Scalable HAIPE Discovery

Glen Nakamoto
202 Burlington Road
Bedford, MA 01730
USA

nakamoto@mitre.org

ABSTRACT

This paper presents a scalable concept for the dynamic discovery of High Assurance Internet Protocol Encryption (HAIPE) devices situated across multiple "striped" network segments. The term "striped" in this context refers to traversing from a red (or classified) network to a black (or unclassified) network to a red network in a multiple concatenated manner (i.e., red-black-red-black-red ...). There are many reasons why network "segmentation" using IP encryption may occur: use of a commercial satellite link, traversing from one secure facility to another on an existing base network, operating over a radio frequency network, and so on. Each of these network segments or enclaves need to be secured (in this case, via IP encryption) which causes the segments to exist. The boundary between red and black sides is assumed to be protected via a HAIPE device (or an equivalent of an IPSEC virtual private network gateway). Our design also addresses mobile enclaves (where whole networks may come and go every 15 minutes) and multi-homed enclaves (where multiple entry/exit points exist). Finding how one traverses this striped environment and operate on a global scale (millions of network) are key challenges and the subject of this paper.

1.0 INTRODUCTION

With the recent advent of IP-based encryption and the introduction of High Assurance Internet Protocol Encryption (HAIPE), the transition to IP encrypted networks has begun in earnest. Legacy encryption using link level encryption operates below the IP layer and has no impact on issues such as peer discovery or routing. However, one pair of link encryptors is needed for every communication link and all communications and network equipment has to be managed by cleared personnel within appropriately protected facilities. Use of IP encryption permits the possibility of using commercial network services and significantly reduces the number as well as the total cost of these encryption devices. In addition, it also allows for statistically multiplexing different types of traffic at different classification levels, making better use of available bandwidth.

However, with those advantages, some new services are needed to regain network transparency in the presence of IP encryption. The first of these services (and the subject of this paper) is a peer discovery function that is scalable to global levels. The fundamental discovery problem is as follows: an IP packet is addressed to a destination host and is forwarded to an IP encryptor (for ensuring the confidentiality of the payload). This encryption is typically done in an encapsulation mode whereby the entire IP packet (including the true destination and source IP addresses) is also encrypted. The (sending) IP encryptor, however, needs to know the IP address of the corresponding IP encryptor that will decrypt this packet and forward the original packet to the true destination. Historically, this security association (SA) between the initiating and destination encryptor was manually created. As such, that approach does not scale well when the destination can be one of hundreds if not thousands of potential hosts. An automated discovery mechanism (i.e., scalable

Nakamoto, G. (2006) Scalable HAIPE Discovery. In *Visualising Network Information* (pp. 5-1 – 5-14). Meeting Proceedings RTO-MP-IST-063, Paper 5. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int/abstracts.asp.

HAIPE discovery) is needed to permit the initiating IP encryptor to find the appropriate "fronting" (or destination) IP encryptor in order to establish this security association with no manual intervention. In addition to discovery, it is also important that segmented (red) enclaves (as part of the "striped" environment) also know how to route packets to the ultimate destination within their own interior gateway protocol (IGP) domain. This is important since segmented red enclaves may be part of the same autonomous system (AS) but may be unable to participate in the IGP without deliberate and sometimes inefficient measures (such as generic route encapsulation tunneling). There are many reasons why "segmentation" may occur: use of a commercial satellite link, traversing from one secure facility to another on a base network, operating over a radio frequency network, and so on. Each of these network segments need to be secured (in this case, via IP encryption, which causes the segments to exist). The term "striped" in this context refers to traversing from a red (or classified) network to a black (or unclassified) network to a red network in a multiple concatenated fashion (i.e., red-black-red-black-red ...). The boundary between red and black sides is assumed to be protected via a HAIPE device or from a commercial sector viewpoint, a device that functions as an IPSEC virtual private network (VPN) gateway. As we transition from a single (link level encrypted) red network (our current legacy environment) to a potentially segmented (IP level encrypted) striped network, the peer discovery problem becomes complex.

2.0 SCALABLE DISCOVERY IN A SINGLE BLACK CLOUD

The discovery mechanism will be first addressed by examining the discovery process through a single black cloud and extended in the following sections to striped environments. Version 3 of the High Assurance Internet Protocol Encryption Interoperability Specification (HAIPE IS) is used as a foundation capability throughout this paper. In v3, a Routing Information Protocol (RIPv2 or RIPng) capability has been added to permit the HAIPE device to "discover" the networks that it protects (passive RIP participation – a local peer enclave discovery process). This alleviates the need to manually enter the network prefixes protected by this HAIPE device. Upon retrieving these prefixes, the HAIPE device can **register** (another v3 function) with a Local Peer Discovery Server (LPDS) of which an experimental prototype has been developed at MITRE. Each LPD server can "manage" approximately 10,000 networks (and possibly more). A HAIPE device could be fronting for one workstation or 200+ prefixes (the number of networks typically in a medium size military base). The sizing for the LPD server is a function of the quantity of prefix/HAIPE associations, the frequency of queries being serviced (i.e., rate of queries – one per new SA), and frequency of change (prefix additions or losses as a result of changes or unit mobility).

