of Service (DDOS) attack is mounted on node S, it can selectively turn off traffic flows. For instance, in the above example, S could decide that the traffic was getting overwhelming and turn off traffic flows originating from say Brazil. To turn off the flows, it Quenches the address that are the targets of undesired flows. To quench an address, node S does two things

1. Updates the MAC entry corresponding to those addresses (which got doled out to requests from Brazil) to say NULL. This is equivalent to updating the ARP entry in the router by replacing certain IP addresses with NULL entries. Those packets now get dropped at the last-hop-router (don't reach S).

   Note that the server could be behind a perimeter-router (PR) belonging to an organization. Here, the last-hop router is the PR and simply updating the ARP entries will not work because the bottleneck is the link between the PR and ISP's router. This problem can be trivially fixed: In such a case it is reasonable to assume that the perimeter router (PR) participates in route- information exchange protocols (such as BGP). Then the perimeter-router can instruct the ISP's router to create a null-routes corresponding to the addresses (that belong to S) that are being quenched.

2. All (or any subset of) new queries from nodes in Brazil get returned a NULL address. This quenches flows right

at the source: if the name resolution fails, the source cannot generate any traffic.

After an address is "quenched" it is relinquished and a new address is acquired. We can make it a bit more difficult to learn multiple addresses of $\mathcal{S}$ by repeated queries as follows. If the source address or source-subnet of a new query is in the RRQ-list then a new address is not revealed, rather the same address (that was previously doled out) is returned again (unless it is being quenched in which case a NULL address is returned).

2.6 Practical Considerations

Overhead on the DNS system : First, note that the overhead on the DNS system is negligible. All that is required is that the Authoritative Nameserver (ANS) for entity S relays requests for S back to S itself. This easily achievable. In fact entity S can set up its own NameServer(s).

Implementation overhead at the server is easily affordable: The implementation of such a scheme at the target server machine is very simple.
A vital question is how big is the list of tokens and how often is it refreshed ? The following simple calculation helps to get an estimate. Assume the server is connected by full duplex T1 links and serves generic web pages (as opposed to huge images, video on demand, big software distributions, etc). From web data [32] assume that each web page is about 50 Kbytes and should be delivered in 40 seconds (a conservative estimate. Patience threshold seems to be closer to 10 seconds). Then number of simultaneous connections sustainable is 154.4 with each request taking 40 seconds to satisfy. TCP times out after 2 mins. Even if we allow a much longer window of 640 seconds (well over 10 minutes), only about 155 x 16 or about **2480 pairs must be maintained in the RRQ-list** in order to keep the data server full. This trivial with today's processing power.

Attacks on the name-resolver entity are easier to handle: Note that an attack on data server is more effective because of the asymmetry. A simple get request on a big object can keep the server tied for a much longer time. On the other hand the adversary needs a much higher effort to mount an attack on a DNS server.

Interaction with DNS and application caching: For our scheme to work efficiently, the name-resolution replies must not be cached by other nameservers in the control plane (routers, nameservers and other network control entities constitute the control plane). In regular DNS, this can be easily achieved by setting the DNS TTL field to indicate no-caching at all, or the absolute minimal available caching interval. It is feasible to achieve low-caching period: Akamai does a similar thing for load balancing/reliability purposes, i.e., they return different IP addresses for successive queries for the same client they serve (such as cnn.com) [42, 11]. Their DNS replies have a TTL field value of 20 seconds, (which corresponds to one or two web pages viewed). So after this time period a client will go back to the Akamai name server to get the IP addresses which enables Akamai to load balance and hide node failures. (There are major distinctions between Akamai's and our methods and they are explained in detail in Section 2.7.2 below).

Note that periodic refreshing of addresses invalidates old cached values of IP addresses corresponding to a name. Hence the attacker must query the nameserver to get current IP addresses of the target entity S.

