Md.Zaheer Abbas, Dr.JVR Murthy / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 3, May-Jun 2012, pp.1347-1352

Authenticated And Policy - Compliant Source Routing

Md.Zaheer Abbas[*], Dr.JVR Murthy[**]
** professor, Dept. of Computer Science Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh.
[*]M.tech-CSE,JNTU,Kakinada , A.P. India

Abstract: - Routing is a black art in today's Internet. End users and ISPs alike have little control over how their packets are handled outside of their networks, stemming in part from limitations of the current wide-area routing protocol, BGP. We believe that many of these constraints are due to policy-based restrictions on route exportation. Separating forwarding policy from route discovery would allow users to select among the possibly many inter-AS paths available to them and enable ISPs to more effectively manage the end-to-end behavior of their customers' traffic.As a concrete mechanism for enforcing forwarding policy,we propose the concept of a network capability that binds together a path request, an accountable resource principal, and an authorizing agent. Network capabilities are central to Platypus, a loose source routing protocol we are designing,which composes network capabilities authorized by multiple ISPs to construct alternative inter-AS routes that can be independently validated and accounted for on the fly.

Keywords: Authentication, BGP, Overlay Networks, Routing,Waypoint.

1. Introduction:

Network operators and academic researchers alike recognize that today's wide-area Internet routing does not realize the full potential of the existing network infrastructure in terms of performance[1], reliability [2], [3], [4], or flexibility [5], [6], [7]. While a number of techniques for intelligent,source-controlled path selection have been proposed to improve end-to-end performance [8], [9], reliability [2], [3],[4], [10], and flexibility [11], [12], [6], [13], [7], they have proven problematic to deploy due to concerns about security and network instability. We attempt to address these issues in developing a scalable, authenticated, policy-compliant, wide-area source routing protocol.We argue that many of the deficiencies of today's routing infrastructure are symptoms of the coupling of routing policy and routing mechanism [14]. In particular, today's primary widearea routing protocol, the Border Gateway Protocol (BGP) is extraordinarily difficult to describe, analyze, or manage, Autonomous systems (ASes) express their local routing policy during BGP outeadvertisement by affecting the routes that are chosen and exported to neighbors. Similarly, ASes often adjust a number of attributes on routes they accept from their neighbours according to local guidelines [15], [16], [12]. As a result, configuring BGP becomes an overly complex task, one for which the outcome is rarely certain. BGP's complexity affects Internet Service Providers (ISPs) and end users alike; ISPs struggle to understand and configure their networks while end users are left to wonder why end-to-end connectivity is so poor. Our approach to reducing this complexity is to separate the issues of connectivity discovery and path selection. Removing policy constraints from route discovery presents an opportunity for end users and edge networks: routes previously hidden by overly conservative policy filters can be revealed by ASes and traversed by packets. The key challenge becomes determining whether a particular source route is appropriate. ASes have no incentive to forward arbitrary traffic; currently they only wish to forward traffic for their customers or peers. We argue, however, that this is simply a poor approximation of the real goal: ASes want to forward traffic only if they are compensated for it.

Henceforth, we will consider traffic policy compliant at a particular point in the network if the AS can identify the appropriate party to bill, and that party has been authorized by the AS to use the portion of the network in question. We present the design and evaluation of Platypus, a source routing system that, like many source-routing protocols before it, can be used to implement efficient overlay forwarding, select among multiple ingress/egress routers, provide virtual AS multi-homing, and address many other common routing deficiencies [14]. The key advantage of Platypus is its ability to ensure policy compliance during packet forwarding. Platypus enables packets to be stamped at the source as being policy compliant, reducing policy enforcement to stamp verification. Hence, Platypus allows for management of routing policy independent of route export and path selection. Platypus uses network

capabilities, primitives that are placed within individual packets, to securely attest to the policy compliance of source routing requests. Network capabilities are

i) transferable: an entity can delegate capabilities to others,

ii) composable: a packet may be accompanied by a set of capabilities,and

iii) cryptographically authenticated.