After the registration process, the system is ready to support discovery queries. *[Note: In practice, a given LPD server could be handling some form of registration almost constantly due to changes that may be taking place within its domain. See section on mobility for further details.]* Referring to figure 1, when a packet arrives at the plaintext (PT) side of the HAIPE for which there is no existing security association (SA), the HAIPE can first examine its cache to determine if prefix/HAIPE information for that destination exists. If so, it can use it (within its time-to-live duration) and no queries are needed. If the information is not there, v3 HAIPE supports a generic client query (which we have emulated in our lab) for requesting discovery services for the destination (or, in the case of striped networks, the "next hop") HAIPE IP address. This query is directed at the LPD server which is generally "geographically near" (at least from a network or administrative viewpoint). As shown in figure 1, the HAIPEs will use RIP to discover their networks and will have previously registered with the LPD server (both planned v3 capabilities). The HAIPE client query is sent to the LPD server to find the CT IP address of the destination HAIPE (which was entered into the LPD server through a previous registration process). The LPD server responds with the requested prefix/HAIPE information to the querying HAIPE. With this information, the source HAIPE can establish an SA with the destination HAIPE and traffic can then flow across this encrypted tunnel.

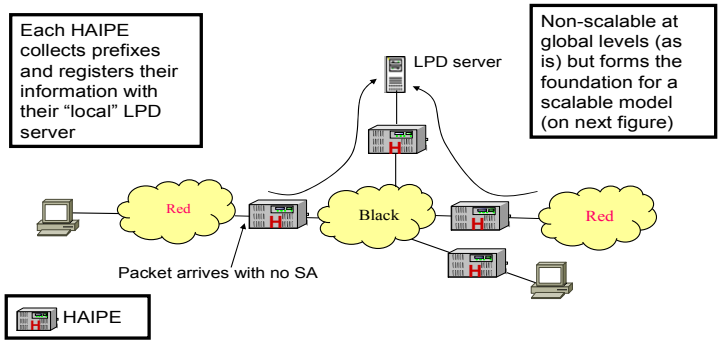


Figure 1: Basic Local Peer Discovery

This model works fine until the data base on the LPD server begins to get too large or the rate of change happens at a pace that cannot be maintained. To solve that issue, a service concept similar to the Domain Name Service (DNS) is used for building scalability. Figure 2 illustrates a potential three tier hierarchy that is estimated to handle one to ten million networks/prefixes.

The LPD server, as indicated earlier, supports a group of HAIPE devices collectively fronting as many as 10,000 networks. There are no assumptions with regards to address ranges being serviced by a given LPD server, i.e., no address summarization is required at the HAIPE (enclave) level. Each LPD server will, in turn, register its set of prefixes to one or more Referral Servers (similar to the .com or .org level in DNS terms). These Referral servers (RS) will be responsible for some percentage (say 10% to 20%) of the total IP address space (implying 5 to 10 servers total on a global scale not counting redundancy or design robustness). The allocation of prefixes to server can either be manually managed based on performance monitoring or dynamically load balanced. This is an area of investigation that is still continuing. A single Referral server (not counting backups or mirror sites) holds peer discovery information for a given prefix (organized along prefix groupings). The HAIPE devices keep sync with their LPD server which, in turn, keeps sync with their Referral servers. In this respect, the model more closely resembles reverse DNS rather than regular DNS. Each LPD server will register with its corresponding Referral server the fact that "it knows about" a given IP address range (prefix). The LPD server also summarizes this information (where possible) to further minimize the volume of data being sent. The actual number (of Referral servers needed) would be a function of administrative control and performance (which is the same for LPD servers as well). A single Root server would keep track of all Referral Servers and permit dynamic management as well as other potential discovery services. We will also be exploring dynamic load balancing concepts to react to potentially rapidly changing conditions. The server quantities do not take into account redundancy and mirroring which are essential concepts for robustness. Again, the DNS model provides solid lessons learned on how to robustly establish such a service.