2.7 Advantages of the proposed scheme

① Multi-level multi-pronged defenses against DDOS attacks (Section 2.5).
② The overhead on DNS system is negligible, the spreading is actually handled by target node itself. Likewise, the memory and computation overhead on the target entity is easily affordable (Section 2.6).
③ Name-resolution query syntax need not change (which ensures backward compatibility).
④ Our scheme enables load balancing and supports end-to-end host mobility at the transport layer
⑤ This scheme does not require cooperation among the routers. It only relies upon collaboration between end node (S) and its ANS server and the last-hop routers connected to S. The solution is therefore closer to a true end-to-end solution and more lightweight than any scheme that requires all routers to collaborate.
⑥ It nicely complements existing DOS mitigation schemes: for example, the destination addresses which are being turned off can themselves serve as part of tags marking flows to be filtered out by intermediate routers.
⑦ Spreading should work even better under IPv6 because of

1. the huge address space.
2. hosts can generate part of the address
3. native support for multiple addresses per interface and multi homing
4. hooks to migrate existing transport layer connections should the network layer address change and has a very rich variety of options and mechanisms.

All these are ideal vehicles to implement Spread-Identity communications.

2.7.1 Potential drawbacks

(1) Address scarcity in IPv4
(2) If other DNS servers don't abide by a low TTL and

cache the entries for longer period then an attacker node could learn multiple addresses by a recursive DNS query (but this problem can be handled the same way Akamai does today, as explained above).
(3) Stale values (cached by either the application or by other DNS servers that might override low TTL) can cause a delay in connection from a genuine client.
(4) DNS caching for very small periods implies more name resolution queries, i.e., more traffic. This overhead is likely to be small because applications can cache the resolved address.
(5) Entity $\mathcal{S}$ could maliciously return fake addresses for others
(6) If the address relinquished by S gets assigned to someone else, that entity will become the unsuspecting victim of the attack

Many of these drawbacks are specific to IPv4. Under IPv6 most of these problems are easily solved.

2.7.2 Comparison with and distinctions from related work

As mentioned above, Akamai does a somewhat similar thing within their own network: the same name resolves to different IP addresses depending upon the source address and time [29, 30]. However their main goals are load balancing, fault tolerance and higher throughput. For example node-xyz.akamai.net could resolve to node-1 for one request and to node-2, node-3... etc for subsequent requests. This way when multiple sources request the same web-object, multiple nodes serve those requests making the throughput higher.
Our approach is substantially different, we list the differences:

[1] Their focus is not on security and hence they do not assign multiple IP address the same interface/node. Unlike this work, they do not consider the fundamental concept of Spreading the Identity.
[2] Another vital difference is that in their scheme the identity/name holder has no say over how the addresses should be doled out (policy is completely decided by Akamai). In a sense, our solution is more distributed, closer to the end-to-end ideal.
[3] Their schemes cannot quench flows right at the source the way our solution does.
[4] Implicit token matching mechanisms is our novel contribution (Akamai obviously does not do such token matching).
[5] Our solution subsumes support for host mobility. Their schemes do not address host mobility.
[6] We show how the same Spread-Identity infrastructure (which is introduced for DOS resilience) can also be leveraged to enhance the security of flows between arbitrary nodes.
[7] Their product is specifically geared toward reliable, high throughput web-content-hosting for their clients. Consequently whatever they have done is restricted within their domain (i.e., it works only inside akamaitech.net). Indeed some of the Akamai documents cited are Proprietary and Confidential. One has to dig through web logs to reverse engineer their methodology.

In closing we note that mechanisms similar to the SI schemes have been proposed to counter email spam [33]. The SI paradigm is more fundamental and general. It's development happened without the knowledge of the existence of the anti email-spam techniques proposed in [33].

3 Spread Identity to enhance security of peer-to-peer flows

The basic security issues of authentication, integrity, confidentiality, and non-repudiation [74, 63] are effectively solved by a strong security infrastructure (henceforth abbreviated SSI) which includes:
(i) a public-key infrastructure used for key exchange, certification and authentication and
(ii) strong symmetric-key algorithms for encryption (e.g. AES) and data integrity (e.g. HMAC-SHA).
This is independent of how many links are available for communication. In other words, a single link with a SSI can effectively address the basic security issues. Hence, any security attributes of communicating across multiple interfaces should be addressed in the context of what is achievable with a single link alone. Hence this section focuses on leveraging multiple interfaces to achieve the same level of security as that of a single link using SSI with less cost (in terms of computation, delay, infrastructure required, etc.). In this section we assume that a host has n physically distinct interfaces where $n > 1$.

