Workgroup: Network Working Group
Internet-Draft: draft-menon-svr-01
Published: 29 March 2022
Intended Status: Informational
Expires: 30 September 2022
Authors: A. Menon, Juniper Networks
P. MeLampy, Juniper Networks
M. Baj, Juniper Networks
P. Timmons, Juniper Networks
H. Kaplan, Juniper Networks

Secure Vector Routing (SVR)

Abstract

This document describes Secure Vector Routing (SVR). SVR is an overlay inter-networking protocol that operates at the session layer. SVR provides end-to-end communication of network requirements not possible or practical using network header layers. SVR uses application layer cookies that eliminate the need to create and maintain non-overlapping address spaces necessary to manage network routing requirements. SVR is an overlay networking protocol that works through middleboxes and address translators including those existing between private networks, the IPv4 public internet, and the IPv6 public internet. SVR enables SD-WAN and multi-cloud use cases and improve security at the networking routing plane. SVR eliminates the need for encapsulation and decapsulation often used to create non-overlapping address spaces.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There exists a need to communicate network requirements between IP routers and networks to provide an end-to-end experience. Selection of specific paths whose attributes meet or exceed the networking requirements are an objective of SVR. There is also a need for applications to communicate their requirements to networks. This need is increasing as workloads move to public clouds and the numbers of cloud locations increase. The standard practice today is to use an overlay network of tunnels to create a virtual network. SVR overlay is being proposed as an alternative to using tunnels. SVR simplifies the network by virtue of having only one network layer. SVR securely transports traffic with authentication and adaptive encryption. The absence of tunneling overhead reduces bandwidth. Since SVR specifies requirements abstractly, it also has the capability to interwork policies between different networks and address spaces.

Most WAN networks are deployed with a virtual private network (VPN) across IP backbone facilities. VPNs have the significant disadvantage of carrying additional network layers increasing packet size and leading to IP fragmentation as well as reduced bandwidth.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Overview

```
                   +---------+
                   |Network2 |
+----------+       |         |       +----------+
|         SVR<---->+<-L3-IP->+<---->SVR         |
|          |       +---------+       |          |
|Network1  |       +---------+       |Network 4 |
|         SVR<--->SVR       SVR<--->SVR         |
+----------+       |         |       +----------+
                   |Network3 |
+----------+       |         |
|Client SVR|<---->SVR        |
+----------+       +---------+
```

Figure 1

An SVR implementation describes a network requirement semantically and shares this as metadata with a routing peer. The requirement to a peer is conveyed by means of a cookie, often referred to as first packet metadata, which is placed in the first packet of a session that is targeted towards the SVR peer. SVR requires session state on every participating SVR router and sets up a bi-flow (matching forward and reverse flows) based on the requirement. Once the session is established bi-directionally, the cookie is not sent in subsequent packets, resulting in elimination of additional overhead.

Benefits from this approach include:

* Tunnel Compression: The metadata contains information required to eliminate tunnel header information for established sessions. This can result in anywhere from 12% to 100% bandwidth savings when compared to IPSEC based tunnels depending on the original packet size.

* Elimination of Elephant Flow problems: Tunnels are very long lived and often contain large aggregates of inner flows. Tunnels are also often fixed on a specific network "hash" while each SVR session has a unique network hash.

* QoS support is per flow, not per packet: Because each SVR flow has a unique 5-tuple on the wire, standard MPLS routing and QoS techniques work seamlessly. Adding QoS to Tunnels requires QoS on entry to a tunnel, tunnel DSCP markings, and policies to copy/map inner packet DSCP to Tunnel Packet DSCP. In practice many core networks do not look at the DSCP markings once a fast path forwarding rules are established.

* Avoid Re-encryption: Tunnels often encrypt all traffic. Much of the traffic in the tunnel is already encrypted, thus there is a re-encryption penalty. SVR support adaptive encryption which performs encryption on only those sessions that require it.

* Firewalls and security proxies can intercept TLS sessions and perform decryption and encryption if they support SVR metadata. This is not possible with IPSEC tunnels by design.

* Scaling of software based encryption is much higher when session state is available. Encryption performance is limited to what is possible in a single processing core for a single session, and at the time of this document being written the limit is currently 1.5GigE for Tunnel termination.

1.3. Definitions

The following terms are used throughout this document.

Authority: This defines the owner of an SVR namespace. Each namespace owner can allocate Tenant names (representing collections of network endpoints sharing common network enforcement policy), and Service names (representing accessible destinations and traffic treatment policy). Authority namespaces must be unique to permit internetworking. Claiming and resolving disputes about authority naming are outside the scope of this document.

Tenant(s): This is a textual description defining network endpoints that share common access policy (allow lists or block lists to network destinations). These may be mapped using any known technique including source IP address mask, a VLAN tag, ingress interface, provided by an authentication system, or even client supplied, and this mapping is outside the scope of this document. Often these are location specific definitions, but the Tenant has global meaning within an authority. Tenant names can conform to domain name syntax, and be expressed as hierarchical structures (i.e., location.department.example).

Service(s): This is a textual description of what server(s) can be accessed with this intent. Examples include Zoom, or Office365/ Outlook. Although outside the scope of this document, these could be defined with any known technique, including URLs, IP address(es) protocol(s) and port(s), CIDR block(s), etc. Having a single text name to describe a network destination makes defining network requirements easier. Other Service specific network requirements including Quality Policies and Security Policies can be associated with Services in data models, but are not described in this document.

Context: This is the original "5-tuple" of an IP packet, including source IP, source port, destination IP, destination port, and protocol. Optionally, Layer 2 information such as MAC Address or VLAN tags may be included for certain use cases if required.

Signature: The metadata packets MUST be cryptographically signed using HMAC by the source router, and all packets traversing an SRV peer pathway SHOULD have an HMAC signature so the next hop router can authenticate the sender of the data and verify its integrity. The portion of the packet that is signed must not include the IP header, as it may go through a NAT or IPv4 to IPv6 conversion.

Direction: This is inferred, and not a specific metadata field. The Direction represents the intended client to server direction. The initial network packet of a communication session indicates this direction. For example, a TCP SYN packet would travel from client to server, defining the direction of an service. Forward direction is always client to server, and reverse is always server to the client. These directions have nothing to do with a network topology (for example, hub and spoke), and a single network path could have forward sessions going bi-directionally -- traffic going from node A to node B may represent the forward direction for some sessions and the reverse direction for other sessions.

Peer: An SVR Peer is a client, server, or router that supports the SVR protocol. The SVR Peer could be either directly adjacent, or reachable through an IP network. The SVR Peer should not be confused with BGP Peer. Since SVR Peers must be able to reach each other, and because SVR Peers are often deployed at network edges, SVR Peers can also be BGP Peers. In this document peer will always mean SVR Peer.

Waypoint: A Waypoint is a reachable IP Address associated with an SVR Routers interface. Some physical interfaces may have multiple IP Addresses, and as such a single physical interface could represent multiple Waypoints. In some cases, routers use dynamically assigned addresses on interfaces. In these cases, a Waypoint address may change dynamically.

Peer Pathway:
An SVR Peer Pathway is a unique pair of Waypoint addresses that can reach each other. The path can be defined as either a pair of IP addresses or a pair of domain names that resolve to IP Addresses. Peer Pathways have attributes related to availability, performance (jitter, latency, packet loss) and cost. Techniques such as BFD [RFC5580].

2. Theory of operation of Secure Vector Routing

Secure Vector Routing is a session stateful routing overlay that operates at edges of networks where stateful NATs are normally performed. It is at these same locations where multi-path routing is being deployed. These locations include edge routers located at branches, data centers, and public clouds. SVR maps local network requirements into administratively defined text strings that have global meaning. These are communicated or signaled by insertion of a networking cookie called SVR metadata directly into IP Packets in transit.

SVR metadata is inserted into existing packets directly after the L4 header (see Section 4.2.) The metadata in the first packet of a new session (TCP or UDP bidirectional flow) can be used in path selection and security. Metadata can be inserted in any subsequent packet to change/update the networking requirements. The metadata is inserted into the payload portion of a packet to guarantee it makes it unchanged between SVR routers.

Sessions supported by SVR include TCP, UDP, UDP Unicast, point-to-point ethernet, and ICMP. Sessions are characterized by having an initial first packet that is unique to an SVR router. Often this is described as a unique 5-tuples as seen by the router. Sessions start when the first packet is processed, and end when either the L4 protocol indicates the session is completed (TCP FIN/FIN ACK) or there has been no activity for a length of time (UDP, ICMP, UDP Unicast, point-to-point ethernet).

2.1. Directionality

SVR utilizes the concept of session direction. The direction of the session is what creates a Secure Vector. Routing policies include a Tenant (source) and Service (destination) pair that exactly match the direction of sessions. When describing metadata in this document, direction is either forward or reverse; it is not tied to network topology, but rather the direction of session establishment. For TCP, the forward direction is always the client side towards the server side. For UDP, the forward direction is from the sender of the first packet. Reverse is the opposite direction. On a given pathway Secure Vector routes could be traversing on the same pathways with opposite directions.

Metadata formats described in this document will be labeled as "forward" or "reverse". Forward metadata is inserted in packets going from client to server. Reverse metadata is inserted in packets that travel from server to client.

2.2. SVR with Other Traffic

SVR co-exists with traditional routing. In fact, the router interface addresses known as Waypoints in this document MUST be reachable via traditional networking for every peer relationship. When packet routing is being decided in the router, should the route resolve to an SVR capable router (i.e., the next hop address returned in the route equals a known Waypoint address of an SVR Peer) then metadata MAY be inserted and session stateful SVR is performed. Otherwise, the packet is forwarded like any traditional IP router.

2.3. Metadata Handshake

To ensure the metadata is received and understood between peers, a handshake is performed. A router that supports SVR peer pathways inserts metadata for each packet flow in the following circumstances:

* It is a "forward" packet representing a new session and the ingress node has not yet received any reverse metadata from the recipient egress node.

* It is a "reverse" packet from the recipient egress node to the initiating ingress node and recipient egress node has not received forward packets from this session without metadata.

These two comprise what is known as the "metadata handshake" -- that is, the initiating router includes metadata in all packets it sends to the recipient router until it receives a reverse packet with metadata from that recipient. Likewise, the recipient continues to send metadata to the initiating router until it receives a packet without metadata. This is how two routers acknowledge receipt of metadata from their counterparts: the absence of metadata in a packet indicates that it has received metadata from its counterpart.

2.4. Pathway Obstructions

Firewalls and middleboxes that sit along a peer pathway may not propagate TCP SYN messages with data in the payload (Despite being valid), or may verify sequence numbers in TCP streams (which are invalidated due to the inclusion of SVR metadata). The two devices that represent the peer pathway endpoints may determine through testing if there is a firewall, NAT, or other active middlebox between the two routers. Procedures like STUN [RFC8489], TURN [RFC6062], and ICE [RFC8445] are well known, and not included in this document.

If a NAT is detected on the Peer Pathway, the SVR Router that determines its Waypoint address is being changed saves this as an attribute of the pathway. The NAT will change the port address assignment, and require NAT keep alives as exemplified in Section 5.2.

If a middlebox is detected, the packets can be UDP-transformed i.e., the protocol byte can be changed from TCP to UDP by the transmitting router and restored to TCP by the receiving router for packets flowing in both directions. See Section 4.2.7 and Section 4.6.3 for more information.

2.5. Metadata removal

To prevent breaking any applications, there MUST be a 100% guarantee that metadata inserted by a participating SVR device is removed prior to the consumption of the data by the application service. If the client and server support metadata, then the network intent can be sent end-to-end. When a mid-stream packet router wants to insert SVR metadata, it must guarantee that the packet is directed to a next hop device that will understand and remove the metadata.

A router can be certain an SVR capable router is on the path when the next-hop address returned from a FIB table exactly matches a known peer Waypoint address. Before sending the packet with metadata to the Waypoint address, the originating SVR router should determine the Peer reachability as exemplified in Section 3.1.

If the next-hop is not a known reachable peer, SVR metadata insertion MUST not be performed.

2.6. Modification of transport addresses

To guarantee that the packet will go to a specific router, the destination address for the packet is changed to the waypoint address of the chosen peer. The original addresses are stored in the forward context (see Section 6.4.1) and can be recovered when needed. This is similar to IPv6 segment routing (see [RFC8986]) or a LISP (see [RFC6830]) RLOC with the exception that the original addresses are stored in metadata within the payload portion of the packet, and not the IP Network Header.