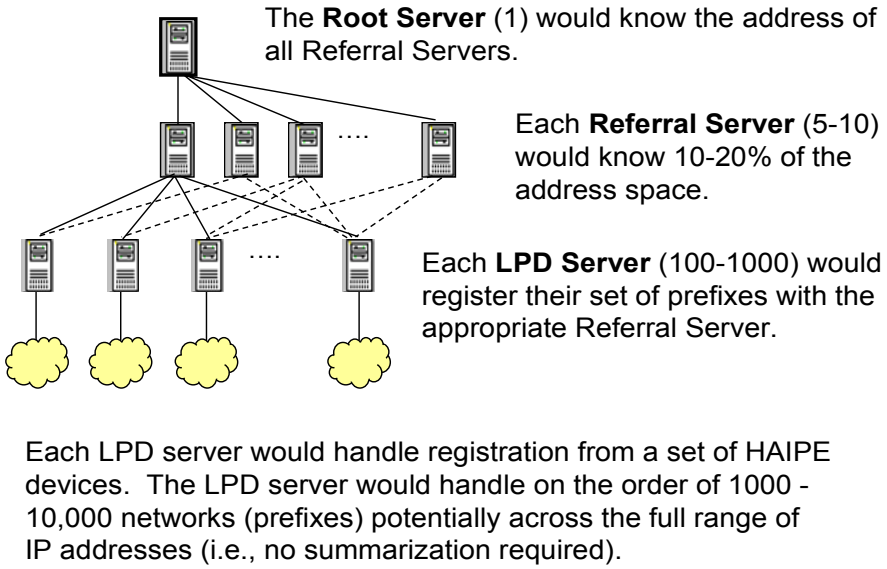


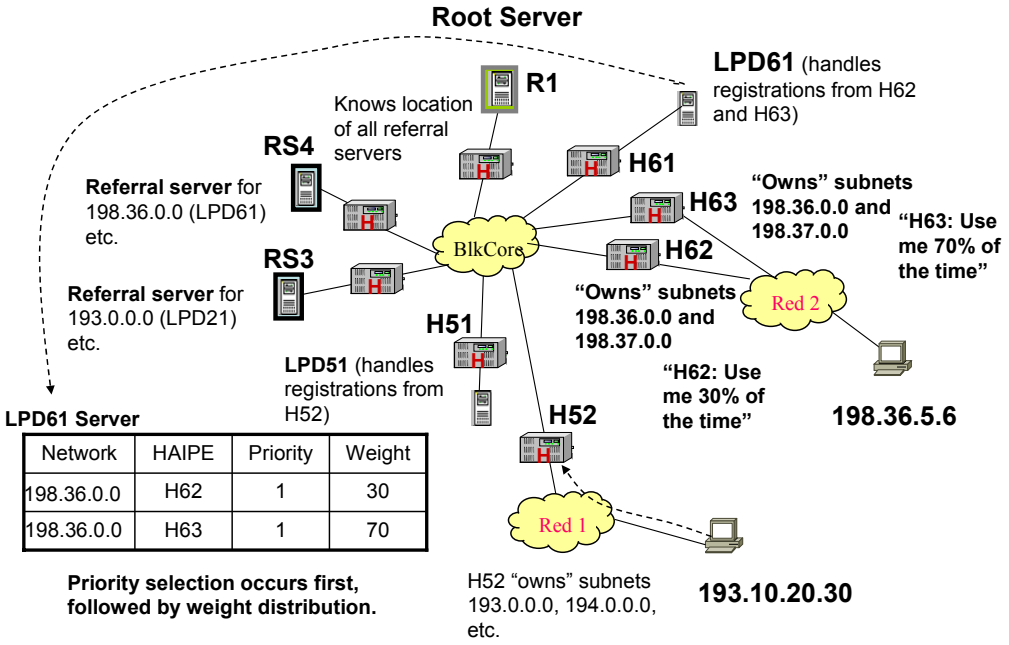
Figure 2. Creating the scalable hierarchy

A "hints" file (again DNS model) at the LPD server level can contain the address information for the Root server as well as all the Referral servers. Going directly to the Referral server eliminates the query to the Root server. If the "hints" file is outdated or wrong, the Root server can still be queried for the updated information. While at first, the Root server seems unnecessary, it can provide a great deal of future flexibility with regards to dynamic discovery services beyond HAIPE including multicast source discovery (a subject of another paper addressing Protocol Independent Multicast/Source Specific Multicast [PIM-SSM] in the striped environment) or capturing community of interest (COI) virtual boundary protection policy information (for determining security requirements for traversing different COIs). If an LPD server can handle 10,000 networks/prefixes (see database sizing discussion), only 100 servers would be needed globally (to support one million prefixes), significantly reducing the overall implementation cost. The real issue is not so much the number of networks serviced (per server) but the response time of the service offered. For a given caliber of machine, the same number of prefix support can still yield radically different performance depending on activity level. It may be conceivable to have one LPD server with 10,000 prefixes servicing slowly changing environments while another LPD service may only have 1000 prefixes due to rapid changes in a mobile deployment situation. Having a dynamic hierarchical design permits rapid changes to prefix allocations to support exercises, deployments, and other activities that require potentially significant changes. Each prefix or HAIPE (or both) could have a time-to-live (TTL) or a cache timeout which forces a periodic refresh. The refresh could be a "keep alive" signal indicating no change, a delta change from "last time," or a full update. The frequency and type of refresh can be customized (where needed) or defaulted (for simplicity) for each site/mobile unit.