Capabilities can be issued by ASes to any parties they know how to bill. Each capability specifies a desired transit point (called a *waypoint*), a resource principal responsible for the traffic, and a stamp of authorization. By presenting a capability along with a routing request, end users and ISPs express their willingness to be held accountable for the traffic, and the included authorization ensures the policy compliance of the request. In addition to its basic design, we also aim to understand how Platypus might be deployed in today's Internet.

2. The state of BGP:

To motivate the need for separating policy enforcement from route advertisement and selection, we enumerate several efficiencies of BGP and describe how the coupling of policy and mechanism either creates or exacerbates the problem. Poor reliability/stability**.** BGP is a notoriously difficult protocol to configure properly [11]. We believe a significant portion of this complexity stems from the need to simultaneously optimize for route efficiency and policy compliance.Part of the problem is that it is very hard to know beforehand what the right configuration might be [18] because policy goals cannot be directly mapped to configuration settings; instead, operators must adjust a number of overloaded parameter values (MED, LOCAL PREF, and COMMUNITY being three of the most prominent) in hopes of coercing both their own internal network and adjacent ASes to select the desired routes. In fact, it's possible for local policy settings to guarantee that the routing configuration will diverge [19]. This issue could largely be avoided if ASes could simply export all possible routes, and determine whether or forward a particular packet (because it did or did not meet the AS's local policy constraints) when it arrived at a border router. We believe that a mechanism for explicit routing can free ASes to fully export routes as routing policy can be decoupled from route computation and distribution.

3. Sample applications :

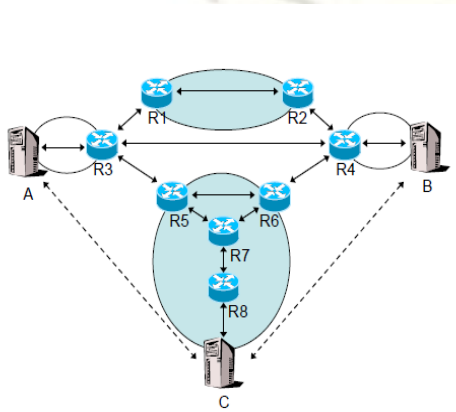

Figure 1: A simple network topology. Hosts A, B, and C all have different ISPs.

An *Platypus* architecture for loose source routing. Platypus allows end hosts to construct arbitrary paths through the network using the network infrastructure itself and allows ISPs to implement sophisticated routing policies. While Platypus is general in principle, we envision it will be used primarily for AS-level source routing, as we expect few ASes would allow intra-AS source routing. Critical to our design is the integration of authorization and Forwarding. A capability explicitly confers the necessary rights for a packet's source routing to be honoured. In addition, Platypus makes it easy for ASes to track the usage of capabilities. We provide a few examples of how capabilities could be used to address common or interesting routing problems below.

3.1 Overlay Construction

Nodes A, B, and C are all willing to transit traffic for each other in an overlay fashion. Let us assume for the moment that A and B ish to exchange traffic, but the default route A↔R3↔R4 ↔ B is unsatisfactory—perhaps because the link R3↔R4 is congested or down. Using existing overlay technologies, A and B can use C as a transit point by tunneling their traffic directly to C. While effective at avoiding the misbehaving link, this route is clearly sub-optimal for all involved. In particular:

1. C is forced to forward each packet itself,
consuming both last-hop bandwidth (in both directions) and processor resources. It would prefer that R8 forward the traffic instead.
2. Any path A ↔ R3 ↔ R5 ↔ R7 : : :R7 ↔ R6 ↔ R4 ↔ B is also sub-optimal from the point of view of both A and B—they would likely prefer shorter, equivalent routes like A ↔ R3 ↔ R5 ↔ R6 ↔ R4 ↔ B.
3. The ISP owning R5;R6;R7 and R8 (and the links between them) would likely prefer not to transit the traffic even to R7 unnecessarily.
4. If avoiding R3 ↔ R4 is the objective, an alternate route exists: A↔ R3 ↔ R1 ↔ R2 ↔ R4 ↔ B. In the case where C's ISP also owns R1 and R2, C should be able to authorize use of the link R1 ↔ R2.