The PKI infrastructure is the most expensive (it must be pre-deployed and is global in nature). Hence the most obvious question is "can the availability of multiple interfaces obviate the need for PKI" ? The answer is "no" because the PKI address a vital issue viz., a (third-party) authentication (i.e., a verdict from a third independent entity that the peer destination entity is indeed who it claims to be. This is obviously important when communicating with one's bank etc., real time control of power grids, etc.). A third party authentication is an attribute that is completely orthogonal-to/independent-of other security attributes such as confidentiality and integrity (which can be effectively addressed by multiple interfaces at a much lower cost).

A third party authentication (for example, digitally signed certificates from certification authorities (CAs)) necessarily involves out-of-band communication to establish trust/prior secret. For instance in the case of digital certificates, the web browser software is already pre-programmed to recognize well known CAs and their public keys. The distribution of the browser software thus constitutes the "out-of-band" communication that bootstraps the trust-establishment process. The entity using the browser can now trust whoever is certified by one of the CAs recognized by the browser.

Multiple links make it possible to realize the "out-of-band" communication in most cases. For instance, a host could reserve a link (the one that is most difficult to tap/snoop on such as a wired or point-to-point infrared link) specifically to communicate with a CA. (for instance the source could bring the link up and acquire a new address only to communicate with the CA and after the communication with the CA is done, it brings the link down again). While this can be considered to offer a channel for the out of band communication, it has not obviated the need for the CA itself (i.e., even if the channel is available, a CA must also be available).

Thus, no amount of multiple interfaces cannot obviate the need for a third party authentication infrastructure. This in turn brings up the question: once such a PKI with CAs is available, what additional gains if any can multiple interfaces provide ? In critical communications multiple interfaces can further enhance reliability, availability and security, however, a single interface can also deliver sufficiently high security (together with PKI, CAs) so that the additional security gained might not add value.

However, there is a large class of communications (for instance transferring files between work and home machines, most of browsing, etc.) that are not so critical as to warrant the certification overhead. Furthermore, in many scenarios like mobile ad-hoc or sensor networks PKI simply may not be available or feasible. In all such scenarios, multiple interfaces can be leveraged to provide a good level of security during the secret establishment phase without needing a third party authentication.

3.0.3 Key Agreement without Public-Key Cryptography

Threshold cryptography [64, 16, 37] : It allows a secret to be split into T pieces in such a way that at least k of the pieces are required to recover the secret, where $k \leq T$. Furthermore, $k-1$ of the pieces can be compromised without revealing any information about the secret. In the multiple interface scenario, the sender sets $T = k = n$, i.e., the key is split into n pieces and each piece can be transmitted along a different interface. The adversary must now compromise all n interfaces (any less would be insufficient) and this requires a considerably larger effort.

In essence, if the possibility of an attacker snooping on all interfaces is sufficiently low then multiple interfaces obviate the need for Public Key cryptography for establishing a symmetric key. This will reduce the computation overhead as well as the infrastructure overhead associated with the PKI. (Key exchange across multiple paths has been investigated by many researchers, for example see [65, 22]

Mitigating the threat of man-in-the-middle attack in an unauthenticated Diffie-Hellman key agreement : In essence, multiple interfaces significantly lower the possibility of man-in-the-middle attack. Note that with multiple interfaces, the adversary must tap/compromise all of them, otherwise a shared secret can be established (in a number of ways including the Diffie-Hellman exchange). Furthermore, merely snooping does not prevent S and D from establishing a secret. Adversary $\mathcal{M}$ must be able to arbitrarily modify the traffic on ALL paths between S and D in order to mount a successful attack.

Hence a Diffie Hellman exchange across multiple interfaces is a better way of establishing a shared secret without using Public-Key cryptography (better than simply sending the key using threshold cryptography because to break that exchange the attacker only needs to be able to snoop on all links. To compromise a Diffie Hellman exchange, the attacker needs to be able to modify traffic on ALL paths).

3.1 Equivalent Security with Reduced Data Encryption Effort

Once a secret is established, (with or without the help of third party authentication mechanisms), multiple interfaces can lower the encryption effort during data transfers as illustrated next.

The "All-or-Nothing Transform" (AONT) was introduced by Rivest [61] to enhance the security of block cipher codes. It has the property that the entire cipher-text must be decrypted before one can determine even a single message block. An example cited in [61] states "if an 8 Megabyte file is encrypted in all-or-nothing CBC mode with a 40 bit DES key, the adversary must decrypt the entire 8 Megabyte file in order to test a single candidate 40-bit key". In [61], an efficient implementation of all-or-nothing transform is proposed as a pre-processing step to be followed by regular block-cipher encryption. (In other words, Rivest proposes two encryption steps: (i) AONT, which is followed by (ii) regular block-cipher encryption).