All key servers should be redundantly set up (similar to DNS set ups) for high availability. In addition, one can also mirror these services across different geographic regions to reduce the network "travel distance" where possible and yet provide another level of redundancy. With a small number of systems (say 3-4), database synchronization techniques work reasonably well, provide continuity of operations, and improve performance. Because all queries are simple and "atomic" (no dependencies), no state is maintained and the servers can easily operate in a highly parallel fashion (again similar to DNS operations). Update race conditions (for which we know of none) should not have any impact to a user given the query behaviour envisioned for this service.

3.0 MULTI-HOMING

Figure 3 illustrates how support for multi-homed enclaves is done. In examining figure 3 in more detail, we can see a packet from a workstation with the IP address of 193.10.20.30 addressed to 198.36.5.6 arriving at H52. On the assumption that no SA exists for this pairing, H52 queries LPD51 (its local LPD server) for the fronting HAIPE (of the destination). LPD51 has no cache and uses its Hints File to find that RS4 holds information for the 198.36.0.0 prefix. LPD51 submits a query to RS4 and obtains information that LPD61 is the authoritative source for that prefix. RS4 also provides the information necessary to access LPD61. LPD51 queries LPD61 for fronting HAIPE data and is returned two paths (H62 and H63) with associated priorities and weights. This information is then formatted in the generic client response (by LPD51) and returned to the original requesting HAIPE (H52). With this information, the SA is established from H52 to either H63 or H62 (most likely H63 based on the initial set of weights shown). The LPD servers could optionally "offer" the (load balanced) "solution" (i.e., use H63 to 70% of the queriers) and based on the granularity of the SA, provide a potentially improved methodology for load balancing. However, providing "all" solutions (to the HAIPE) in response to a query eliminates the need for a subsequent query if the first "solution" does not work.



Network	HAIPE	Priority	Weight
198.36.0.0	H62	1	30
198.36.0.0	H63	1	70

Figure 3. Support for multi-homing

The concept for priority and weights as it applies to multi-homing is also derived from the DNS service (SRV) record. Priority is first chosen followed by weights within a given priority. In the previous example, both paths had the same priority but different weights, implying load balancing the traffic (actually the SAs). In another case, if one path had priority of 1 (high) while the other path was priority 2 (lower than 1), all traffic should be directed to the priority 1 path unless it is down or inaccessible. An optional feature (requires a slight change to v3 HAIPE IS) is the ability to selectively "flush" a portion of the (HAIPE) cache in the event a topology change occurs that permits potentially better network connectivity. If an enclave or terminal has two possible "connection paths" and one is preferred due to higher bandwidth (as an example), a situation can be created when the high bandwidth (preferred HAIPE) path goes down for some reason and then later recovers. In that situation, the less preferred path is selected (when the preferred path "goes down") in the discovery process either based on SA failure (which can take minutes depending on how the HAIPE is set up) or some other feedback mechanism (still under investigation) to detect a true link failure. Under that condition, the source HAIPE can initiate a discovery process or use any locally cached information to establish the alternate (less preferred) path. Once established, that SA will stay up for the configured duration (typically 24 hours). [*Note: One can configure the "less preferred" path with a shorter SA time as an alternative. However, this will create additional overhead.*] The problem that occurs is that when the "preferred" path recovers, the old SA does not disappear and will stay up until the time expires (or the link fails) and only new SA connections will be able to use the "preferred" path. In a tactical environment, this failure may happen with some frequency and the ability to quickly respond is critical. At the same time, we wish to customize the configuration appropriate for the particular environment without having "ripple" or route flapping effects (i.e., race conditions, hysteresis). One aspect of our design permits the updated

information (discovered locally through IGP interaction) to update the LPD server (as part of its normal process). The LPD server can (optionally) maintain a list of "recent" (within past 24 hours) LPD servers that have requested discovery information regarding the impacted (multi-homed) networks and notify that set that updated information is available. The affected LPD servers that have potential HAIPE devices that support that SA (destination network) can send a "flush cache" message (targeted for the specific prefix involved – for example, a specially formatted Solicitation Response message) to the HAIPE which can optionally listen to it or ignore it (and let the SA expire naturally). This would require an addition of a function on HAIPE to receive a UDP message with a prefix notification to refresh discovery for that prefix (or the use of an existing message formatted for this function). Upon receiving this message, the HAIPE can specifically end any SAs associated with that prefix and force the discovery of the "preferred" path or alternatively, redirect the flow of traffic to the preferred path while maintaining the backup SA (if possible). This analysis is still on-going and further experimentation is envisioned before arriving at a final recommended concept. Our goal is to have the transition to the new SA to be transparent to the application with little or no dropped packets.