The first issue can be addressed if node C were able to request its upstream router to redirect packets from A to B,such as with the recently proposed *reflection* primitive [21];C could ask R8 to *reflect* packets from A to B. Unfortunately, C's ISP's now cannot implicitly limit C's bandwidth use by restricting C's last hop. C's ISP is now liable for transiting a potentially large amount of traffic and needs some way to account for this usage. The ISP would likely want to rate limit the flow at the router using a token-bucket type scheme. The second issue can be only partially addressed using the reflection primitive recursively. If R8 propagates the reflection up to R7, the perceived path from A to B no longer traverses R8 or C, but this form of path relaxation cannot avoid R7, since R7

is likely unaware that a better path exists between R5 and R6. In Platypus, however, C could provide A with a capability allowing it to source route through C's ISP—naming C as the resource principal and C's ISP could intelligently route A's packets, addressing issues three and four.

3.2Routing Accountability

Today's routing infrastructure depends a large part on the good behavior of ASes and the correct configuration of BGP. BGP makes it easy for malicious speakers to falsely announce routes for prefixes they do not own. Future traffic to and from a hijacked prefix cannot be easily differentiated from valid traffic by third parties. However, networks in possession of capabilities to affect routing decisions can benefit from not only the increased flexibility of such capabilities, but also from the verifiability of their packets and routes. Furthermore, this benefits transit providers, as they can now verify packets easily, allowing for convenient accounting and billing of distinct resource principals.

4. Network capabilities:

Platypus addresses both of these issues through the use of network capabilities. Abstractly, a network capability is made up of two fields: a waypoint and a resource principal identifier. The waypoint specifies a topological network location through which the packet should be routed and the resource principal specifies the entity willing to be charged for the routing request. Using intra-AS routing mechanisms, an AS can route packets for a given waypoint to different Platypus routers, thus giving it more control over the effects of source-routed traffic on an ISP's traffic engineering. We return to this issue in Section VII-D. For now, we will consider waypoints to correspond to a specific router within an AS. In Platypus, packets are stamped with a source-routing request by inserting a Platypus header immediately after the IP header of each packet and including some number of capabilities, encapsulating the existing payload. Fig. 2 shows the Platypus header format with one capability attached. The header contains fields for the protocol version (currently 0), a set of bit flags (whose use is described in Section IV-A.1), a length field (specified in terms of 32-bit words), a pointer to the current capability (also in terms of 32-bit words), and an encapsulated protocol field to facilitate de-encapsulation. Capabilities are apended immediately after the Platypus header. The latypus header and capabilities may be added by in-network stampers while the packet is in transit. Since anyone can use a capability to forward packets through the specified waypoint and bill the indicated resource principal, Platypus must ensure that eavesdroppers watching packets in the network cannot use capabilities they

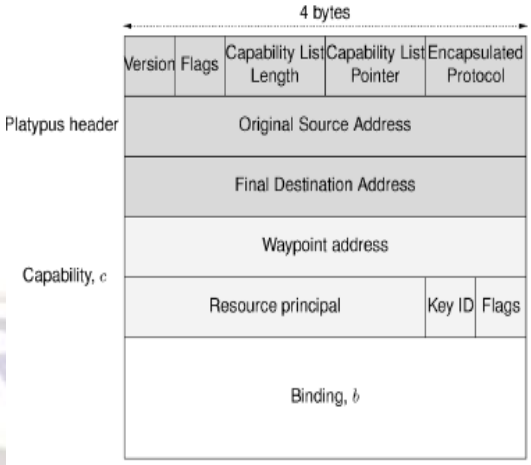

Figure 2: Platypus header format with a single capability and binding attached.