Selection of the Waypoint address to send is implementation specific. In the general case a standard FIB lookup returns one or more IP Address(es) (Waypoints) of the next SVR peer. When more than one Waypoint address is returned from the FIB, additional logic can be applied to select the best Waypoint based on observed peer pathway quality OR session layer load balancing. See Section 3.1 for exemplary details.

To provide a return path for the return flow the source SVR peer changes the source address to be its own egress Waypoint address. This provides a guarantee of a symmetric flow. The state of the session MUST be held in both the source SVR router and the destination SVR peer.

The address translation rules for the session become state information that is processed on every packet after the metadata handshake. All 5 tuples of addressing information are updated bidirectionally for the session. This action replaces tunnel encapsulation and decapsulation (tunnel compression), and is an order of magnitude simpler computationally.

2.7. Optional use of Tenants and Service names for Routing

The metadata contains contextual IP Addresses (sources, destinations, and waypoints) along with textual service names (i.e., Zoom, Office365, etc.). The SVR routers can apply policies and route sessions based on the textual names if they have a route information base that contains service names. When performing name based routing, a destination NAT is often required when exiting the SVR network. The primary use case for this is networking between public clouds such as AWS and Azure.

With semantic based routing, the use of Dynamic DNS to locate a service can be eliminated if clients support SVR. Clients can simply request the service by name, and the SVR router can resolve the route, and deliver the session to the best location. The last SVR Router on egress performs a destination NAT for the chosen best instance of a service.

A local DNS server resolving service addresses to a nearby SVR router can also provide semantic based routing. This can eliminate the need to use dynamic DNS for locating services inside data centers.

2.8. Unique 5-Tuples for Every Session

To avoid sharing a hash with all traffic, and to make sessions completely independent on peer pathways, the source port and destination port can be assigned any values that are unique by the source router. When there are no NATs between the two router interfaces, this permits 2^32 (4,294,967,296) different unique sessions on a peer pathway. If there are source NATs, this will be reduced to 2^16 (65,536) different unique sessions. Ports can be reassigned if not in active use. It is also possible that middle boxes will limit what destination ports are permissible, reducing the number of possibilities. Due to all these reasons, range of ports that can be used on a peer pathway are provisioned by an administrator.

The ingress SVR peer (client side) assigns both source and destination ports, even ports for local (source port) and odd ports for remote (destination port). This provides total uniqueness between any two peers, with no negotiation or collision possibilities. This reduces the number of sessions originating by a router to half of the total sessions (or 2^30). Think of the two ports as a Session Identification Tag. Even if a session traveling in the opposite direction was allocated the same exact ports, because the source address and destination addresses would be swapped, the 5-tuples on the wire remain unique.

This unique tuple per TCP/UDP session also allows any DSCP or QoS scheme to work properly. Those fields in the original packet were not modified and the underlay network routers will see those fields on a session-by-session basis.

2.9. Session Packets Post Metadata Exchange

After the metadata handshake has been completed. all subsequent packets are simply translated (all 5-tuples, bidirectionally). This is a very efficient process compared to IPSEC encapsulation which requires memory copies, new header creation, completely new packet checksums, and mandatory encryption.

2.10. Session State Requirements

Each participant (peer) in secure vector routing must maintain state for every active session. This includes the full set of original addresses and translations required. This allows participants to stop sending metadata once it has been received by the peer. There are two possible scenarios for how state could be lost. Either the ingress of the SVR session (source peer) could lose state, or an intermediate (downstream peer) SVR peer could lose state.

Determining if an SVR router is an ingress verses a peer SVR router is based on the arriving packet's destination address. If the address is NOT the interface address of the router, it is an ingress SVR router. Alternatively, if the address matches the interface address of the router, there are two possibilities.

Packet is a legitimate SVR packet from a peer and State has been lost:

Every packet in an SVR session SHOULD have an HMAC checksum to prevent replay attacks. If a packet arrives at an SVR router, with the destination address of the router, and a source address of a known peer, the HMAC checksum can be verified. If verified, this is indeed, a case of lost state with a SVR Peer.

Packet is not a valid SVR Session Packet: If either the source address of a packet does not map to a valid peer or the HMAC signature does not validate; the packet is invalid and MUST be dropped. This represents a security event and should be noted as such.

After determining if the router is an ingress or egress SVR router when there is a flow miss, the state recovery techniques for each type of lost state is listed below.

Ingress SVR Loses State: The ingress router will treat this packet as a new session, allocate and insert metadata. The packet will be forwarded to the next SVR router. This upstream SVR peer may or may not have state for the existing session. By reviewing the metadata's forward context (original packet 5-tuples) the router can determine if there is a collision with an active SVR session. If so, the terminating SVR router will accept the new metadata, and adopt the new proposed addresses and UUID's, essentially merging the two sessions. If there is not a collision with an existing session, the packet is routed as a new session.

SVR Peer Loses State: The peer router without state will create reverse metadata asking for the remote peer SVR (i.e., where the SVR packet was sent from) router to retransmit metadata for this session. The metadata request will be sent back to the peer router using the exact address and ports for the packet received without state, only reversed. Please see Section 6.3.6 for the reverse metadata sent. This reverse metadata request is sent on the peer pathway that sent the packet, with the source port and destination port matching the packet with missing state. The upstream router will include first packet metadata for the session in the next packet of the session.

2.11. NATs and Session Keep Alive

Each SVR router (peer) must statefully remember the source address that a session with metadata was received on. This may not be the same address the router sent a packet from due to a NAT or Firewall in the pathway. Routers use both provisioned and learned waypoint addresses. Routers MUST store the actual waypoint addresses received on the wire from a peer.

When a firewall or middlebox is detected, the SVR router behind such a device must send metadata packets periodically on idle sessions to keep any firewall pinholes translations from being removed. For every UDP and TCP session that has seen no packets after a programmable length of time (20 seconds is recommended), then the SVR Peer should send an SVR Control Message on the peer path with the source and dest ports from the idle session's saved state. See Section 6.3.6 for more information and see Section 5.2 for an example.

3. SVR Example

3.1. SVR Multi-path Routing Description

The example below shows two SVR capable routing peers with multiple peer pathways.

```
 Client
  LAN
10.x.x.x
   |
   |  +--------+                                  +---------+
   |  |        |                                  |         |
   |  |        |                                  |         |
   +->] East  [172.15.0.1<----MPLS----->172.15.10.1] West   |
      | SVR    |                                  | SVR     |
      | Router[107.0.8.186<-Internet-+->52.42.68.58] Router |
      |        |                      |           |         |
      |       [192.0.2.1<-----LTE---/             |        [<--+
      |        |                                  |         |   |
      +--------+                                  +---------+   |
                <========= Peer Pathways ========>              |
                                                                |
                                                                |
                                                          172.15.11.x
                                                              LAN
                                                             Servers
```

Figure 2

Note: The client, server, and MPLS network support the private address space 172.15.x.x natively, but the internet and LTE networks do not. This is an example of using secure vectors to join networks together.

The first step is that routers would apply any locally defined static L3 routes, and begin advertising and receiving routes using L3 networking protocols (BGP, OSPF, etc.) from their IP peers to build a forward information base or FIB. This is required initially to ensure that the waypoints are reachable bidirectionally.

The second step is for both the East and West routers to establish their SVR peering. East and West independently attempt to communicate with BFD to each other's interfaces and measure path characteristics such as jitter, latency, and packet loss. In our example, assuming 100 percent success, the resulting peer pathways would be:

```
East's Peer Pathways
  Name      Description                   Characteristics
  MPLS      172.15.0.1->172.15.10.1       20ms Lat, 0 Loss,  2 Jit
  Internet  107.0.8.186->52.42.68.58      30ms Lat, 0 Loss,  3 Jit
  LTE       192.0.2.1->52.42.68.58        50ms Lat, 0 Loss, 15 Jit

West's Peer Pathways
  Name      Description                   Characteristics
  MPLS      172.15.10.1->172.15.0.1       20ms Lat, 0 Loss,  2 Jit
  Internet  52.42.68.58->107.0.8.186      30ms Lat, 0 Loss,  3 Jit
  LTE       52.42.68.58->192.0.2.1        50ms Lat, 0 Loss, 15 Jit
```

Figure 3

For this example, our assumption is that there are servers that are located inside 172.15.11.0/24 at the West location. West advertises this route to East on each path available to it. East's FIB will look like this:

```
  East's Forward Information Base (FIB)
     Route              Next-Hop IP Addr
     ----------------   -----------------
     172.15.11.0/24     172.15.10.1
     172.15.11.0/24     52.42.68.58
     ....
     [FIB Entries to reach waypoints omitted]
```

Figure 4

Additionally we will assume there exists a network policy created by the authority Example that defines a tenant "engineering" as 10.0.0.0/25 VLAN2, and "github.example" as 172.15.11.23 using TCP port 22. The provisioning and/or discovery of this policy is outside the scope of this protocol description.

A first packet from an engineering client with github as a destination received at the East SVR Router will result in a search of the FIB and result in two possible next-hop IP Addresses. East will consult its SVR Peer Pathway list and recognize that three of its peer pathways have an exact match of this next-hop IP Address. These represent the three possible pathways that may be used for routing this session. The resulting potential routes are:

```
Possible Routes
  MPLS      20ms Latency, 0 Loss,  2 Jitter
  Internet  30ms Latency, 0 Loss,  3 Jitter
  LTE       50ms Latency, 0 Loss, 15 Jitter
```

Figure 5

The East router can now choose which pathway (peer pathway) is desired for the specific session. If the East router has quality service levels to maintain, it can choose from any of the peer pathways based on their current quality metrics. If all things are equal, the East router could load balance using approaches like "least busy" or other techniques. Once a peer pathway is chosen, the first packet metadata is constructed, inserted into the first packet, and sent down the chosen pathway to the West peer router.

For this example, the private address space in the LAN supported by the East Router is different. This is often the case with large networks. This is illustrative of a branch router performing network address translation (NAT) on a source address to solve overlapping address problems.

In this specific case, assuming MPLS was chosen, East would perform first packet processing resulting in the insertion of metadata in the first packet (see Section 3.6.1) and send it out East's interface with a source address of 172.15.0.1 and a destination address of 172.15.10.1. These are the exact addresses of the MPLS Peer Pathway.

Both the East and West routers would use the same address pairs (only reversed) for the bidirectional session, using the allocated source and destination ports to recognize the specific session. All packets from all sessions on a peer path will have the same exact IP addresses, differentiated solely by their port numbers.

3.2. Optional FIB Containing Service Names

SVR first packet metadata contains text strings that contain service names. SVR routing can route traffic by these names if the FIB contained text entries. There are some use cases where this might make sense:

Avoiding Dynamic DNS: Dynamic DNS is used to augment network routing protocols by answering the question: What best IP Address is available and best for a session now? Dynamic DNS can be plagued by delays in real time updates and additonal complexity and cost. In private networks, path service state may not be reflected in Dynamic DNS responses.

Multi-Cloud Networking: Public clouds run service instances on dynamically allocated private IP addresses. They provide very accurate and responsive DNS updates to help find IP addresses for networking. These DNS services are not available outside the cloud, making internetworking difficult. SVR Routers can use DNS resolution to find IP Addresses for named services.

Below is an example FIB that contains named services and traditional FIB entries. The next-hop addresses were changed to Waypoint Addresses to reflect the FIB is now an SVR fib containing service names, protocols, and ports.

```
East's Extended SVR Forward Information Base (OPTIONAL)

                                                        Egress
Service Name        Route                Waypoint    Action
--------------     ------------------  ------------  --------
github.example     172.15.11.23:TCP:22  172.15.10.1  FWD
github.example     172.15.11.23:TCP:22  52.42.68.58  FWD
logsvc.example     172.15.11.20:UDP:514 172.15.10.1  DNS
logsvc.example     172.15.11.20:UDP:514 52.42.68.58  DNS
https.example      172.15.11.24:TCP:443 172.15.10.1  DEST NAT
                                                     -196.168.1.1
                                                     -196.168.1.2
                                                     -196.168.1.3
   [FIB Entries to reach waypoints omitted]
```

Figure 6

Longest prefix matching (LPM), protocol and port will be used to match Routes for packets intended for github on ingress to SVR. The text string "github.example" will be used by all other SVR routers until egress from SVR. The SVR fib can be used to LPM match on IP addresses and exactly match protocol and ports. In the above illustrative example, only three protocols are supported (SSH, Syslog, and HTTPs). All other packets will be denied by default.