4.0 MOBILITY

In figure 4, one of three scenarios is illustrated for mobility support. In this situation, the red mobile unit "moves" and needs to re-associate its HAIPE with a different CT (black) IP address as well as a new LPD server with the associated update process to the Referral servers (which are the same servers given the PT prefixes have not changed). Upon obtaining its new CT address (24.20.50.20) and information about its new LPD server, the HAIPE registers with the new LPD server (LPD2032). LPD2032 sends updates to Referral servers RS10 and RS20 that it (LPD2032) "knows about" prefixes 1-150 in this example. The update process takes a couple of seconds at most and could also (optionally) involve a verification process before the update takes place. RS10, when receiving an update from LPD2032 to add prefixes 1-150 to 24.20.50.20, could initiate a query to the original owner (LPD1005). Upon confirming that the prefix is no longer "there" or there are no responses coming back, RS10 can perform the update process. A logging process would also permit a full audit of changes for management oversight.

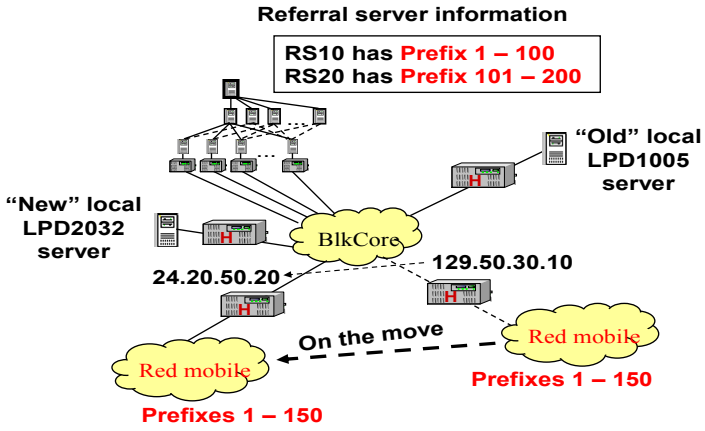


Figure 4. Support for mobility

Additionally, it is possible that prefix information can be pre-staged (at the new location, if known) for potential deployment scenarios (ships moving into a theatre). The data could be sent ahead of time (pre-loaded into the "new" LPD server) and activated at the right time (thus, eliminating much of the data transfer time on transition). Once activated, an update process can adjust any changes that may have taken place from initial load to activation. A slight modification to dynamic host configuration protocol (DHCP) could permit the automatic assignment of CT side addresses for the HAIPE device. Since the HAIPE does not typically need DNS services (on the CT side), it could use the DNS address information field (contained in the DHCP configuration database) as the LPD server address. If the LPD address information is considered sensitive but unclassified, the DHCP process could undergo a certificate-based authentication "handshake" which could also form the foundation for providing a shared secret for encrypting the LPD server address. The IP address of this new LPD server can be entered manually or HAIPE could be modified (v3.x) to potentially read this address (from DHCP service) and be automatically set up to do the re-registration upon network connection with little or no user intervention (depends on trust level and concept of operation). We have also addressed two other cases: 1) dealing with mobile units that "own" their LPD server and moves with it and 2) mobile units that change their red prefixes (adds or deletes) with no associated CT IP address change.

5.0 DATABASE SIZING

As previously stated, an LPD server with 10,000 prefixes would only create a 1 MB data base resulting in a global server count of 100 (to support one million networks). This sizing estimate is based on a message size of 51 bytes of prefix and HAIPE information (using v3 generic client message format data) with 49 bytes of future or reserved fields – totalling 100 bytes for a transaction record. If an LPD server has 10,000 prefix entries each with 100 bytes of data per prefix entry, the resulting size is 1 MB (not counting any cache management). One hundred LPD servers (each with a 1 MB database on average), registering with up to 10 Referral servers, will result in a given Referral server with a 10 MB database (for 1 million prefixes) which is not a very large database. However, the size of the database is probably not the big issue vice the performance load of the server. Since the servers can be parallelized, the solution is straightforward. If operated like a DNS service, these databases are fully memory resident and provides further improved performance. The actual number of prefixes supported for a given LPD server can be dynamically established as a function of performance need. Prefix re-allocation can be done rapidly with changes being advertised at the Root or Referral server on a global basis.