Bindingsare a function of the capability, the packet contents, and a secret known only to the owner of the capability. When a Platypus packet arrives at a waypoint, the Platypus router validates the corresponding capability and its binding. If the capability/binding pair is valid, the router updates the waypoint pointer (indicating the packet has already passed through this waypoint), sets the packet's current destination IP to the waypoint field of the next capability in the capability list, replaces the current source IP with its own (to prevent ingress filters from dropping the packet), and forwards the packet on. If no additional capabilities remain, the router replaces the original destination address. *A. MAC-Based Authentication*
Platypus prevents forgery of capabilities or their bindings with the *cascade construction* of Bellare *et al.* [22], which is provably secure given an underlying MAC that is a pseudorandom function (PRF), as most modern MACs are believed to be. We define a secret temporal key, , generated from the capability, , using a message authentication code (MAC) such as HMAC [24]. The MAC is keyed with , the key of the specified waypoint. This value is securely transferred to the resource Principal (in a manner described in Section IV). In order to use a capability, an individual packet must be stamped with the capability and a binding, ,where is the invariant [23] contents of the packet
(not including Platypus headers) with the end-to-end source and destination addresses substituted and the packet length field omitted.

5. Capability Manager:

Platypus gains significant flexibility from the ability to transfer capabilities. Entities can collect capabilities from multiple resource principals and construct source routes to which no single entity would otherwise have rights. We describe capability management in several steps: First, we detail how capabilities are generated both in general and in special cases. Second, we describe how resource principals

obtain temporal secrets for their own capabilities and capabilities delegated to them by others. Third, we present a policy framework for applying capabilities to IP packets.

Capability GenerationWhile capabilities are generally minted by an ISP, there are two important cases when individuals may wish to create new capabilities based on those provided to them by their ISPs.

Reply Capabilities Protocols such as TCP have been shown to work best when forward and reverse path characteristics are similar [18]. In order to use Platypus source routes, however, both ends of a flow must have their own capabilities and perform their own routing. Fortunately, it may often be the case that one of the communicating parties may wish to be responsible for both directions of the flow. For example, a client may wish to provide a server with a capability to enable the server to provide it with better performance. Platypus allows for resource principals to include a *reply capability* and its corresponding temporal secret as part of a packet stream for the recipient to use in response.

Capability Distribution There are three main aspects to wide-area capability distribution:

Bootstrapping, lookup, and revocation. We describe our approaches to each in turn.

Bootstrapping- To bootstrap the capability distribution process, we expect that each AS provides an interface (likely a Web server) through which resource principals establish their accounts. This can occur in many ways. For example, the server and resource principal can set up a secure channel (using SSL, for example), and, after negotiating payment, the server sends a resource principal ID, randomly generated capability master key , and the capability information to the resource principal.

Ordinary Capability Lookup*:* To look up the current temporal secret associated with a capability, a resource principal generates a request by encoding the capability and a special request opcode as a string and prepends it to the key-lookup subdomain (specified during the bootstrap process) in a DNS TXT lookup request, which is routed by DNS to an appropriate key server. For example,a request for a capability issued by ucsd.edu with key-lookup subdomain platypus.ucsd.edu would be request .platypus.ucsd.edu. The DNS response is a similarly encoded DNS TXT record containing the temporal secret for the requested key ID encrypted under the capability master key. The resource principal decrypts and verifies the response, yielding the current temporal secret for the specified capability. The use of DNS for key lookup may seem clumsy; a more natural approach might be to contact the key server directly. To contact the server, however, a resource principal would have to first perform a DNS lookup for the key server and then transmit its lookup request, requiring multiple round trips. Instead, Platypus piggybacks the request for a key, shortening the lookup latency to about one RTT, allowing for extremely short expiration intervals.By using DNS to distribute keys, Platypus realizes caching, distributed authority, and failure resistance without having to build a separate key distribution infrastructure. In particular, Platypus key lookups are cacheable since requests are plain text and replies are encrypted under the capability master key for the requested capability. If multiple requests are made for the same shared capability, DNS caching will automatically decrease the load on the key server.