The egress action in the SVR fib can be used to support three different egress actions:

Forward Packet (Default): Restore the IP Addresses and forward. If a source NAT is provided in the metadata, NAT the source address.

DNS: Use DNS to resovle the service name locally. In this example DNS resolution procedures would be used on egress to resolve "logsvc.example".

DEST NAT: NAT the destination address to one (or load balance to a pool of addresses). This is identical to load balancers.

These named routes can co-exist with traditional FIB entries shown above. SVR will always matched a named route first, and fall through to the generic routes second.

3.3. SVR Security Definitions

For basic SVR functionality to work between peers, there must be a Authority wide provisioned set of rules. These rules include:

HMAC Method: This describes the method/technique for signing SVR packets. This could be SHA1, SHA256, or SHA256-128.

Use Time Based HMAC: This is either YES or NO.

HMAC Metadata or ALL: This is NONE, Metadata Only, ALL

Metadata Block Cipher: This is either NONE, AES128, AES256.

SVR does not limit the use of ciphers and techniques to just those listed. The requirements for both signatures and encryption are that the results are fixed well known block sizes.

Security Policies are used during session setup to setup payload encryption specifically for individual sessions. These are exchanged in first packet metadata.

For this example will use the following SVR security definitions.

```
HMAC: (On, time-based, SHA256-128, ALL Packets)
Metadata Encryption (On, AES256)
```

Figure 7

3.4. Time Based HMAC Details

To positively authenticate and provide integrity for SVR session, SVR peers use Time Based HMAC signatures. HMAC signatures are defined in [RFC2104]. Please see Section 4.5.1.

In our example, we are using SHA256-128 with a size of 16 Bytes.

3.5. Security Rekeying Considerations

Every metadata transaction includes a security ID header TLV (see Section 6.3.2). Although key management is outside the scope of this document, managing which key version to use is an important aspect of this design.

Each SRV Router will have its initial key (version 1) and may have an updated key (version n) over time. The security key version is always sent in metadata to ensure the peer knows which key to use to decrypt the metadata just sent. If a peer only has version 1 of a key, and metadata arrives specifying it is now at version 2, the SVR router must obtain the new key before it can process any packets.

For networks that are large and actively performing key management, there may be multiple versions of a key active, and SVR routers MUST be able to utilize any key for a reasonable amount of time.

3.6. New Session Initiation Detailed

The diagram below shows the example github TCP session flowing between a client and server through the East and West routers in our example network.

Ladder Diagram for SSH Example:

```
Engineering                                      Github
Client . . . . . . . . . . . . . . . . . . . . . Server
  |                                                 |
  +         East Router             West Router     |
  |            |                        |           |
  +---SYN----->|                        |           |
  |            |--SYN[MD]-------------->|           |
  |            |                        |--SYN----->|
  |            |                        |           |
  |            |                        |<--SYN/ACK-|
  |            |<------SYN/ACK[RMD]-----|           |
  |<--SYN/ACK--|                        |           |
  |            |                        |           |
  |            |                        |           |
  |<==== Session Packets Flow with No Metadata ====>|
```

The East Router MUST construct and insert metadata[MD] in the first packet of the SSH session, which will be a TCP SYN packet. The West Router must remove the metadata, and forward the SYN packet, and wait for the server to respond with a SYN/ACK. Upon receipt of the SYN/ACK, the West Router will create reverse metadata [RMD], and insert it into the SYN/ACK. This will create the metadata handshake for the SSH session. All forward and reverse metadata are inserted into existing packets if possible.

When a client or router detects that a new session is being established, the East Router will insert metadata into the first packet to communicate intent to the West Router. At both East and West Routers, the first packet will require specialized handling. Detecting a first packet for a session is protocol specific. For TCP, it's a new 5-Tuple packet (new flow) with the just the SYN flag set. For UDP, it's simply a new 5-Tuple packet not currently in active use.

3.6.1. East First Packet Processing

Utilizing the same example, assume that the packet shown below arrives on the East Router from the Client LAN. The packet is the result of an engineer attempting to access a github service via SSH.

Arriving Packet at East Router

```
Packet received on LAN side East Router [1]
Engineer using SSH to access Github
+---------+--------------------+-------------+----------+
|L2 HDR   | IP Header          | TCP Header  | PAYLOAD  |
|  VLAN=2 |    SRC=10.0.1.1    |   Sport=6969 |   Data   |
|         |    DST=172.15.11.23 |   Dport=22  |  (N/A)   |
+---------+--------------------+-------------+----------+
```

3.6.1.1. Determine Tenant

Determine the Tenant. The tenant is a text name which describes the routes and policies that are available for a group of source IP addresses. Tenants are like security zones. In our example, the "engineer" is based upon VLAN 2, and the tenant will be "engineer" as named by the authority "example". The configuration and data models to map physical network attributes to named tenants is implementation specific. Associating a default tenant with every logical interface on a SVR Router is recommended.

3.6.1.2. Determine Service

There are multiple ways to determine what an intended service is. Application Identification technology is used that understands all popular SaaS offerings. These techniques use combinations of IP address ranges and ports, SNI fields in TLS, Common Name from Certificates, and extraction of URLs from http requests. Most popular SaaS vendors today publish and update frequently their CIDR blocks and ports used by their services. This is out of scope for this document.

Longest prefix matching algorithms are used to extract the major and key services at a site. If there is traffic which cannot be identified accurately, often it will be placed into a "catch-all" service called "internet".

We will assume for this document, that the address 172.15.11.23 is a well known address for git servers at Example, and port 22 is known to be SSH.

3.6.1.3. Determine Network Requirements

Once the tenant and service have been determined, a lookup for network requirements can be determined. The requirements should include

Example Network Requirements

```
SERVICE: github
  Tenants Allowed: engineering
  Tenants Denied: release.engineering
  Quality: latency < 40ms
  Payload Encryption:Not Required
```

The above definition for github defines an example network requirement. Access policies determine which tenants are allowed, and if any specifically denied. The Quality policy defines the service level experience requirements. Secure Vector Routing exchanges tenants, services, and security policies using character strings in metadata. Access and quality policies are defined and used locally within a router and logically tied to the service. The implementation of quality and access policy controls are site specific. For example, VLAN based subnets may have different meanings at various locations. Also, QoS management schemes may be different for different network areas.

3.6.1.4. Picking a Peer Path

As stated previously, the East Router has three peer paths that can reach the destination based on L3 reachability. The next step is to apply the network requirements to see which of the peer paths remain. Our policy requires latency to be less than 40 Msecs, and this effectively eliminates East's LTE pathway from consideration. The remaining two pathways MPLS and Internet are both possible. We will choose MPLS as it has the lowest latency, offering the user the best experience.

Many different criteria can be used in selecting a peer pathway. In practice, how busy a peer path is and its capacity result in new sessions routing to 2nd best options. Often simple load balancing is used. In cases where there are higher costs (such as LTE or 5G networking), these may be held in reserve for backup or disaster recovery. The actual algorithms for picking peer pathways are outside the scope of this protocol.

3.6.1.5. Allocate Source NAT if Necessary

In this github example, there is a source NAT at the East Router on the MPLS interface to the datacenter. This by design allows all of the remote branch sites to use overlapping addresses, and is very common in larger networks. Routers that perform source NAT have two options: use the interface address and allocate a new source port, or use an IP address pool and allocate full IP addresses for each session. Either way, this allocated address only needs to be placed into metadata, as the existing packet address will be translated to waypoint addresses shortly. The egress SVR router will apply the source NAT.

3.6.1.6. Allocation of Ports

The next step is to allocate new ports for the SVR session. The ports being allocated must not be in use, and should not have been used recently to avoid any issues with middleboxes. See Section 4.2.

The range of ports that can be used may be site specific and tied to policies that exist in upstream firewalls or middleboxes. For these reasons, the actual pool of available addresses is provisioned on every SVR router. The East router has ports 8000 to 24000 available for both the source and destination ports. In this example we will allocate an even source port of 8000, and an odd destination port of 8001.

3.6.1.7. Session State and Metadata Construction

The router now has reached a point where it can forward the packet. It has valid network requirements, available peer paths, and has available SVR ports. The next step is to create and save all session state information for subsequent packet processing. A session UUID is created for end-to-end tracking of sessions. The table below refers to metadata TLVs and specific contents that are documented in Section 6.

Session State Table Entry

```
State Information & Mappings to Metadata Fields

              Metadata TLV                       |------TLV------|
Category       -Field              VALUE          Type   Len  Hdr
--------      ------------------   ----------------
Header                                                12
Header TLVs
              Security ID          1                 16      4    4
              Path Metrics                           26     10    4
               -Tx Color           5
               -Tx TimeValue       4200 MSecs
               -Rx Color           3
               -Rx TimeVlue        3950 MSecs
               -Drop               No
               -Prev Color Count   950 Packets
                                                            ---  ---
                            Total Header Length = 34 (26+8)  26    8

Payload TLVs
               Forward Context                       2      13    4
               - Source IP Addr     10.0.0.1
               - Dest IP Addr       172.15.11.23
               - Protocol           TCP
               - Source Port        6969
               - Dest Port          22
               Tenant Name          engineering       7     11    4
               Service Name         github           10      6    4
               UUID                 ABCDEFGHIJKLMNOP  6     16    4
               Source Router Name   East Router      14     11    4
               Source NAT Address   172.15.0.1       25      4    4
               Security Policy      NONE             15      4    4
               Peer Path                             19     22    4
               - Source Addr        172.15.0.1
               - Dest Addr          172.15.10.1
                                                            ---  ---
                         Total Payload Length = 119 (87+32)  87   32

                                 To West      Fr West
              Allocated Ports     Router       Router
               -Source Port        8000         8001
               -Dest Port          8001         8000
```

The required and optional metadata attributes that must be inserted in a first packet of a new sessions are defined in Section 4.3.1.

One optional metadata attribute is included in this example for the pathway metrics. This is documented in Section 6.3.7.

The order of the TLVs is arbitrary, but header TLVs must be before any payload TLVs. If a TLV is received that is unknown to a peer, it MUST ignore it. In this example, the header length including the two header TLVs is 34, and the 8 payload TLV's are 119 bytes long.

3.6.1.8. Encryption of Metadata

The next step is to encrypt the metadata block as defined in Section 4.4. In our example, our provisioned security definitions include AES256 for metadata encryption. AES has a 128 bit block size for all key lengths. In our example, the metadata payload TLVs are 119 bytes large. Padding will be added during encryption to make it 128 bytes (or 9 bytes of padding). In addition, to make the encrypted data stateless, we must also include a 16 byte initialization vector directly after the encrypted block. The resultant encrypted metadata block is 178 bytes and looks like this:

Metadata Block

```
+--------------+--------------+---------+----------------+
| Metadata     | Metadata     |Padding  | Initialization |
| Header )     | Payload TLVs |         |    Vector      |
| (Unecrypted) | Payload TLVs |         |    Vector      |
|  34 Bytes    | 119 Bytes    | 9 Bytes |  16 Bytes      |
+--------------+--------------+---------+----------------+
|<---Clear---->|<---Encrypted Portion-->|

|<----------------178 Byte Metadata Block--------------->|
```

3.6.1.9. Insert Metadata

The metadata block is inserted into the packet directly after the L4 Header. The total length of this specific metadata block is 178 bytes, 34 of which are header bytes and 119 for payload TLVs. If there is data in the payload portion of the IP Packet, the payload data is moved down to make room for the metadata. The packet structure will now look like:

Metadata Added

```
Packet with metadata inserted
+--------------------+--------------+----------+-----------+
| IP Header          | TCP Header   |Metadata  |  PAYLOAD  |
|    SRC=10.0.1.1    |   Sport=6969 |Block     |    Data   |
|    DST=172.15.11.23 |   Dport=22   |178 Bytes | (optional)|
+--------------------+--------------+----------+-----------+
```

The transport addresses in the packet are updated to use the selected peer path.

Transport Addresses Updated

```
  Final Transformed Packet with metadata inserted
  +---------------------+-------------+----------+-----------+
  | IP Header           | TCP Header  |Metadata  |  PAYLOAD  |
  |    SRC=172.15.0.1   |   Sport=8000 |Block     |    Data   |
  |    DST=172.15.10.1  |   Dport=8001 |178 Bytes | (optional)|
  +---------------------+-------------+----------+-----------+
```

3.6.1.10. Signing SVR Packet

The packet containing metadata is now signed with a HMAC signature (See Section 3.4). The HMAC signature is placed at the very end of the packet, extending the packet size by the signature's length. The IP header is excluded from the signature. The shared keys used for signing and verifying the authenticity of the packet is outside the scope of this document. In this case the HMAC is 16 bytes.