Within our network lab spaces, we have developed two LPD servers, three Referral servers, and one Root server for performance and scalability testing. Our preliminary test results indicate that a "reasonable" size server can handle approximately 10,000 queries per second across the full range of IP addresses (no summarization constraint). A "query" in this context includes a v3 client query to the LPD server, the LPD server query to the Root server, the LPD server query to the Referral server, and finally the response to the HAIPE client with the SA establishment information. This would be the worst case scenario for the number of queries necessary to find an arbitrary destination anywhere in the world.

6.0 SCALABLE DISCOVERY IN A STRIPED NETWORK ENVIRONMENT

Two key assumptions are made with regards to operating within a striped network environment. These assumptions are illustrated in figure 5.

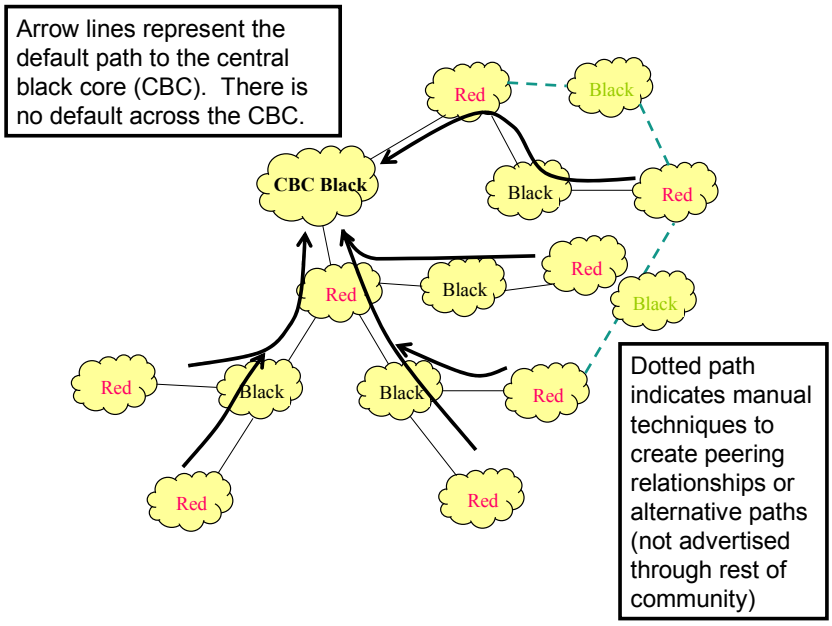


Figure 5. Key assumptions for striped network discovery

The first assumption is that all "black" clouds are hierarchically organized with the central black core (CBC) at the "root." The CBC is the "black core" that would, in theory (based on the "Black Core" vision), grow over time (subsuming the other black clouds). The second assumption is that for subjugated "red" networks (red networks not directly connected to the CBC), there is always a default path toward the central black core (but not across it).

To describe how discovery works through this striped environment, a two step process, first of registration then traffic flow, will be described. Figure 6 shows the network topology and illustrates the registration process that a HAIPE will do as part of the discovery process.