Delegated Capability Lookup*:* Lookup of delegated capabilities is fundamentally different from ordinary capability lookup: parties must receive capabilities from a resource principal rather than from a capability server. We have devised a DNS-based mechanism that allows a server to distribute delegated capabilities to clients, leveraging the DNS lookup that typically precedes client-server exchanges on the Internet. If both the client and the server are Platypus-aware, the server can delegate a capability to the client as follows. Suppose a client wishes to contact a server server.ucsd.edu. Normally, a DNS resolver near the client issues a DNS query asking for the A record (IP address record) for server.ucsd.edu, which eventually is answered by the name server authoritative for ucsd.edu. Instead, we have the resolver issue a query for a TXT record for deleg.server.ucsd.edu (that is, it

prepends deleg to the DNS name). The DNS server recognises this as a request for (a) the IP address of server.ucsd.edu and (b) a delegated capability for sending traffic to server.ucsd.edu; it returns a TXT response to the resolver containing these two items. The DNS resolver installs the received delegated capability in a client-side stamper and returns the address to the client; the stamper can subsequently stamp traffic from the client to the server.

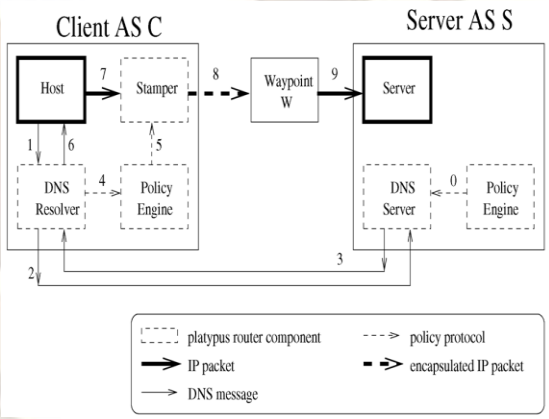

Fig. 3. Delegation and stamping in our policy framework.

Revocation

While expiration provides for coarse-grained control of temporal secrets, a resource principal may want to immediately revoke the current temporal secret when it suspects compromise. Platypus enables such revocation: to revoke a particular temporal secret, the resource principal computes the MAC of the capability and the current time under the capability master key and sends the pair, MAC,and the revocation opcode

encoded as a DNS request. Platypus routers periodically receive updated revocation lists from their associated key servers which they consult whenever validating packets. The revocation list for the current key ID is flushed upon key ID rotation.

6. Elliptic Curve Cryptography:

In general, public-key cryptography systems use hard-to-solve problems as the basis of the algorithm. The most predominant algorithm today for public-key cryptography is RSA, based on the prime factors of very large integers. Elliptic curves combine number theory and algebraic geometry. These curves can be defined over any field of numbers (i.e., real, integer, complex) although we generally see them used over finite fields for applications in cryptography. An elliptic curve consists of the set of real numbers (x,y) that satisfies the equation

$$:y^2 = x^3 + ax + b$$

The set of all of the solutions to the equation forms the elliptic curve. Changing *a* and *b* changes the shape of the curve, and small changes in these parameters can result in major changes in the set of (x,y) solutions.

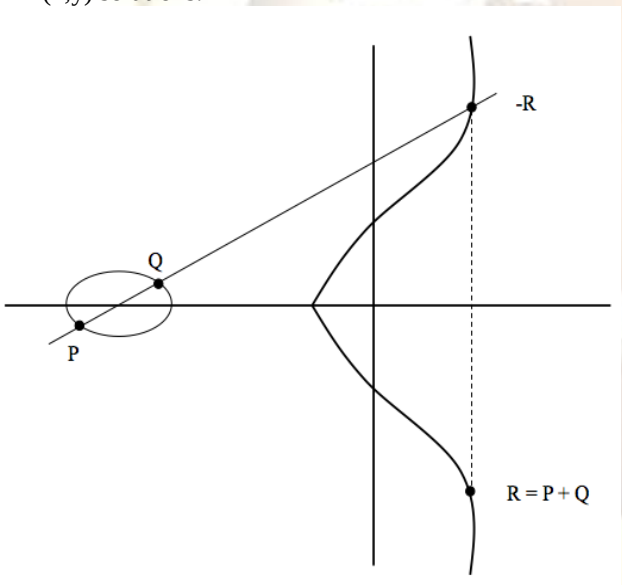

Figure 3 shows the addition of two points on an elliptic curve. Elliptic curves have the interesting property that adding two points on the elliptic curve yields a third point on the curve. Therefore, adding two points, P and Q, gets us to point R, also on the curve. Small changes in P or Q can cause a large change in the position of R.