HMAC Signature Added

```
 Packet with metadata inserted
 +-------------------+-------------+----------+---------+-----+
 |IP Header          | TCP Header  |Encrypted | PAYLOAD | HMAC|
 |  SRC=172.15.0.1   |   Sport=8000 | metadata |   Data  | 16  |
 |  DST=172.15.10.1  |   Dport=8001 |          |         |Bytes|
 +-------------------+-------------+----------+---------+-----+
                     |                                  |
                     |<=========HMAC Signed Data========>|
```

3.6.1.11. Sending the First Packet

The packet length and checksum is corrected, and the packet is transmitted. The sending side will include the same exact metadata on every packet until a packet in the opposite direction (reverse direction) arrives with reverse metadata indicating a complete handshake. For TCP, the SYN packet contains metadata, and typically a SYN-ACK from the server side responds with metadata, and there is no further metadata inserted in a session.

```
     Client ---->    TCP SYN w/Metadata  ----> Server
     Server <---- TCP SYN-ACK w/Metadata <---- Server
```

For UDP, metadata can be inserted in packets until there is a reverse flow packet with metadata, except for unidirectional flows as noted in Section 3.5.7.

3.6.2. West First Packet Processing

If a packet arrives at the West Router having the West Routers Waypoint (interface address) as a destination address (i.e., the packet was sent to the router, and not to a destination beyond the router) the packet may likely contain metadata. When this occurs, the following steps are taken.

3.6.2.1. Verify Source Address is a Waypoint

Packets arriving on the routers must be verified to be valid before they are processed (see xref target="std_metadata_checking" />). These simple checks that can eliminate any potential attack vectors. If the packet fails authentication or validation the packet MAY be dropped or responded to with an ICMP Destination Unreachable packet.

In the example case we are using, there are only three source addresses that could be possible:

Possible Source Addresses

```
172.15.0.1       MPLS Peer Pathway
107.0.8.186      Internet Peer Pathway
169.254.231.106  LTE Peer Pathway
```

3.6.2.2. Verify Metadata Block

The very first and most efficient test is to verify that the metadata is present is to look for header magic number (see Section 4.6.1).

The next verification step is to check the HMAC signature (see Section 4.5.1). If the signature is invalid, the packet should be dropped and a security event noted. If valid, processing continues.

The unencrypted portions of the metadata header should be verified for reasonableness. The Header Length and Payload Length must be less than the metadata block size.

3.6.2.3. Parse Metadata and Save State and Translations

The next step is to decrypt the metadata (See Section 4.6.2.2). If there are any reasons why the metadata block can not be decrypted, or the decryption fails, the packet is dropped.

The payload TLVs can now be parsed and the necessary state and translations loaded into memory. If there is a failure to parse all TLV's, the packet is dropped.

Next the metadata block and HMAC signatures are removed from the packet.

3.6.2.4. Restore Addresses and Route Packet

The metadata information is used to restore the original context to the packet. The packet is then recursively processed exactly like the first packet described in Section 3.6.1 with a few differences. The Context, Tenant, Service, Security Policy and Session UUID strings are used from the metadata (as opposed to locally determining them) eliminating these steps. These are then used for applying policy and routing decisions locally. The end result is the packet may go through another SVR Peer Pathway or be delivered via standard networking techniques. In this example, the West Router delivers the packet to the Server LAN.

When the packet is forwarded to another SVR Peer, there are some differences. The Tenant, Service, Session UUID, Security Policy and the original 5-tuple addresses are all cloned. This provides consistent data across a multi-hop SVR network. It should be noted that the metadata must be decrypted at every SVR Router and then reencrypted because the Waypoint addresses are different for each selected peer pathway.

3.6.2.5. Detection of a Looping Session

Because every hop between SVR Routers utilizes the same session UUID, a looping first packet is easy to detect. There MUST never be two sessions with the same UUID. Any session that loops must be dropped. By detecting looping packets during the first packet transmitted, subsequent packets can be dropped on ingress by the SVR Router that detected the looping behavior. SVR routers must also decrement the TTL and operate in all ways like a traditional router to prevent looping packets that are not detected by SVR.

When a packet arrives with metadata after the metadata handshake has been completed, it is assumed to be an update and not classified as looping. Updates can be used to change any attribute, but most commonly to change a peer pathway for a session. See Section 5.1.

3.6.3. Return Packet Path Pre-Established

After processing the first forward packet at both East and West routers, both the East and West routers have established packet forwarding rules and translations for both directions. This means that eastbound rules and westbound rules are all established and installed. The router is thus capable now of recognizing 5-tuples in either direction and acting on the packets without consulting routing tables. This is known as fast path processing.

3.6.4. Sending Reverse Metadata

On a session-by-session basis, SVR Routers must know the status of a metadata handshake. If a packet for a session arrives and the metadata handshake is not complete, the SVR Router must insert metadata for the session. This will continue until there is verification that the SVR Peer has received the information. As stated previously, for TCP SYN this is normally the first reverse packet which is a TCP SYN/ACK. The purpose of reverse metadata is:

* To indicate to the sender that it can stop sending metadata. (Completion of the metadata handshake.)

* Provide backward information about the service for routing of future instances.

In this example, the reverse metadata includes:

Reverse Metadata Response

```
          Reverse Metadata Response
State Information & Mappings to Metadata Fields

              Metadata TLV                          |------TLV------|
Category       -Field               VALUE            Type   Len  Hdr
--------      ------------------   ----------------
Header                                                  12
Header TLVs
              Security ID          1                   16     4    4
              Path Metrics                             26    10    4
               -Tx Color           3
               -Tx TimeValue       4100 MSecs
               -Rx Color           5
               -Rx TimeVlue        4050 MSecs
               -Drop               No
               -Prev Color Count   1950 Packets
                                                            ---  ---
                           Total Header Length = 34 (26+8)   26    8

Payload TLVs
               Reverse Context                         4     13    4
               - Source IP Addr     172.15.0.1
               - Dest IP Addr       172.15.11.23
               - Protocol           TCP
               - Source Port        7891
               - Dest Port          6969
               Peer Path                               19    22    4
               - Source Addr        172.15.10.1
               - Dest Addr          172.15.0.1
                                                            ---  ---
                          Total Payload Length = 43 (35+8)  35     8

                                    To East    From East
               Allocated Ports      Router      Router
               - Source Port         8001        8000
               - Dest Port           8000        8001
```

See Section 4.3 for required and optional TLVs in reverse metadata.

One optional metadata attribute is included in this example for the pathway metrics. This is documented in Section 6.3.7.

One of the outstanding benefits of SVR is the complete tracking end-to-end of sessions. In this example, the metadata state located in the SVR router contains all addresses used. The forward context provides the egress SVR router with the addresses being used pre-NAT, and the source NAT information. The reverse context would likewise supply the ingress SVR destination NAT addresses. Also knowing the waypoint addresses used along with the ports used provides a complete end-to-end visibility of each session.

This metadata will be encrypted, inserted, and an HMAC checksum will be computed and attached as per the previous example. The reverse packet in this example will have 34 bytes of header data, and 43 bytes of payload data, 5 bytes of padding, and a 16 byte initialization vector resulting in a metadata block that is 98 bytes long.

3.6.5. Subsequent Packet Processing

As soon as an SVR peer receives a packet of a session from another SVR peer and there is no metadata, the SVR Handshake is complete, and it can stop sending metadata. This work for both the East Router and the West Router. Both will transmit metadata until they receive a packet without metadata.

3.6.6. Session Termination

No metadata is sent upon normal session termination. The router can monitor the TCP state machine and have a guard timer after seeing a FIN/ACK or RST exchange. After the guard timer, the session can be removed from the system. If a new session arrives during this period (a TCP SYN), then it will cause immediate termination of the existing session. In addition, all protocols also have an associated inactivity timeout, after which the session gets terminated if no packets flow in either direction. Should an existing session send a packet after the inactivity timeout, it will be processed as a new session.

3.6.7. Unidirectional/Asymmetric Flows

When there are unidirectional flows, or path asymmetry (e.g. TCP sequence numbers advance with no reverse packets observed), and there is end-to-end communication, one can stop sending metadata. For UDP asymmetry, the sending router will send a maximum of 11 packets with metadata; if no reverse packets are seen during that time, the receiving peer router generates and sends a disable metadata packet to the originating router to complete the metadata handshake.

3.6.8. Multi-Hop Session Ladder Diagram

The diagram below shows a typical normal TCP session flowing between a client and server through routers in a network.

Ladder Diagram for Session Initiation with Metadata:

```
     Client . . . . . . . . . . . . . . . . . . . . . . . . Server
       |                                                     |
       +         RouterA       RouterB      RouterC          |
       |            |             |            |             |
       +---SYN----->|             |            |             |
       |            |--SYN[MD1]-->|            |             |
       |            |             |--SYN[MD2]->|             |
       |            |             |            |--SYN------->|
       |            |             |            |             |
       |            |             |            |<--SYN/ACK-|
       |            |             |<--SYN/ACK--|             |
       |            |<--SYN/ACK---|   [RMD2]   |             |
       |<--SYN/ACK--|    [RMD1]   |            |             |
       |            |             |            |             |
       |            |             |            |             |
       |<===== Session Packets Flow with No Metadata =====>|
```

Note that each router constructs metadata for the next chosen peer in the routed pathway as depicted by metadata 1 [MD1] and metadata 2 [MD2] in the above diagram. Upon receipt of first reverse packet, reverse metadata [RMD2] and [RMD1] is inserted. Each router allocates its own transport addresses (waypoints) for each session. The context, service name, tenant name, and session UUID are sent unchanged between all routers, and can be used for determining routing policies to apply. The session UUID is the same in MD1, MD2, RMD1, and RMD2 in the above diagram.

Likewise, the diagram below shows a session teardown sequence for a typical TCP session.

Ladder Diagram for Session Teardown Metadata:

```
     Client . . . . . . . . . . . . . . . . . . . . . . . Server
       |                                                    |
       +         RouterA       RouterB      RouterC         |
       |            |             |            |            |
       +---FIN----->|             |            |            |
       |            |-----FIN---->|            |            |
       |            |             |----FIN---->|            |
       |            |             |            |-----FIN-->|
       |            |             |            |            |
       |            |             |            |<--FIN/ACK-|
       |            |             |<--FIN/ACK--|            |
       |            |<--FIN/ACK---|            |            |
       |<--FIN/ACK--|             |            |            |
       |            |             |            |            |
       |            |             |            |            |
```

No metadata is sent or required when sessions terminate. Each router keeps its state information for a programmed length of time in case a FIN/ACK is delayed or dropped, then the state information is removed.

4. SVR Protocol Definition

4.1. SVR Session Definitions and Types

SVR implementations MUST support TCP, UDP, and ICMP. SVR implementations SHOULD support UDP Unicast. Sessions are characterized by having an initial first packet that is a unique to an SVR router. Often this is described as a unique 5-tuples as seen by the router. Sessions start when the first packet is processed, and end when either the L4 protocol indicates the session is completed (TCP FIN/FIN ACK) or there has been no activity for a length of time (UDP, ICMP, UDP Unicast, point-to-point ethernet).

SVR is always OPTIONAL. SVR implementations can choose when to use SVR on a session-by-session basis. SVR implementations MUST support non-SVR traffic.

4.2. SVR Metadata Insertion

4.2.1. Metadata Packet Location

SVR implementations MUST insert metadata into packets directly after the L4 header, even if the resulting increase in packet size would cause the packet to require fragmentation. For Ethernet point-to-point and ICMP error messages, IP Headers and L4 headers MUST be created, and if associated with an existing session MUST share the exact transport 5-tuples (SVR Waypoints and Ports) as the session the ICMP error message relates to. The metadata MUST be in the very first packet of a new session (TCP or UDP bidirectional flow) to have any role in path selection or security. Metadata SHALL be sent in any subsequent packet in any direction to change or update the networking requirements. The metadata is inserted into the payload portion of a packet to guarantee it makes it unchanged through the network. Packet lengths and checksums MUST be adjusted accordingly. TCP sequence numbers MUST NOT be adjusted.

4.2.2. Metadata Prerequisites

A prerequisite for SVR metadata insertion is that a Peer Pathway MUST be selected relating to a specific session. This is similar to choosing a tunnel between two networks. This Peer Pathway has IP addresses on either side (Waypoint Addresses), and these addresses will always be the transport IP addresses for packets containing SVR metadata.