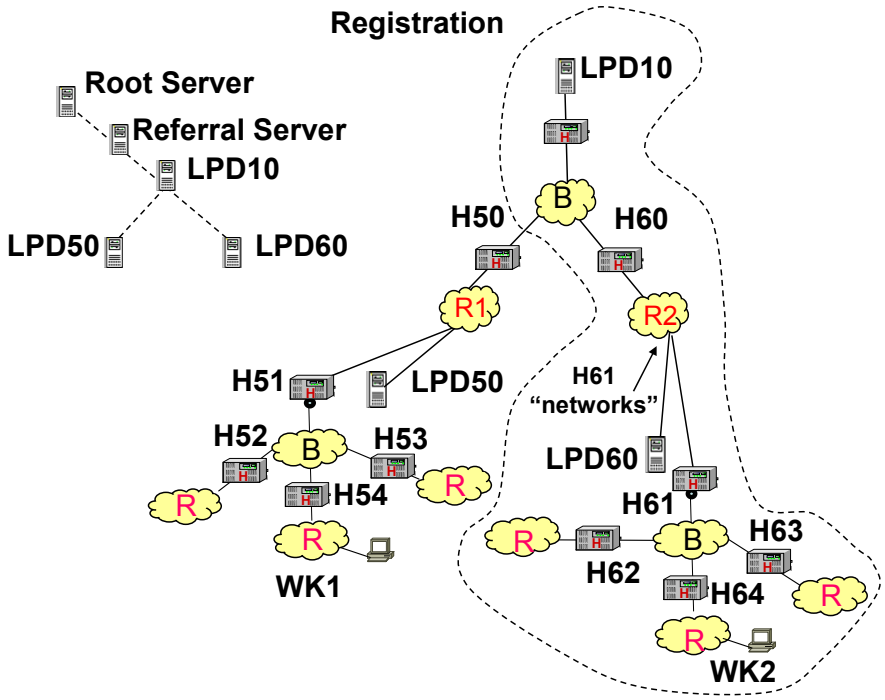


Figure 6. Support for striped networks – registration

H62, **H63**, and **H64** collect IGP information from their local red network (via RIP). Each of these HAIPEs "registers" with their LPD server (**LPD60)** which summarizes data where possible. **LPD60** "injects" (via IGP) route reachability data (for the enclaves protected by H62, H63, and H64) via the router port connected to **H61** (this information includes fact that WK2 network "sits" behind H61 interface, for example). **H60** collects IGP information from the R2 network including the **LPD60** injected routes (i.e., the three networks fronted by H62, H63, and H64). **H60** registers with **LPD10** all prefixes (including the "striped" segments). Any network change causes a "ripple" effect to the IGP and can trigger an update registration process (administratively regulated to prevent router flap propagation while supporting rapid mobility updates). **H50** goes through a similar registration process (ending up at LPD10). Note that for this "striped" network, **LPD10** is the LPD server layer (bottom tier) in the scalable hierarchy (LPD50 and LPD60 are subordinate to it) and registers its information with the Referral servers. Note that **LPD50** and **LPD60** do not participate further in the registration process other than receiving the registration data from devices under its purview. Its registration knowledge is "route injected" (unique to the striped configuration) and is passed on to the HAIPE that is adjacent to the central black core. That HAIPE, in turn, registers with **LPD10**.

Once the registration process is done, the traffic flows as illustrated in figure 7.

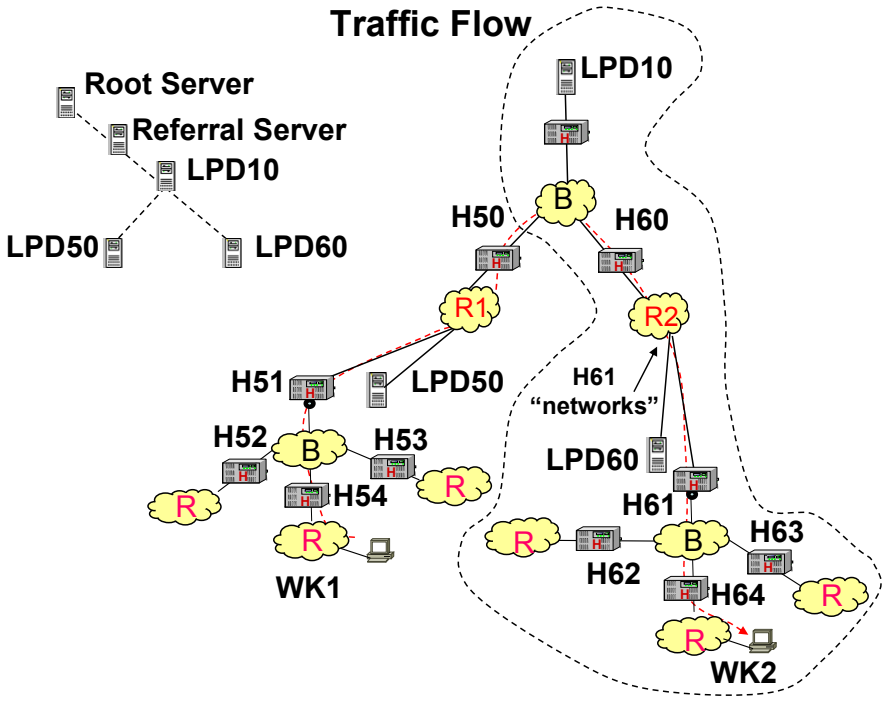


Figure 7. Support for striped network – traffic flow

A packet from **WK1** addressed to **WK2** arrives at **H54** (default route to H54 in local network). **H54** queries **LPD50** for "next hop" HAIPE info. **LPD50** is configured to support default path toward **H50** (via H51). **LPD50** returns **H51** address (SA between H54 and H51 established and data is passed). Data is decrypted at **H51** and put on R1 network which is default routed to **H50**. **H50** queries **LPD10** for "next hop" HAIPE info. No defaults are permitted. If unknown, the LPD server sends a FAIL message. **LPD10** returns **H60** address (SA between H50 and H60 is established and data is passed). Data is decrypted at **H60** and put on **R2** network. The default path (going toward the black core) is not applicable in this "direction." The packet is routed within the **R2** network toward interface connected to **H61** (which, during "registration," advertised to R2, the route path to the **WK2** subnet). **H61 queries LPD60** for HAIPE info (to get to WK2). **LPD60** returns **H64** address (SA between H61 and H64 is established and data is passed). Data is decrypted at **H64** and put on local network where standard IGP routes packet to **WK2.**