Multiple interfaces make it feasible to reduce the key-length used in block-cipher encryption (i.e., in the 2nd step of Rivest's 2-step encryption scheme) while maintaining the same level of security. In the best case scenario, the block-cipher encryption step can be skipped altogether, i.e., simply apply AONT to the message and send chunks along the

n interfaces.

Let the number of interfaces be n and the probability of compromising one interface/link be p_l. This implies that the expected number of attempts needed to compromise a link once is $(1/p_l)$. Assuming that compromising distinct interfaces can be approximated to be independent events,

$$\begin{aligned} P(\text{compromising } n \text{ interfaces}) &= p_l^n, \\ \text{Expected \# of attempts to break } n \text{ interfaces} &= \\ & (1/p_l^n) \qquad (2) \end{aligned}$$

As per [61], in general the work required to search for an unknown b-bit key to a known block-cipher is 2^b in the worst case and 2^{b-1} on the average. The total $\mathcal{E}$ effort required by the attacker can be approximated as

$$\begin{aligned} \mathcal{E} \approx\ & \text{Effort to compromise link(s)} \\ & + \mathcal{E}_{D-AONT}(\text{Effort to decrypt AONT}) \\ & + \text{Effort to break block-cipher} \qquad (3) \end{aligned}$$

Suppose that the availability of multiple interfaces allows us to get away with a block cipher with only b_r bit keys (in the second step that performs block-cipher coding) where $b_r < b$. We find b_r by maintaining the attacker's effort level:

$$\begin{aligned} \frac{1}{p_l} + \mathcal{E}_{D-AONT} + 2^{b-1} &= (\frac{1}{p_l})^n + \mathcal{E}_{D-AONT} + 2^{b_r-1} \\ & \Longrightarrow \\ \lg\left[\frac{1}{p_l} + 2^{b-1} - (\frac{1}{p_l})^n\right] + 1 &= b_r \qquad (4) \end{aligned}$$

If the quantity inside the square brackets is less than 1, then $b_r \leq 1$, i.e., the second step can be skipped. In other words if the condition

$$\left(\frac{1}{p_l}\right)^n - \frac{1}{p_l} \geq (2^{b-1} - 1) \qquad (5)$$

holds, then the second step can be skipped. Suppose $b = 128$ bits, $n = 4$ then the block cipher coding step can be skipped if the probability of compromising a link is $p_l \approx 2^{-32} \approx 10^{-9.63}$ or smaller. This probability could be considerably smaller than the probability of tapping a real link, especially a wireless one. The Extended-Multi-Homing (Section 2.1.1) can "amplify" the effect of multiple interfaces and makes it feasible to skip the second step in realistic scenarios.

3.2 Spread-Identity Pushes Man-in-the-Middle Attacker Toward the Connection End-Points

Suppose source $\mathcal{S}$ has n interfaces with k addresses per interface. Destination $\mathcal{D}$ has l interfaces with m addresses per interface. Then the number of distinct source-destination address pairs is $(k \cdot n \cdot l \cdot m)$. The number of distinct paths (i.e., at least one hop is different between every pair of paths) is $(n \cdot l)$ (from each physically distinct interface at the source to each distinct interface at the destination). In this scenario, if the attacker $\mathcal{M}$ is far away from both $\mathcal{S}$ and $\mathcal{D}$, it's job is harder for the following reasons:

[a] Attacker $\mathcal{M}$ would have to know which of the $(n \cdot l)$ paths are being used and whether multiple of those paths pass through single link. This is a much harder task, especially since *routers might dynamically switch paths depending on the load*. Inside the network, $\mathcal{M}$ would have to tap $(n \cdot l)$ paths in the worst case. In contrast, at the edges, $\mathcal{M}$ would have to tap only $\min(n, l)$ paths which requires a lot less effort.

[b] Even if there is a link where all the paths between $\mathcal{S}$ and $\mathcal{D}$ converge, if the attacker chooses to concentrate its effort on tapping that link, it has to figure out which addresses $\mathcal{S}$ is using and which addresses $\mathcal{D}$ is using. To find out $\mathcal{S}$'s or $\mathcal{D}$'s addresses via nameserver queries is a hard task in the proposed framework (as demonstrated above in Section 2)

This property could be useful: It implies that with Spread-Identity communications, efforts to detect and prevent man-in-the-middle attacks need to be focused at the end points of a connection where they will be most effective.