4.2.3. Metadata Port Allocation

The SVR peer originating the session (client side) MUST allocate both source and destination ports. The ingress side MUST choose even ports for local (source port) and odd ports for remote (destination port) This provides total uniqueness between any two peers, with no negotiation or collision possibilities. The range of ports to use for allocation is provisioned. Ports in use MUST be excluded from allocation. Ports MUST be unallocated when session state is removed. Ports MUST have a 60 second guard time before being reallocated

4.2.4. Metadata on Idle Session

SVR implementations MAY need to send metadata to a peer at a time when there are no existing packets. In these cases an IP packet MUST be created and inserted into the appropriate existing session with an indication the packet should be dropped. See Section 5.2 for an example. The packet MUST be processed, interpreted, and dropped by the directly adjacent peer and not forwarded to any other SVR peer.

4.2.5. Metadata Packet Structure

```
Existing IP Packet with metadata inserted
+------------------+-----------------+---------+----------+----+
| Existing IP Hdr  | Existing L4 Hdr |Metadata | PAYLOAD  |HMAC|
|   Source IP Addr |   Source Port   |Block    |   Data   |    |
|   Dest IP Addr   |   Dest Port     |         |(optional)|    |
+------------------+-----------------+---------+----------+----+

GeneratedIP Packet with metadata inserted
+-------------------+------------------+---------+----+
| Created  IP Hdr   | Created L4 Hdr   |Metadata |HMAC|
|   Source IP Addr  |   Source Port    |Block    |    |
|   Dest IP Addr    |   Dest Port      |         |    |
+-------------------+------------------+---------+----+

ICMP Packet with metadata inserted
+-----------------+----------------+----------+--------+----+
| Created IP Hdr  |Created UDP Hdr |Metadata  |  ICMP  |HMAC|
|   Source IP Addr|   Source Port  |Block     |  MSG   |    |
|   Dest IP Addr  |   Dest Port    |          |        |    |
+-----------------+----------------+----------+--------+----+

Ethernet Packet with metadata inserted
+-----------------+----------------+---------+---------+----+
| Created IP Hdr  |Created UDP Hdr |Metadata | Ethernet|HMAC|
|   Source IP Addr|   Source Port  |Block    | MSG     |    |
|   Dest IP Addr  |   Dest Port    |         |         |    |
+-----------------+----------------+---------+---------+----+
```

If UDP protocol, the UDP Header MUST be updated to have the correct packet length.

The Layer 4 header (TCP/UDP) MUST have its checksum recalculated per the appropriate procedures.

The IP Packet length field MUST be updated to reflect the number of bytes added for the metadata block AND the HMAC signature.

The IP Header Checksum MUST be updated after the IP Packet length is adjusted.

If TCP protocol, the TCP Sequence numbers MUST NOT be changed.

4.2.6. Prevention of False Positives

Metadata is sent inside the payload portion of TCP and UDP packets. Given that no byte sequence is truly unique in the payload of a packet, in the scenario where the original payload after the L4 header contained the same byte sequence as the SVR magic number, false positive logic is enacted on the packet. This guarantees downstream SVR routers will not confuse metadata magic number signatures.

False positives SHALL NOT occur when first packets are processed, since valid metadata will always be inserted regardless of the contents of the first 8 bytes of the payload. False positive can only occur during existing valid SVR sessions between peers.

To implement false positive logic, SVR implementations MUST insert an empty metadata header (12 byte header with 0 TLVs). This creates a contract with downstream SVR routers that if the magic number is present, there MUST be valid metadata that requires processing and removal.

The structure of a false positive metadata includes just a header of length 12 bytes, with zero header TLVs and zero payload TLVs. The SVR router receiving a packet with false positive metadata will strip out the metadata header and any TLV's as is normally expected. The inserted metadata header has no TLV's and is not encrypted.

Metadata Location

```
Received Midstream SVR Packet matching SVR Magic Number
+-------+--------+-----------------------+
|IP Hdr | L4 Hdr |0x4c48dbc6ddf6670c ..... |
+-------+--------+-----------------------+

Midstream SVR Packet with False Positive metadata inserted
+--------+--------+--------+-------------------------+
| IP Hdr | L4 Hdr |Metadata| 0x4c48dbc6ddf6670c ...... |
|        |        |  HDR   |                         |
+--------+--------+--------+-------------------------+
```

Insertion of header or payload TLV's is OPTIONAL and at the discretion of the implementation. If adding TLV's, standard procedures MUST be applied including encryption if payload TLV's are added.

4.2.7. TCP to UDP Transformation

TCP to UDP transformation is required when a middlebox blocks certain TCP packets that contain metadata. SVR implementations typically test Peer Pathways to ensure metadata insertion into TCP SYN packets will pass through any middleboxes. If TCP SYN packets with metadata are dropped by a middle box, then TCP packets are transformed to UDP for SVR processing, and restored when exiting SVR processing. The steps to transform TCP to UDP are:

The protocol field in the IP header MUST be changed from 0x06 (TCP) to 0x11(UDP).

The UDP checksum will write over the sequence number. To save the sequence number, it is copied to the 32 bit checksum/urgent pointer location of the TCP header.

To positively communicate that TCP to UDP transformation has occurred, one must add TLV 12 to the metadata being transmitted. See Section 6.4.9.

The UDP transformation is for every packet in a session, not just the packets with metadata. The restoration process is depicted in Section 4.6.3.

4.3. Required and Optional TLVs

4.3.1. New IP Sessions TLVs

The metadata TLVs that MUST be inserted in a first forward metadata packet of a new sessions include:

- *Header: Security Identifier: see Section 6.3.2.
- *Payload: Forward Context: see Section 6.4.1, Section 6.4.2.
- *Payload: Tenant Name: see Section 6.4.6.
- *Payload: Service Name: see Section 6.4.7.
- *Payload: Session UUID: see Section 6.4.5.
- *Payload: Source Router Name: see Section 6.4.10.
- *Payload: Security Policy: see Section 6.4.11.
- *Payload: Peer Pathway ID: see Section 6.4.12.

Optional metadata TLV's that MAY be included in forward metadata are:

- *Header: Patch Metrics: see Section 6.3.7.
- *Payload: Session Encrypted: see Section 6.4.8.
- *Payload: TCP Syn Packet: see Section 6.4.9.
- *Payload: IPv4 Source NAT Address: see Section 6.4.13.

The order of the TLVs is arbitrary, but header TLVs must be before any payload TLVs. If a TLV is received that is unknown to a peer, it MUST ignore it.

The metadata TLVs that MUST be inserted in a first reverse packet of a new sessions include:

* Header: Security Identifier: see Section 6.3.2.

* Payload: Reverse Context: see Section 6.4.3, Section 6.4.4.

* Payload: Peer Pathway ID: see Section 6.4.12.

Optional metadata TLV's that MAY be included reverse metadata are:

* Payload: Patch Metrics: see Section 6.3.7.

4.3.2. ICMP TLVs

The metadata TLVs that MUST be inserted when returning an ICMP Error include:

* Header: ICMP Error Location Address: see Section 6.3.4, Section 6.3.5.

Optional metadata TLV's that MAY be included reverse metadata are:

* Header: Patch Metrics: see Section 6.3.7.

4.4. Metadata Encryption

Encryption of metadata utilizes block mode ciphers. Cipher's MUST have a consistent block size. The cipher to use and its block size MUST be provisioned and communicated to peers in advance. The provisioning methodology is outside the scope of this document. The keys, and key rotation are also outside the scope of this document. When data is encrypted with block mode ciphers, the block will be padded with zeros (0x0's) to equal an increment of the block size used by the cipher. An initialization vector allows the decryption to be performed without any state.

Metadata Block

```
      Cipher      Block Size        IV Size
     -------   -----------------     -------
      AES256   128 Bits(16 Bytes)    16 Bytes
      AES128   128 Bits(16 Bytes)    16 Bytes

   +----------+--------+---------+--------+----------------+
   | Metadata | Header | Payload |Padding | Initialization |
   | Header   | TLVs   | TLVs    |        |    Vector      |
   +----------+--------+---------+--------+----------------+
   |<------Clear------>|<-- Encrypted --->|

   |<---------------------- Metadata Block ---------------->|
```

The padding can be computed as the length of the metadata payload TLVs MOD block size.

4.5. SVR Packet Authentication

4.5.1. HMAC Signatures

Through provisioning (outside the scope of this document), an SVR Authority MUST define if HMAC signatures are to be used. An SVR Authority MUST also define if Time Based HMAC is to be used. AN SVR Authority MUST determine if ALL packets are signed, or just packets containing metadata. Due to the possibility of replay attacks, it is RECOMMENDED that Time Based HMAC signatures be used on ALL SVR packets. Key distribution to support HMAC signatures is outside the scope of this document.

SVR Peers SHOULD sign all packets with HMAC signatures defined in [RFC2104]. When present there MUST be only one HMAC signature in an IP packet even if it fragments across multiple physical IP packets. Time-based HMAC signatures are RECOMMENDED. For time-based HMAC signatures, SVR routers append the current time since epoch (measured in seconds) divided by 2 to the data being signed. SVR routers MUST have clocks synchronized accurately. Methods for synchronizing clocks and measuring any differences or drifts are outside the scope of this document. Minimally NTP [RFC5905] should be implemented. In cases where the current time cannot be relied on, one may need to disable the time based HMAC and use a standard HMAC, but this is NOT RECOMMENDED.

The HMAC signature is always added to the very end of a packet. The size of the HMAC signature depends on which signature is used. Well known HMAC types are used with SVR including SHA1, SHA256-128, and SHA256.

```
SVR Packet with metadata inserted
+-----------+-------------+---------+---------+-------+
|IP Header  |  L4 Header  |Metadata | PAYLOAD | HMAC  |
|           |             |         |(optional)|      |
+-----------+-------------+---------+---------+-------+
            |                                 |
            |<======= HMAC Signed Data ========>|

Subsequent SVR Packet
+-----------+-------------+---------+-------+
|IP Header  |  L4 Header  |Payload  | HMAC  |
|           |             |         |       |
+-----------+-------------+---------+-------+
            |                       |
            |<== HMAC Signed Data ==>|


   HMAC TYPE          LENGTH OF SIGNATURE
  ------------------  ----------------------
     SHA1             20 Bytes
     SHA256-128       16 Bytes
     SHA256           32 Bytes
```

4.5.2. HMAC Verification

If HMAC signatures are present in an SVR implementation, SVR implementations MUST verify and remove the signature. Verification provides both authentication of the SVR router that sent the packet, and integrity that the packet has not been modified in any way intentionally, or through transmission errors between two SVR routers.

Through provisioning (outside the scope of this document), an SVR Authority MUST define if HMAC signatures are present. An SVR Authority MUST also define if Time Based HMAC is to be used. AN SVR Authority MUST determine if ALL packets are signed, or just packets containing metadata. Due to the possibility of replay attacks, it is RECOMMENDED that Time Based HMAC signatures be used on ALL SVR packets. Key distribution to support HMAC signatures is outside the scope of this document.

To verify the HMAC signature, a new signature is generated on the packet and bytewise compared to the signature transmitted in the packet.

```
SVR Packet with HMAC Signature
+-----------+-------------+----------+-------+
|IP Header  |  L4 Header  | PAYLOAD  | HMAC  |
|           |             |(optional)|       |
+-----------+-------------+----------+-------+
            |                        |
            |<== Signed Data ========>|


SVR Packet with HMAC Signature removed
+-----------+-------------+----------+
|IP Header  |  L4 Header  | PAYLOAD  |
|           |             |(optional)|
+-----------+-------------+----------+
```

For efficiency reasons, when verifying an Time Based HMAC signature, implementers SHOULD compute the HMAC on the packet (not including the IP header) and save the preliminary result. Then try updating the HMAC signature with the current window value. If this fails to match the signature, one must try updating the preliminary result using the next time window by adding 2 seconds (or previous by subtracting 2). If the time window is determined to be the next time window; it will remain that way for all packets received from a particular peer until it advances with clock time. Keeping an active time window per peer can make this process much more efficient.

If the signature does not match after checking adjacent time windows, then the packet is dropped and a security event noted.

If the signature matches exactly the signature in the packet, then the packet has been authenticated as being sent by the previous SVR router, and assured that the packets integrity between the two routers is good. The HMAC signature MUST be removed from the packet.

The IP Packet length field MUST be updated to reflect the number of bytes removed.

The IP Header Checksum MUST be updated after the IP Packet length is adjusted.