Note that the "**H61** queries LPD**60** for HAIPE info" step requires a change to the HAIPE (v3) client query to "go out" the PT side instead of the typical CT side for the LPD server information query. The query would be identical in format and function and appropriate safeguards would be needed to assure that no one can spoof or otherwise invalidate the LPD function (from the PT side). In order to alleviate the need for key management of special authentication techniques for the LPD server on the PT side, two techniques are being studied to permit the LPD server to securely communicate with its local HAIPE device. One technique

involves adding another HAIPE "in front of" the LPD server to permit query access via the CT side and using the second port (facing the PT network) only for "route injection" (and no LPD query/response functions on that interface). The second technique uses the existing topology design (no additional HAIPE needed) but places restrictions on access to the LPD from the local PT environment such as using non-routable multicast protocols (with TTL=1) for communications between the LPD and HAIPE (thus ensuring communications stays on the local area network and other machines have no access for spoofing opportunities). The existing multicast discovery protocol (with TTL=1) could be a candidate protocol to run on the PT side.

Utilizing a "red gateway" concept, we can also insert performance enhancing proxies (PEP) devices in key locations ("in front of" geosynchronous SATCOM, etc.) to support long latency communication links. This could be used in conjunction with multi-homing (alternate path) with a type-of-service (TOS) flag (or DSCP) for low latency service. For example, an incoming (arriving PT side to HAIPE) packet could have a "low latency" TOS bit set. When the peer discovery query returns multiple paths with one identified as PEP enabled (a modification to v3 HAIPE IS solicitation response), the HAIPE could select that path. The ability to support striped networks is the key enabling capability that permits support for these "services." The same "red gateway" concept permits the creation of communities of interest (COI) that support a consistent security policy of interconnected enclaves via a virtual (HAIPE encrypted) intranet. This intranet can connect to the "black core" via a red gateway and provide classic perimeter protection (network based intrusion detection, content/packet analysis, two-way proxy support, network monitoring for worms and other aberrant behavior). Even with this "intranet" concept, the peer discovery works to find any destination host regardless of its location (even when separated by multiple "black cloud hops"). In a typical intranet environment, systems outside the intranet would not be able to access systems inside the intranet due to lack of (IP address) location visibility. The peer discovery mechanism supporting the striped environment enables the discovery process to function without advertising any addresses on the black core and could provide a (red side) network address translation-like function (for the intranet) if security policy requires. We can also support the need for anonymization if needed. The discovery process can offer an anonymization service that can support a policy that two CT HAIPE addresses cannot "associate" with each other (given the CT address may be "linked" to a given organization). An anonymization flag (triggered, again, by TOS or DSCP or source/destination pairing, or possibly a metric in the registration record) can be used to further "down select" multi-home options indicating an anonymization path (or service) exists.

Another key factor in our design is the lack of a transition phase to go from striped to single black core. The design supports full interoperation of striped with directly connected workstations (with embedded HAIPE network interface cards). With the use of red gateways and route injection, we can interoperate with other proposed discovery mechanisms such as the previously proposed implicit discovery (IM-PEPD) approach.

While outside the scope of this paper, this project is exploring other "black core" issues such as implementing QoS without "advertising" DiffServ values, creating and using traffic engineered "paths" that meet minimum protection attributes (quality of protection), anonymous routing for hiding source and destination IP addresses of HAIPE devices, and security related mitigation concepts.

7.0 CONCLUSION

In conclusion, we believe our approach provides a realistic way to use planned (version 3) HAIPE devices in a true networked environment while addressing many of the issues that IP encryption introduces. We see no interoperability issues with existing management capability. From a performance viewpoint, our design exhibits enhanced DNS-like behaviour with three queries (maximum) for initial conditions (estimated time <500 ms) for a global access model (currently being confirmed in lab testing and simulation analysis). Conventional caching techniques makes discovery virtually transparent (except at start up). From a scalability

viewpoint, the design is globally scalable to many millions of networks. The hierarchical architecture permits growth from one to three tiers in a top down or bottom up fashion (and permits "disjointed" evolution similar to "real world" DNS implementation). Our approach keeps HAIPE and LPD/Referral/Root server design simple, extensible, and scalable. The LPD, Referral, and Root servers have been prototyped at MITRE and are currently undergoing further testing, and scalability analysis**.** The simplicity of the design is highly survivable due to its distributed architecture (based on lessons learned from DNS implementation). Finally, the design takes special consideration for mobility and permits rapid updates to take place on a global level.