4 Conclusion

This paper investigated fundamental theoretical questions and issues related to security. It addresses fundamental issues like the security implications multiple network interfaces and addresses. It unifies several apparently diverse problems like DOS resilience and single-flow protection by revealing their solutions via the novel "Spread Identity" communications paradigm. Furthermore it demonstrates pathways to integrate Spread Identity/Security issues together with bandwidth aggregation, load balancing and other attributes into a unified, coherent "Dynamic Transport Selection" framework. The most significant impact could arise from cross-fertilization: i.e., the application of the "Spread Identity" principles to other areas like storage and archival systems, databases, and beyond.

References

[1] Denial of service (dos) attack resources. http://www.denialinfo.com/.

[2] Filer Deployment Strategies for Evolving LAN Topologies. http://www.netapp.com/tech_library/3009.html.

[3] Protocol Engineering Lab at University of Delaware CIS Dept. http://www.cis.udel.edu/ iyengar/research/SCTP.

[4] SCTP: An Overview. http://sctp.chicago.il.us/sctpoverview.html.

[5] SCTP for beginners. http://tdrwww.exp-math.uni-essen.de/pages/forschung/sctp_fb.

[6] SCTP site. www.sctp.org.

[7] SCTP site in Germany. www.sctp.de.

[8] Sun StorEdge Network Data Replicator White Paper. http://www.sun.com/storage/white-papers/sndr.html.

[9] H. Adiseshu, G. Parulkar, and G. Varghese. A Reliable and Scalable Striping Protocol. In *Proceedings of the ACM SIGCOMM*, pages 131–141, 1996.

[10] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Proc. 17th ACM SOSP, Kiawah Island, SC* , 1999.

[11] Akamai Inc. Fast internet content delivery with free flow, April 2000. http://www.cs.washington.edu/homes/ratul/akamai/freeflow.pdf.

[12] M. Allman, H. Kruse, and S. Ostermann. An Application-Level Solution to TCP's Satellite Inefficiencies. In *Proceedings of the First International Workshop on Satellite-based Information Services (WOSBIS)*, Rye, NY, November 1996.

[13] D. C. Anderson, J. S. Chase, and A. M. Vahdat. Interposed request routing for scalable network storage. In *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, October 2000.

[14] R. Atkinson. *Key Exchange Delegation Record for the DNS*, November 1997. RFC 2230.

[15] H. Balakrishnan, K. Lakshminarayanan, and S. R. et. al. A Layered Naming Architecture for the Internet. In *Proceedings of SIGCOMM'04, Portland, OR*, Sep. 2004.

[16] G. R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, volume 48, pages 242–268. American Federation of Information Processing Societies, 1979.

[17] E. C. Perkins. *IP Mobility Support*, October 1996. RFC 2002.

[18] I. Cidon, R. Rom, and Y. Shavim. Analysis of multi-path routing. *IEEE/ACM Transactions on Networking*, 7(6):885–896, Dec. 1999.

[19] Cisco Systems. http://www.cisco.com/warp/public/779/largeent/learn/ technologies/fast_echannel.html.

[20] M. Crawford and C. Huitema. *DNS Extensions to Support IPv6 Address Aggregation and Renumbering*, July 2000. RFC 2874.

[21] D. Dittrich. Distributed denial of service (ddos) attacks/tools. http://staff.washington.edu/dittrich/misc/ddos/.

[22] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, Jan. 1993.

[23] D. Eastlake. *Secure Domain Name System Dynamic Update*, April 1997. RFC 2137.

[24] D. Eastlake. *Storage of Diffie-Hellman Keys in the Domain Name System (DNS)*, March 1999. RFC 2539.

[25] D. Eastlake and O. Gudmundsson. *Storing Certificates in the Domain Name System (DNS)*, March 1999. RFC 2538.

[26] T. Goff and D. S. Phatak. Unified Transport Layer Support for Data Striping and Host Mobility. *IEEE Journal of Selected Areas in Communications*, 22, May 2004. to appear.