So let's go back to the original problem statement from above. The point Q is calculated as a multiple of the starting point, P, *or*, Q = nP. An attacker might know P and Q but finding the integer, *n*, is a difficult problem to solve. Q (i.e., nP) is the public key and *n* is the private key.

ECC may be employed with many Internet standards, including CCITT X.509 certificates and certificate revocation lists (CRLs), Internet Key Exchange (IKE), Transport Layer Security (TLS), XML signatures, and applications or protocols based on the cryptographic message syntax (CMS). RFC 5639 proposes a set of elliptic curve domain parameters over finite prime fields for use in these cryptographic applications.RSA had been the mainstay of PKC for over a quarter-century. ECC, however, is emerging as a replacement in some environments because it provides similar levels of security compared to RSA but with significantly reduced key sizes. ECC key sizes are so much shorter than comparable RSA keys, the length of the public key and private key is much shorter in elliptic curve cryptosystems. This results into faster processing times, and lower demands on memory and bandwidth

7. Waypoint Deployment:

We now consider the impact of waypoint deployment on the effectiveness of Platypus -like source routing. Clearly, the more numerous the waypoints, the more control Platypus can assert
over a packet's path. By clustering the routers into groups which could be represented by a single Platypus waypoint, we attempt to determine the number of Platypus waypoints an ISP must deploy to provide a useful service to customers.

Packet size	68 byte	348 byte	1500 byte
Packet processing			
Null	172 ns	173 ns	181 ns
UMAC	695 ns	998 ns	1908 ns
Destination cache lookup	289 ns		
IP hdr build and verify	145 ns		
Packet transmission	1480 ns	1482 ns	1493 ns

Table 1 : Micro Bench marks for prkm All time are as measured by the CPU cycle counter

In particular,we study the impact on end-to-end one-way path latency of routing indirectly through a set of waypoints; we vary the number of waypoints available. Previous research indicates that it is often possible to achieve significant performance improvements by inserting one level of indirection in a packet's route [3], [8], [28]. We consider how the best achievable path latency increases as more waypoint choices are available, as this indicates how well chosen waypoints must be. Intuitively, since POPs represent a collection of routers in a region, and networks are dense near large cities and sparse elsewhere, routers that have similar latencies to a given set of observation points can be naturally clustered together. It may be sufficient to place Platypus routers in only a few locations, as speed of light delays comprise most of the delay seen by packets in uncongested wide-area backbones. Thus, multiple, local waypoints would not significantly affect latency. As expected, the more

waypoints, the closer the performance of the optimal cluster comes to performance of the optimal router. Somewhat surprisingly, however, the best cluster centers approach the optimal at a relatively small number of clusters, suggesting that a small number of indirection points are likely sufficient for substantial benefit; this applies equally to Platypus and any overlay or source routing system; this is likely due to geographic or POP locality among potential waypoints.

7.1 Waypoint Load Balancing

In all our application scenarios (from Section II-A), Platypus users forward their traffic through selected waypoints. Consider, for example, aWeb site that purchases Platypus service from an ISP; traffic that the server sends to specific clients uses Platypus to selectively improve performance. However, given the popularity of the website, it may overload a single waypoint at certain times of the day. To remedy this issue, we consider a policy in which the server selects a set of waypoints to forward traffic through and load balances across them. This functionality is important in many applications, since it is unlikely that a single waypoint can suffice for an arbitrarily large traffic volume.Using the Platypus policy framework described in Section V-C, we evaluate a Web server application scenario with probabilistic load balancing across two waypoints. Each client makes ordinary HTTP requests to the server. The server's replies are stamped according to a policy that begins by sending all response traffic through a single waypoint. Halfway through the experiment we change the policy such that the response traffic is load balanced at the granularity of a TCP flow.

7. Conclusion

We argue that capabilities are uniquely well-suited for use in wide-area Internet routing. The Internet serves an extremely large number of users with an even larger number of motivations, all attempting to simultaneously share widely distributed resources. Most importantly, there exists no single arbiter (for example, a system administrator or user logged in at the console) who can make informed access decisions. Moreover, we believe that much of the complexity of Internet routing policy stems from inflexibility of existing routing protocols. We aim to study how one might implement inter-AS traffic engineering policies through capability pricing strategies. For example, an AS with multiple peering routers that wishes to encourage load balancing may be able to do so through variable pricing of capabilities for the corresponding Platypus waypoints.