4.6. Processing SVR Packets with Potential Metadata

Routers MUST process SVR traffic and non-SVR traffic. SVR Routers MUST keep track of sessions that are using SVR. Only sessions setup with SVR may use the procedures described below. Traffic that is using SVR will always originate and terminate on Waypoint addresses (known peer pathways). This provides efficient separation of non-SVR traffic and SVR traffic.

Packets received on known Peer Pathways MUST be assumed to either have metadata or be packets associated with existing SVR sessions..

4.6.1. Detection of Potential Metadata in Packets

Any packet could arrive at any time with metadata. DPI MUST be used to scan for the presence of metadata on every packet. Metadata MAY be expected and required for first packet processing, and the absence of metadata will result in dropped packets.

The HMAC verification step (defined above) MUST be performed prior to performing any other metadata verification steps. This prevents attacks by modifying packet on the wire.

If the first 8 bytes of the payload (TCP or UDP) exactly matches the SVR magic number (0x4c48dbc6ddf6670c) it indicates that packet MUST have metadata. If the first 8 bytes do not match, the packet does not contain metadata. If metadata is not present the packet SHOULD be routed if part of an existing session (See Section 4.6.4). If not part of an existing session the packet MUST be dropped and a security event noted.

4.6.2. Verification of Metadata in Packets

4.6.2.1. TLV Parsing

The metadata header is parsed (see Section 6.1). If the header length and payload length are both zero, the metadata is simply removed and the packet is forwarded. Please see Section 4.2.6 for description of false positive metadata header insertion.. The next step is to walk the header TLV's to ensure they are reasonable. If the payload length is zero, then the metadata can be accepted and processed. Decryption of metadata is only required when there are payload TLV's.

If a TLV is sent that is unknown to the implementation, the TLV should be skipped and the TLV MUST not be forwarded.

If the metadata TLVs are not reasonable, the packet MUST be dropped and security events noted.

4.6.2.2. Decryption of Metadata Blocks

If the peers have been provisioned to encrypt metadata with a specific cipher AND the payload length in the header is non-zero, then the SVR implementation MUST assume that an encrypted metadata block was transmitted.

To decrypt the encrypted metadata block, an SVR implementation MUST have the pre-provisioned Cipher, block size, and initialization vector size. Once these are known, it is possible based on the payload length in the metadata header to determine the exact structure of the packet, and how to decrypt it.

Encrypted Metadata Block

```
     Known in advice: Cipher, Block Size, IV size
     From Metadata Header: Payload TLV size

+----------+--------+-------+-------+---------------+--~~~
| Metadata | Header |Payload|Padding| Initialization | Rest...
| Header   | TLVs   |TLVs   |       |    Vector (IV) | of  ...
|          |        |       |       |                | Pkt ...
+----------+--------+-------+-------+---------------+--~~~
|<------Clear------>|<- Encrypted ->|

|<------------------ Metadata Block ---------------->|
```

The padding is equal to the payload length from the header MOD cipher block size. The "block" is then decrypted assuming that the IV size bytes following the "block" is the Initialization vector.

If the decryption fails, then the packet MUST be assumed invalid and dropped. When this happens a security event is noted.

After the decryption succeeds, the payload TLV's MUST be reviewed for reasonableness and completeness. See Section 4.3 for minimum required TLV's. If there are insufficient TLV's present for the SVR implementation, the packets MUST be dropped and errors noted.

After review of the TLV's, the metadata is considered valid and accepted by the SVR implementation. The metadata block is removed from the packet, and the IP header length and checksum MUST be corrected. The packet signatures and decryption provide a very high degree of assurance that the metadata is authentic and has integrity.

4.6.3. UDP to TCP Transformation

If the received metadata block contains a TCP SYN Packet TLV (see Section 6.4.9) then the following procedures MUST be performed on EVERY packet of the session. This also signals to the SVR Router that packets flowing in the opposite direction MUST also be UDP transformed. See Section 4.2.7. The steps performed are:

The protocol field in the IP header MUST be changed from 0x11 (UDP) to 0x06 (TCP).

Copy the 32 bit integer in the checksum/urgent pointer location of the TCP header to the sequence number, effectively restoring it.

The TCP Checksum MUST be recalculated.

4.6.4. SVR Session Packets

Any packet that is has a source and destination IP address that maps to a Peer Pathway is an SVR packet. SVR Packets that do not have metadata are SVR session packets. Each of these MUST have corresponding known session state. If no session state exists, these packets MUST be dropped, or there must be an attempt to restore session state (see Section 2.10).

Packets ingressing to a peer pathway that are part of existing SVR sessions that do not contain metadata MUST be translated (all 5-tuples, bidirectionally). The source address MUST be replaced with the local Waypoint address associated with the peer pathway. The destination address MUST be replaced with the Waypoint of the SVR Peer chosen. The protocol either remains the same, or is modified if UDP Transformation is required (See Section 4.2.7). The source and destination port fields MUST be replaced with the ports allocated for this SVR session. For efficiency, implementors SHOULD save a single checksum delta as part of the session state because the address/protocol/port modifications will always be identical for each packet of a session.

Packets egressing from a peer pathway must have their addresses restored. SVR session state MUST contain the original packet context 5-tuples for every SVR session. The original Source IP Address MUST be restored. The original Destination IP Address MUST be restored. The original protocol must be restored, and if it is changes from UDP to TCP then one MUST follow the procedures defined in Section 4.6.3. The source port MUST be restored. The destination port MUST be restored.

4.6.5. Tenant/Service Overview

A provisioned SVR Policy SHOULD include both a tenant and service. Absence of a applicable SVR policy SHOULD prevent SVR sessions from being established. Traditional IP routing can be used when SVR policies do not apply.

4.6.5.1. Interpretation of the Service

Services are textual names for sets of CIDR blocks, protocols, and ports. Services map directly to our human understanding of a network use case. Examples include "Zoom" or "Office365".

Service Definition

```
svc_name
    protocol:TCP/UDP
    port ranges[]
    CIDR Blocks[]
```

When a packet arrives with metadata at an SVR Router the name of the service MUST be in first packet metadata.

When a first packet arrives without metadata, the service must be determined through a lookup of the IP destination address, port, and protocol. The resultant string becomes the service name. If this fails to result in a service, the name of the service can be determined by using application recognition techniques. These are omitted from this document, but include HTTP Request Analysis, TLS SNI, and Common names in certificates.

Services can have associated quality policies and security policies associated with them via provisioning. This is outside the scope of this document.

When egressing an SVR Peer Pathway, the service name can be used to route the packet to another SVR Peer, or to the final destination. If another SVR peer is chosen, the service name MUST be used as provided by the previous SVR peer. When exiting SVR and returning to traditional network routing, the textual service name MUST be resolved to an IP address. SVR supports several options:

Use Destination from Context: This is the default action. The original destination address will be restored and the packet will be forwarded to the destination.

Destination NAT Based on Local Configuration: Some provisioned service configurations locally (nearest the destination SVR router) will map the service to one or more local IP addresses through implementation of a destination NAT. This effectively becomes a load balancing algorithm to destination service instances, and is very useful in public clouds.

Resolve Destination using Local DNS: DNS resolution can be provisioned for services when the IP address is not known. This if often the case with services in private clouds.

Services SHOULD be provisioned to have lists of Tenants that are permitted to use a Service, and tenants that are denied using a service. These access controls are RECOMMENDED.

4.6.5.2. Determination and Interpretation of the Tenant

Tenant is a text string hierarchy delimited by periods. Tenants are logically similar to VLANs, CIDR block subnets, and Security Zones. The entire text string, including the full hierarchy is used to define a tenant, and for policy application, the tenant MAY match right to left in full segments (delimited by periods). The longest match will always be used (the most segments).

Tenants SHOULD be referenced and associated with Services to create a from-to vector. This has the benefits of associating ACLs directly with Destinations. A provisioned SVR Policy SHOULD include both a tenant and service. Absence of a applicable SVR policy prevents SVR sessions from being established. The deny by default approach is RECOMMENDED.

It is RECOMMENDED that a tenant be associated with physical interfaces and logical interfaces (VLANs) as a default for arriving sessions. CIDR block based tenants SHOULD override these defaults. Tenant definitions directly from clients that self assert their tenancy SHOULD override all other tenant definitions.

All network interface based tenant definitions are local to an SVR router. The tenant definitions on ingress to SVR MAY not match those on egress from SVR. This permits the use of different segmentation techniques in different networks.

4.6.6. Security Policy and Payload Encryption

If payload encryption is required, a Security Policy is used to describe all aspects of the agreed upon methods. Key management is outside the scope of this document. Using a semantically named Security Policy permits implementations to use whatever ciphers and techniques they wish, as long as they can be named.

5. Additional Metadata Exchanges and Use Cases

Metadata can be inserted and used to share network intent between routers. Below are examples for specific use cases. The metadata is not limited to these use cases, these are just illustrative.

5.1. Moving a Session

To change the pathway of a session between two routers, any SVR Router simply reinserts the metadata described in section Section 3.6.1.7 and transmits the packet on a different peer path, but retains the same Session UUID of the existing session that is being moved.

* Update its fast path forwarding tables to reflect the new IP addresses and ports (waypoints) for transport. All other aspects of the session remains the same. The presence of middle boxes means that routers on both sides must once again perform NATP detection and update real transmit addresses/ports to ensure that sessions will continue.

After 5 seconds the old path state entries can be removed. By keeping the old and new fast path entries during this 5 second transition, no packets in flight will be dropped. The diagram below shows the sequence for moving sessions around a failed mid-pathway router.

Ladder Diagram for Existing Session Reroute with Metadata:

```
                RTR-A      RTR-B      RTR-C      RTR-D
     Client . . . . . . . . . . . . . . . . . . . . . . . Server
       |          |          |          |          |          |
       |--PUSH--->|          |          |          |          |
       |          |--PUSH-------------->|          |          |
       |          |          |          |--PUSH--->|          |
       |          |          |          |          |--PUSH--->|
       |          |          |          |          |<---ACK---|
       |          |          |          |<---ACK---|          |
       |          |<--------------ACK---|          |          |
       |<---ACK---|          |          |          |          |
       |          |          |          |          |          |
       ......................RTR-C Fails.......................
       |--PUSH--->|          |          |          |          |
       |          |--PUSH--->|          |          |          |
       |          |  [MD1]   |          |          |          |
       |          |          |--PUSH[MD2]--------->|          |
       |          |          |          |          |--PUSH--->|
       |          |          |          |          |<--ACK----|
       |          |          |<-----ACK[RMD2]------|          |
       |          |<--ACK----|          |          |          |
       |<--ACK----|  [RMD1]  |          |          |          |
       |          |          |          |          |          |
       |<======== Session Packets Flow without Metadata =====>|
```

When router C fails, metadata [MD1,MD2] can be included in the very next packet being sent in either direction. Confirmation that the move was completed is confirmed with reverse metadata [RMD2, RMD1]. For established TCP sessions, this is either a PUSH (as shown) or an

ACK (Not shown). This can reestablish the SVR session state into a new router (Router B in this example) that previously did not have any involvement in the session. This technique can also be used to modify paths between two routers effectively moving TCP sessions from one transport (MPLS for example) to another (LTE). A session move can be initiated by any router at any time.

Ladder Diagram for Session Reroute Between Peers with Metadata:

```
              +-------+              +--------+
              |       +-----MPLS-----+        |
      Client--| Rtr-A |              | Rtr-B  +----Server
              |       +------LTE-----+        |
              +-------+              +--------+

     Client . . . . . . . . . . . . . . . . . . . . . . Server
       |                                                   |
       |         RouterA                    RouterC        |
       |            |                          |           |
       |---PUSH---->|                          |           |
       |            |---PUSH over MPLS-------->|           |
       |            |                          |---PUSH--->|
       ...............MPLS has Poor Quality ...............
       |            |                          |           |
       |---PUSH---->|                          |           |
       |            |---PUSH over LTE[MD]----->|           |
       |            |                          |---PUSH--->|
       |            |                          |<---ACK----|
       |            |<---ACK over LTE[RMD]-----|           |
       |<---ACK-----|                          |           |
       |            |                          |           |
       |<===== Session Packets Flow without Metadata =====>|
```

The diagram shows moving an active TCP session from one transport network to another by injecting metadata [MD] into any packet that is part of the transport in either direction. Reverse metadata is sent on any packet going in the reverse direction to confirm that the move was successful [RMD].