[27] D. Gu, G. Pei, M. Gerla, and X. Hong. Integrated Hierarchical Routing for Heterogeneous Multi-hop Networks. In *Proceedings of the IEEE MILCOM, Los Angeles, CA*, October 2000.

[28] R. Haagens. iSCSI Requirements. http://www.ietf.org/proceedings/00jul/SLIDES/ips-iscsi-reqs.pdf.

[29] T. Hacker, B. Athey, and B. Noble. The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network. In *Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS)*, April 2002.

[30] H.-Y. Hsieh and R. Sivakumar. A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts. In *Proceedings of the ACM MOBICOM*, pages 287–297, Sept. 2002.

[31] C. Huitema. Multi-homed TCP. Internet Draft, Internet Engineering Task Force, 1995. Expired.

[32] C. Huitema and R. Draves. Host-Centric IPv6 Multihoming. Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[33] J. Ioannidis. Fighting spam by encapsulating policy in email addresses. In *Proceedings of the Symposium of Network and Distributed Systems Security (NDSS)*, San Diego, California, Feb. 2003.

[34] V. Jacobson. Modified TCP Congestion Avoidance Algorithm. end2end interest group mailing list, April 1990.

[35] S. Kent and R. Atkinson. *IP Authentication Header*, November 1998. RFC 2402.

[36] Y. Ko and N. H. Vaidya. Location-Aided Routing (LAR) Mobile Ad Hoc Networks. In *Proceedings of MOBICOM'98, Dallas*, Oct. 1998.

[37] S. C. Kothari. Generalized linear threshold scheme. In *Advances in Cryptography—CRYPTO '84*, pages 231–241. Springer-Verlag, 1985.

[38] S.-J. Lee and M. Gerla. Dynamic Load-Aware Routing in Ad hoc Networks. In *Proceedings of ICC'01, Helsinki, Finland*, June 2001.

[39] S.-J. Lee and M. Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks . In *Proceedings of ICC'01, Helsinki, Finland*, June 2001.

[40] Liao, W-H., et. al. A Multi-Path QoS Routing Protocol in a Wireless Mobile Ad Hoc Network. In *Proceedings of the IEEE ICN'00, CREF, Colmar, France*, July 2001.

[41] L. Magalhaes and R. Kravets. Transport Level Mechanisms for Bandwidth Aggregation on Mobile Hosts. In *Proceedings of The 9th International Conference on Network Protocols (ICNP)*, 2001.

[42] R. Mahajan. How akamai works. http://www.cs.washington.edu/homes/ratul/akamai.html.

[43] G. Malkin. *Nortel Networks Multi-link Multi-node PPP Bundle Discovery Protocol*, September 1999. RFC 2701.

[44] D. A. Maltz and P. Bhagwat. MSOCKS: An Architecture for Transport Layer Mobility. In *Proceedings of the IEEE INFOCOM*, pages 1037–1047, San Francisco, California, March/April 1998.

[45] J. Mirkovic, J. Martin, and P. Reiher. A taxonomy of ddos attacks and ddos defense mechanisms. Technical report, UCLA CS Department, 2002.

[46] R. Moskowitz. Host Identity Payload, February 2001. IETF Draft http://homebase.htt-consult.com/draft-moskowitz-hip-arch-02.txt.

[47] J. Mysore and V. Bharghavan. A New Multicasting Based Architecture for Internet Host Mobility. In *Proceedings of the ACM MOBICOM 1997*, pages 161–172, Sep. 1997.

[48] A. Nasipuri and S. R. Das. On-Demand Multipath Routing for Ad-Hoc Networks. In *Proceedings of the IEEE ICCN,99, Boston*, pages 64–70, October 1999.

[49] G. Navalakha and N. Vaidya. Using Multi-path Source Routing to Improve Transport Layer Performance over Wireless Ad Hoc Networks, July 2002. personal communication.

[50] R. Ogier, V. Ruenburg, and N. Shacham. Distributed Algorithms for Computing Shortest Pairs of Disjoint Paths. *IEEE Transactions on Information Theory*, 39(3):443–455, Mar. 1993.

[51] E. P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. *Dynamic Updates in the Domain Name System (DNS UPDATE)*, April 1997. RFC 2136.

[52] M. Pearlman, Z. Haas, P. Scholander, and S. Tabrizi. On the Impact of Alternate Path Routing for Load Balancing in Mobile Ad Hoc Networks. In *Proceedings of ACM MobiHOC'00, Boston*, pages 119–128, August 2000.