8. References:

[1] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "Theend-to-end effects of Internet path selection," in *Proc. ACM SIGCOMM*, Sep. 1999.

[2] S. Agarwal, C.-N. Chuah, and R. H. Katz, "OPCA: Robust interdomain policy routing and traffic control," in *Proc. IEEE OPENARCH*, Apr. 2003, pp. 55–64.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. T. Morris, "Resilient overlay networks," in *Proc. ACM SOSP*, Oct. 2001.

[4] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian,"Delayed Internet routing convergence,"*IEEE/ACMTrans. Networking*, vol. 9, no. 3, pp. 293–306, Jun. 2001.

[5] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in cyberspace: Defining tomorrow's Internet," in *Proc. ACM SIGCOMM*, Aug. 2002.

[6] G. Huston, "Commentary on inter-domain routing in the Internet," in *IETF, RFC 3221*, Dec. 2001.

[7] X. Yang, "NIRA: A new Internet routing architecture," in *Proc. ACM SIGCOMM FDNA*, Aug. 2003.

[8] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of Internet path selection," in *Proc. ACM SIGCOMM*, Sep. 1999.

[9] I. Stoica and H. Zhang, "LIRA: An approach for service differentiation in the Internet," in *Proc. NOSSDAV*, Jun. 1998.

[10] D. Zhu, M. Gritter, and D. R. Cheriton, "Feedback based routing," in *Proc. HotNets*, Oct. 2002.

[11] I. Castiñeyra, N. Chiappa, and M. Steenstrup, "The Nimrod routing architecture," in *IETF, RFC 1992*, Aug. 1996.

[12] D. Estrin, T. Li, Y. Rekhter, K. Varadhan, and D. Zappala, "Source demand routing: Packet format and forwarding specification," in *IETF, RFC 1940*, May 1996.

[13] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proc. ACM SIGCOMM*, Aug. 2002.

[14] A. C. Snoeren and B. Raghavan, "Decoupling policy from mechanism in Internet routing," in *Proc. HotNets*, Nov. 2003.

[15] M. Caesar and J. Rexford, "BGP policies in ISP networks," *IEEE Network*, vol. 19, no. 6, pp. 5–11, Nov. 2005.

.

[16] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 5, pp. 19–30, 2003.

[17] W. B. Norton, "Internet service providers and peering," in *Proc. NANOG*, Jun. 2000.

[18] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, "The effects of asymmetry on TCP performance," in *Proc. ACM Mobicom*, Sep. 1997.

[19] J. Black and P. Rogaway, "A block-cipher mode of operation for parallelizable message authentication," in *Advances in Cryptology (EUROCRYPT' 02)*, 2002, vol. LNCS 2332.

[20[K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D.Wetherall, "Improving the reliability of Internet paths with one-hop source routing," in *Proc. USENIX OSDI*, 2004.

[21] JANNOTTI, J. Network layer support for overlay networks. In *Proc. IEEE OPENARCH* (New York, New York, June 2002), pp. 3–13.

[22] FERGUSON, P., AND SENIE, D. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2267, Internet Engineering Task Force, Jan. 1998.

[23] POSTEL, J. Internet Protocol. RFC 791, Internet Engineering Task Force, Sept. 1981.

[24] TAHILRAMANI KAUR, H., , WEISS, A., KANWAR, S., KALYANARAMAN, S., AND GANDHI, A. BANANAS: An evolutionary framework for explicit and multipath routing in the internet. In *Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture* (Karlsruhe, Germany, Aug. 2003), pp. 277–288.

Mohammad Zaheer Abbas received his B.Tech In Information&Technnolog and Engineering from Nalanda Engineering College,Guntur, Andhrapradesh ,India,in 2009. He is Pursuing M.Tech in Computer Science and Engineering in JNTU Kakinada,A.P,India During 2010-2012. His research invites Network Security