5.2. NAT Keep Alive

If an SVR Router determines there is one or more NATs on a peer pathway (See Section 2.4, the SVR Peer must maintain the NAT bindings for each active session by sending keep alive metadata in the direction of the NAT. For keep alive, SVR utilizes a packet that matches the L4 header of the idle session that includes metadata type 24 with the drop reason set to Keep Alive.

Ladder Diagram for NAT Keep Alive with Metadata:

```
              RTR-A       NAT       RTR-B
    Client . . . . . . . . . . . . . . . . . . Server
      |          |          |          |          |
      ..................Existing SVR Session......
      |--PUSH--->|          |          |          |
      |          |--PUSH--->|          |          |
      |          |          |---PUSH-->|          |
      |          |          |          |--PUSH--->|
      |          |          |          |<---ACK---|
      |          |          |<---ACK---|          |
      |          |<--PUSH---|          |          |
      |<--PUSH---|          |          |          |
      .........NO PACKETS EITHER DIRECTION FOR 20 SECS........
      |          |          |          |          |
      |          |--[MD1]-->|          |          |
      |          |          |--[MD1]-->|          |
      |          |          |          |          |
      .........NO PACKETS EITHER DIRECTION FOR 20 SECS........
      |          |          |          |          |
      |          |--[MD1]-->|          |          |
      |          |          |--[MD1]-->|          |
      |          |          |          |          |
```

The metadata attributes that MUST be inserted in a keep alive for existing packet sessions includes:

*Header: SVR Control Message: see Section 6.3.6.

Because there are only header attributes, encryption is not required.

5.3. Adaptive Encryption

Unlike a tunnel where all packets must be encrypted, each session in SVR is unique and independent. Most of the modern applications sessions are already using TLS or DTLS. SVR Routers have the capability of encrypting only sessions that are not already encrypted. Below is an example of adaptive encryption. With adaptive encryption, every session begins unencrypted. By analyzing the first 4 packets, the router can determine that encryption is required or not. If the fourth packet in a TLS Client hello message, encryption is NOT required. Any sequence of packets that does not indicate TLS or DTLS setup would immediately toggle encryption on.

Ladder Diagram of Adaptive Encryption with Client Hello:

```
     Client . . . . . . . . . . . . . . . . . . . . Server
       |                                            |
       +         RouterA            RouterB         |
       +---SYN----->|                  |            |
       |            |----SYN[MD1]----->|            |
       |            |                  |--SYN----->|
       |            |                  |<--SYN/ACK-|
       |            |<----SYN/ACK------|            |
       |<--SYN/ACK--|     [RMD1]       |            |
       |---ACK----->|                  |            |
       |            |------ACK-------->|            |
       |            |                  |--ACK----->|
       |--Client--->|                  |            |
       |  Hello     |<== ENCRYPTION===>|            |
       |            |   Not Required   |            |
       |            |                  |            |
       |            |-----Client------>|            |
       |            |      Hello       |--Client-->|
       |            |                  |            |
```

If the fourth packet is not an indication that encryption will be performed by the transport layer, then the ingress SVR Routers must encrypt and the egress SVR router must decrypt the session bidirectionally. This ensures that any data between the SVR Routers is encrypted.

Ladder Diagram of Adaptive Encryption with data:

```
     Client . . . . . . . . .  . . . . . . . . Server
       |                                        |
       +         RouterA       RouterB          |
       +---SYN----->|              |            |
       |            |--SYN[MD1]--->|            |
       |            |              |--SYN----->|
       |            |              |<--SYN/ACK-|
       |            |<--SYN/ACK----|            |
       |<--SYN/ACK--|    [RMD1]    |            |
       |---ACK----->|              |            |
       |            |----ACK------>|            |
       |            |              |--ACK----->|
       |---Data---->|              |            |
       |            |<==ENCRYPT===>|            |
       |            |  Required    |            |
       |            |              |            |
       |            |--Encrypted-->|            |
       |            |   Data       |---Data--->|
```

Adaptive encryption is part of the security provisioning. Security policies are associated with services, and as such certain applications can mandate encryption; others may allow adaptive encryption, and still others may specify no encryption.

5.4. Packet Fragmentation

When a fragmented packet is presented to a SVR Router, the packet must be completely assembled to be processed. The SVR Router routes IP packets, and as all SVR actions require the entire packet. As such, the HMAC must be applied to the entire packet, and the entire packet must be routed as a whole. Each resulting fragment must be turned into an IP packet with 5-tuples that match the corresponding session to ingress and pass through an SVR. The SVR Router will simply use the same L4 header on all fragments from the session state table (peer pathway and transit ports). a time based HMAC signature is created for the entire packet and appended to the last fragment. Each fragment must also have metadata inserted that clearly identifies the fragment to the SVR routing peer.

Ladder Diagram Fragmented Packets:

```
  Client . . . . . . . . . . . . . . . . . . . . . . . . . Server
     |                                                     |
     |         RouterA                    RouterB          |
     |            |                          |             |
     |--Frag 1--->|                          |             |
     |--Frag 3--->|                          |             |
     |--Frag 2--->|                          |             |
     |        +---+----+                     |             |
     |        |Assemble|                     |             |
     |        +---+----+                     |             |
     |            |----Frag 1[L4/MD]-------->|             |
     |            |                          |             |
     |            |----Frag 2[L4/MD]-------->|             |
     |            |                          |             |
     |            |----Frag 3[L4/MD]-------->|             |
     |            |                     +--------+         |
     |            |                     |Assemble|         |
     |            |                     +--------+         |
     |            |                          |--Frag 1-->|
     |            |                          |--Frag 2-->|
     |            |                          |--Frag 3-->|
```

In the diagram above, Router A collects all the fragments 1 2, and 3. Reassembly is performed. Router A records two things from the inbound fragments: The Original ID, and the largest fragment size received. Router A then proceeds to send the jumbo packet by fragmenting it again, but this time sending each piece inside a packet with a newly created L4 which maps exactly to the peer pathway chosen with ports assigned from the session state table. The fragment size will be the lesser of the smallest MTU on the path OR the largest fragment seen, whichever is smaller. The Metadata header and header TLV's are not encrypted. The packet construction looks like this:

SVR Fragment Packet Layout

```
 Fragment 1
+-----+-----+----------+----------+---------+
|Peer |Peer | Metadata | Header   | First   |
|IP   |L4   | Header   | TLV-1,16 | Fragment|
|HDR  |HDR  | 12 Bytes | 22 Bytes |         |
+-----+-----+----------+----------+---------+

 Fragment 2
+-----+-----+----------+----------+---------+
|Peer |Peer | Metadata | Header   | Second  |
|IP   |L4   | Header   | TLV-1    | Fragment|
|HDR  |HDR  | 12 Bytes | 14 Bytes |         |
+-----+-----+----------+----------+---------+

 Fragment 3
+-----+-----+----------+----------+---------+----------+
|Peer |Peer | Metadata | Header   | Third   | PKT      |
|IP   |L4   | Header   | TLV-1    | Fragment| HMAC     |
|HDR  |HDR  | 12 Bytes | 14 Bytes |         | SIGNATURE|
+-----+-----+----------+----------+---------+----------+
```

The metadata type 1 inside the SVR fragment will have its own extended ID assigned. This allows a different number of fragments to be between router A and B than the Client and Server have. It also allows independent fragmentation by SVR should it be required. Router B will process the fragments from router A. Router B will look at its egress MTU size, and the largest fragment seen recorded by RouterA and transmitted in Metadata to determine the proper size fragments to send, and the packet is fragmented and sent.

There are no other metadata fields required. All information about the session state is tied to the 5-tuple peer pathway and transports ports.

The details on packet fragmentation are identical to what is standardly performed in IP fragmentation, exception for the full L4 headers and metadata insertion.

If a packet traversing an SVR needs to be fragmented by the router for an SVR segment for any reason, including the insertion of metadata, the initiating router inserts metadata on the first packet and duplicates the L4 header (either TCP or UDP) on subsequent fragments and inserts metadata. In this case the Largest Fragment Seen and Original ID field in the metadata is left blank.

Ladder Diagram Fragmented Packets:

```
Client . . . . . . . . . . . . . . . . . . . . . . . . . Server
  |                                                        |
  |         RouterA                     RouterB           |
  |            |                           |               |
  |--Lg Pkt--->|                           |               |
  |            |--------Frag 1[MD]------->|               |
  |            |                           |               |
  |            |----Frag 2[L4 Hdr|MD]---->|               |
  |            |                           |--Lg Pkt-->|
  |            |                           |               |
```

5.5. ICMP and SVR

There are two types of ICMP messages. There are messages associated with specific packet delivery network errors. This includes:

*Type 3: Destination Unreachable

*Type 11: Time Exceeded (TTL)

These messages have information from the packet that generated the error by including the IP header + 8 bytes in the ICMP message (See [RFC0792]. It is important to deliver the ICMP message back to the origin. For these ICMP messages, the router MUST determine what active session the ICMP message belongs to by parsing the IP header information inside the ICMP message. Once a session is found, the ICMP message is transported across the SVR and reverse metadata is applied by having its destination address changed to the waypoint addresses of the routers.

Metadata type 20 and 21 are used to send the source of the ICMP error backward through the networks. See Section 6.3.4 and Section 6.3.5 for information about these metadata formats. This repeats until the ICMP packet arrives at the initial SVR router. At this point the ICMP packet is recreated and the source address is changed to the address communicated through metadata type 20 and 21.

SVR Fragment Packet Layout

```
+------------+------------+----------------+--------------+
|  IP HEADER | UDP HEADER | Metadata 20/21 | ICMP Packet  |
+------------+------------+----------------+--------------+
```

ICMP over SVR for Network Failures

```
 Client . . . . . . . . . . . . . . . . . . . . . . . .No Network
   |                                                  Found
   |         RouterA                    RouterB          |
   |            |                          |             |
   |----PKT---->|                          |             |
   |            |------PKT[MD]------------>|             |
   |            |                          |<--ICMP------|
   |            |                          |  (Router B) |
   |            |<--UDP[ICMP[RMD]]---------|             |
   |<--ICMP-----|                          |             |
   | (Client)   |                          |             |
   |            |                          |             |
```

The first ICMP message is directed to Router B. Router B examines the ICMP error to find the session, and forwards backwards to the correct waypoint for Router A. Router A recreates the ICMP message, and sends to the Client. The address of where the error was detected is in

The second type of ICMP message is not related to any specific sessions. These types of messages include ICMP ECHO for example. These are always encapsulated as UDP, and a session is created for the ICMP message. The identifier field in ICMP and the IP addresses are used as the 5-tuple session key. This includes:

*Type 8:ECHO Request (Ping)

ICMP over SVR for Information

```
     Client . . . . . . . . . . . . . . . . . . . . . . . Target
       |                                                     |
       |             RouterA             RouterB             |
       |                |                   |                |
       |--ICMP ECHO---->|                   |                |
       |                |---UDP[ICMP ECHO]->|                |
       |                |       [MD1]       |                |
       |                |                   |---ICMP ECHO--->|
       |                |                   |<--ECHO RESP----|
       |                |<--UDP[ECHO RESP]--|                |
       |                |       [RMD1]      |                |
       |<--ECHO RESP----|                   |                |
```

The ICMP message creates a session on Router A directed towards Router B. Metadata [MD1] is inserted just like any UDP session to establish the return pathway for the response. Reverse metadata is inserted into the ECHO Response, effectively creating an ICMP session. Subsequent identical ICMP messages will utilize this path without metadata being inserted. This session state MUST be guarded with an inactivity timer and the state deleted.

6. Metadata Format and Composition

The format of metadata has both Header attributes as well as Payload attributes. Header attributes are always guaranteed to be unencrypted. These headers may be inspected by intermediate network elements but can't be changed. Header attributes do not have a forward or reverse direction. Header attributes are used for router and peer pathway controls.

Payload attributes optionally can be encrypted by the sender. Payload attributes are associated with sessions, and as such have a forward and reverse direction. For encryption, the pre-existing security association and key sharing is outside the scope of this document. Each SVR attribute defined will indicate whether it is a header attribute (unencrypted) or payload attribute (optionally encrypted). There are no attributes that can exist in both sections.