[53] G. Pei and M. Gerla. Mobility Management for Hierarchical Wireless Networks. *ACM/Kluwer Mobile Networks and Applications*, 2000.

[54] C. Perkins. *IP Encapsulation within IP*, October 1996. RFC 2003.

[55] C. Perkins. *Minimal Encapsulation within IP*, October 1996. RFC 2004.

[56] D. Phatak, T. Goff, and V. Gupta. Leveraging extended-multi-homing for security by spreading identity. August 2004. UMBC CSEE Technical Report No. TR-CS-04-09.

[57] D. S. Phatak and T. Goff. A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments. In *Proceedings of the IEEE INFOCOM*, pages 773–781, Jun. 2002.

[58] D. S. Phatak, T. Goff, and J. Plusquellic. IP-in-IP Tunneling to Enable the Simultaneous Use of Multiple IP Interfaces for Network Level Connection Striping. *Computer Networks*, 43:787–804, Dec. 2003.

[59] J. Raju and J. Garcia-Luna-Aceves. A New Approach to On-Demand Multipath Routing. In *Proceedings of the IEEE ICCN,99, Boston*, pages 522–527, October 1999.

[60] D. Rand. *PPP Reliable Transmission*, July 1994. RFC 1663.

[61] R. L. Rivest. All-or-nothing encryption. In *Fast Software Encryption*, number 1267 in Springer Lecture Notes in Computer Science, pages 210–218, 1997.

[62] L. Rizzo. Dummynet: a Simple Approach to the Evaluation of Network Protocols. *ACM Computer Communication Review*, 27(1):31–41, Jan. 1997.

[63] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley, 1994.

[64] A. Shamir. How to share a secret. *Communications of the ACM*, 24(11):612–613, Nov. 1979.

[65] D. R. Simon and V. N. Padmanabhan. Secure traceroute to detect faulty or malicious routing, Oct. 24 2002. Proceedings of Hornets 2002.

[66] H. Sivakumar, S. Bailey, and R. L. Grossman. PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks. In *Proceedings of Supercomputing 2000*, pages 240–246, Dallas, TX, November 2000. IEEE.

[67] K. Sklower, B. Lloyd, G. McGregor, D. Carr, and T. Coradetti. *The PPP Multilink Protocol (MP)*, August 1996. RFC 1990.

[68] A. Snoeren. Adaptive Inverse Multiplexing for Wide-Area Wirelss Networks. In *IEEE Globecom*, pages 1665–1672, 1999.

[69] A. C. Snoeren and H. Balakrishnan. An End-to-End Approach to Host Mobility. In *Proceedings of the ACM MOBICOM, Boston*, Aug. 2000.

[70] J. Solomon. *Applicability Statement for IP Mobility Support*, October 1996. RFC 2005.

[71] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. *Stream Control Transmission Protocol*, October 2000. RFC 2960.

[72] Stewart, R. R., Ramalho, M. A. et. al. SCTP Extensions for Dynamic Reconfiguration of IP Addresses and Enforcement of Flow and Message Limits.
http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-addip-sctp-02.txt, June 2001.

[73] N. Taft-Plotkin, B. Bellur, and R. Ogier. Quality of Service Routing Using Maximally Disjoint Paths. In *Proceedings of the IEEE IWQoS'99, London, UK.*, pages 119–128, June 1999.

[74] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 2002.

[75] S. Tilak and N. Abu-Ghazaleh. A Concurrent Migration Extension to an End-to-End Host Mobility Architecture. *Mobile Computing and Communications Review*, 5(3):26–31, July 2001.

[76] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. *Layer Two Tunneling Protocol "L2TP"*, August 1999. RFC 2661.

[77] E. W. Simpson. *The Point-to-Point Protocol (PPP)*, July 1994. RFC 1661.

[78] E. W. Simpson. *PPP in HDLC-like Framing*, July 1994. RFC 1662.

[79] H. J. Wang, R. Katz, and J. Giese. Policy-Enabled Handoffs Across Heterogeneous Wireless Networks. In *Proceedings of WMCSA, New Orleans*, 1999.

[80] B. Wellington. *Secure Domain Name System (DNS) Dynamic Update*, November 2000. RFC 3007.