6.1. Metadata Header

The metadata header is shown below. A well-known "cookie" (0x4c48dbc6ddf6670c in network byte order byte order) is built into the header which is used in concert with contextual awareness of the packet itself to determine the presence of metadata within a packet. This is an eight-byte pattern that immediately follows the L4 header and is an indicator to a receiving router that a packet contains metadata. NOTE: Normal IP traffic will never have the Waypoint Address as its destination. If a packet arrives at a SVR Router Waypoint it has to have Metadata or be associated with an active SVR session. Please see Section 2.10 for a discussion of state recovery techniques.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                             Cookie                            +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|   Header Length       |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Header TLVs ...           |       Payload TLVs ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8

Cookie (8 bytes):

The fingerprint of metadata. This value is used to determine the existence of metadata within a packet.

Version (4 bits): Field representing the version of the metadata header. The current version of metadata is 0x1.

Header Length (12 bits): Length of the metadata header including any added Header TLV attributes that are guaranteed to be unencrypted. When there are no Header TLVs, the value Header Length is 12 Bytes or OxC.

Payload Length (2 bytes): Length of data following the metadata header, not including the size of the header. This data could be encrypted. The value of this field is the number of bytes in the Payload TLV's. If there are no TLV's the value is zero.

6.1.1. False Positives

Given that no byte sequence is truly unique in the payload of a packet, in the scenario where the original payload after the L4 header contained the same byte sequence as the cookie, false positive logic is enacted on the packet. If the metadata HMAC signature can't verify that the metadata is valid, then a false positive metadata header is added to the packet to indicate that the first eight bytes of the payload matches the cookie.

The structure of a false positive metadata includes just a header of length 12 bytes, with zero header TLVs and zero payload TLVs. The receiving side of a packet with false positive metadata will strip out the metadata header.

In the scenario where a router receives a false positive metadata header but intends to add metadata to the packet, the false positive metadata header is modified to contain the newly added attributes. Once attributes are added, the metadata header is no longer considered to be false positive.

6.1.2. Forward and Reverse Attributes

Payload metadata attributes may be valid in the forward direction, the reverse direction, or both. If not valid, it is ignored quietly by the receiving side.

6.2. TLVs for Attributes

All metadata attributes are expressed as Tag Length Values or TLV's. This includes Header and Payload TLVs. It is recommended that Payload TLVs be encrypted, but not mandatory. When debugging networks, or if mid-stream routers need to consult the TLV's, they can be transmitted in clear text. The entire metadata block is signed, and thus the integrity of the data can be verified. No midstream router or middlebox can modify any aspect of the metadata. Doing so will invalidate the signature, and the metadata will be dropped.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Variable Length Values .....                        |
\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 9

Type (2 bytes): Type of data that follows. Each of different Header and Payload TLV's are defined below.

Length (2 bytes): Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

6.3. Header Attributes

6.3.1. Fragment

When a packet is fragmented to insert metadata, a new fragmentation mechanism must be added to prevent fragmentation attacks and to support reassembly (which requires protocol and port information). If a packet is received that IS a fragment, and it must transit through a metadata signaled pathway, it must also have this metadata attached to properly bind the fragment with the correct session.

All fragments will have a metadata header and the fragment TLV added to the guaranteed unencrypted portion of the metadata header. If the original packet already has a metadata header on it, the fragment TLV will be added to it. See [RFC0791] for information about IP Fragmentation. For a detailed example of packet fragmentation in SVR please see Section 5.4

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 1           |           Length = 10         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Extended ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Original ID             |Flags|    Fragment Offset      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Largest Seen Fragment      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10

TLV: Type 1, Length 10.

Extended ID (4 bytes): Uniquely identifies a packet that is broken into fragments This ID is assigned by the SVR that is processing fragmented packets. IPv6 uses a 32-bit Extended ID, and IPv4 uses a 16 bit ID. We use the same algorithm for fragmenting packets for both IPv6 and IPv4, therefore we chose a 32-Bit Extended ID. .

Original ID (2 bytes): Original identification value of the L3 header of a received packet that is already fragmented.

Flags (3 bits): Field used for identifying fragment attributes. They are (in order, from most significant to least significant):

bit 0: Reserved; must be zero.

bit 1: Don't fragment (DF).

bit 2: More fragments (MF).

Fragment Offset (13 bits): Field associated with the number of eight-byte segments the fragment payload contains.

Largest Seen Fragment (2 bytes): Each SVR router keeps track of the largest fragment processed from each interface. This allows the router to make inferences about the MTU size when fragmenting packets in the opposite direction. This information is used along with a given egress network interface MTU to determine the fragment size of a reassembled packet.

6.3.2. Security Identifier

A versioning identifier used to determine which security key version should be used when handling features dealing with security and authenticity of a packet.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 16           |           Length = 4          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Security Key Version                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11

TLV: Type 16, Length 4.

Security Key Version (4 bytes): This is a four-byte security key version identifier. This is used to identify the algorithmic version used for metadata authentication and encryption.

6.3.3. Disable Forward Metadata

An indication that forward metadata should be disabled. This is sent in the reverse metadata to acknowledge receipt of the metadata. This is the second part of the metadata handshake.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 18           |        Length = 0             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12

TLV: Type 18, Length 0.

No other data is required. The specific session that is being referred to is looked up based on the 5-tuple address of the packet. See metadata handshake in Section 2.3.

6.3.4. IPv4 ICMP Error Location Address

This is exclusively used to implement ICMP messages that need to travel backwards through SVR pathways. See Section 5.5 for more information. The IPv4 address of the source of the ICMP message is placed into metadata. This metadata travels in the reverse direction backwards to the originating SVR, which restores the information and sends an ICMP message to the originator of the packet.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 20           |           Length = 4          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13

TLV: Type 20, Length 4.

Source Address (4 bytes): Original IPv4 source address of the originating router.

6.3.5. IPv6 ICMP Error Location Address

This is exclusively used to implement ICMP messages that need to travel backwards through SVR pathways. See Section 5.5 for more information. The IPv6 address of the source of the ICMP message is placed into metadata. This metadata travels in the reverse direction backwards to the originating SVR, which restores the information and sends an ICMP message to the originator of the packet.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 21           |           Length = 16         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                        Source Address                         +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14

TLV: Type 21, Length 16.

Source Address (16 bytes): Original IPv6 source address of the originating router.

6.3.6. SVR Control Message

The SVR Control Message is used for protocol specific purposes that are limited to a single peer pathway. This message is sent by an SVR router to a peer. This metadata is always sent in a UDP message originating by the SVR control plane.

Keep Alive: When an SVR peer is behind a NAT device and the SVR peer has active sessions, the SVR peer will generate a "Keep Alive" often enough (i.e., 20 seconds) to prevent the firewall from closing a pinhole. This message is generated completely by the SVR router, and directed to the SVR peer for a session. The UDP address and ports fields must exactly match the session that has been idle longer than the provisioned time.

Turn On Metadata: When a packet is received, and there is missing SVR Session State, the correction procedure may involve sending this request to a peer SVR router that has the information. Please see Section 2.10 for more information.

Turn Off Metadata: Disable Metadata on a specific 5-tuple. In certain cases, the SVR peer may continue so send metadata because there are no reverse flow packets or because metadata was enabled to recover from a loss of state. This message is not part of the normal metadata handshake and only has a scope of a single peer pathway.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type = 24            |           Length = 1          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Drop Reason  |
+-+-+-+-+-+-+-+-+
```

Figure 15

TLV: Type 24, Length 1.

Drop Reason (1 byte): Reason why this packet should be dropped.

*0 = Unknown. This value is reserved and used for backwards compatibility.

*1 = Keep Alive. A packet that is dropped by the receiving node. Used only to keep NAT pinholes alive on middleboxes.

*2 = Enable Metadata. Begin sending metadata on the peer pathway for the 5-tuple matching this control packet.

*3 = Disable Metadata. Stop sending metadata on the peer pathway for a 5-tuple matching this control packet.

6.3.7. Path Metrics

This metadata type is used to allows peers to measure and compute inline flow metrics for a specific peer pathway and a chosen subset of traffic. class. The flow metrics can include jitter, latency and packet loss. This is an optional metadata type.

When a peer sends this metadata, it provides the information for the period of time to the peer.

When a peer receives this metadata type 26, it responds with metadata type 26.

After several exchanges, each side can compute accurate path metrics for the traffic included. This metadata can be sent at any time, but is normally sent when metadata is being sent for other reasons. The metadata includes "colors" which represent blocks of packets. Packet loss and latency can be determined between routers using this in line method. Using colors to measure inline flow performance is outside the scope of this document. Please refer to [RFC8321]

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 26           |           Length = 10         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Tx Co |                Transmit TimeValue                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rx Co |                Received TimeValue                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|D|   Previous Rx Color Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16

TLV: Type 26, Length 10.

Transmit Color (4 bits): Current color of a transmitting node.

Transmit Time Value (28 bits): Current time value in milliseconds at time of marking. This time value represents the amount of time which has elapsed since the start of a transmit color.

Received Color (4 bits): Most recently received color from a remote node. This represents the color last received from a specific peer.

Receive Time Value (28 bits): Cached time value in milliseconds from adjacent node adjusted by the elapsed time between caching of the value and current time. This time value is associated with the received color.

Drop Bit (1 bit): Should this packet be dropped. This is required if a packet is being sent solely to measure quality on an otherwise idle link.

Previous Rx Color Count (15 bits): Number of packets received from the previous color block. This count is in reference to the color previous to the current RX color which is defined above.

6.4. Payload Attributes

Payload attributes are used for session establishment and SHOULD be encrypted to provide privacy. Encryption can be disabled for debugging.

6.4.1. Forward Context IPv4

The context contains a five-tuple associated with the original addresses, ports, and protocol of the packet. This is also known as the Forward Session Key.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             Type = 2          |           Length = 13         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          Source Address                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       Destination Address                     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Source Port           |      Destination Port         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Protocol    |
 +-+-+-+-+-+-+-+-+
```

Figure 17

TLV:
Type 2, Length 13.

Source Address (4 bytes): Original IPv4 source address of the packet.

Destination Address (4 bytes): Original IPv4 destination address of the packet.

Source Port (2 bytes): Original source port of the packet.

Destination Port (2 bytes): Original destination port of the packet.

Protocol (1 byte): Original protocol of the packet.

6.4.2. Forward Context IPv6

A five-tuple associated with the original addresses, ports, and protocol of the packet for IPv6.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 3            |           Length = 37         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                       Destination Address                     +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |        Destination Port       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Protocol    |
+-+-+-+-+-+-+-+-+
```

Figure 18

TLV: Type 3, Length 37.

Source Address (16 bytes): Original IPv6 source address of the packet.

Destination Address (16 bytes): Original IPv6 destination address of the packet.

Source Port (2 bytes): Original source port of the packet.

Destination Port (2 bytes): Original destination port of the packet.

Protocol (1 byte): Original protocol of the packet.

6.4.3. Reverse Context IPv4

Five-tuple associated with the egress (router) addresses, ports, and protocol of the packet. Forward context and reverse context session keys are not guaranteed to be symmetrical due to functions which apply source NAT, destination NAT, or both to a packet before leaving the router.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type = 4          |           Length = 13         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Source Port           |      Destination Port         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Protocol    |
+-+-+-+-+-+-+-+-+
```

Figure 19

TLV: Type 4, Length 13.

Source Address (4 bytes): Egress IPv4 source address of the packet.

Destination Address (4 bytes): Egress IPv4 destination address of the packet.

Source Port (2 bytes): Egress source port of the packet.

Destination Port (2 bytes): Egress destination port of the packet.

Protocol (1 byte): Original protocol of the packet.

6.4.4. Reverse Context IPv6

Five-tuple associated with the egress (router) addresses, ports, and protocol of the packet. Forward and reverse session keys are not guaranteed to be symmetrical due to functions which apply source NAT, destination NAT, or both to a packet before leaving the router.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 5            |          Length = 37          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                       Destination Address                     +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |        Destination Port       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Protocol    |
+-+-+-+-+-+-+-+-+
```

Figure 20

TLV: Type 5, Length 37.

Source Address (16 bytes): Egress IPv6 source address of the packet.

Destination Address (16 bytes): Egress IPv6 destination address of the packet.

Source Port (2 bytes): Egress source port of the packet.

Destination Port (2 bytes): Egress destination port of the packet.

Protocol (1 byte): Original protocol of the packet.

6.4.5. Session UUID

Unique identifier of a session. The UUID MUST be conformant to [RFC4122]This is assigned by the peer that is initiating a session. Once assigned, it is maintained through all participating routers end-to-end.

The UUID is used to track sessions across multiple routers. The UUID also can be used to detect a looping session. The UUID metadata field is required for all session establishment.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 6           |          Length = 16          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                             UUID                              +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21

TLV: Type 6, Length 16.

UUID (16 bytes): Unique identifier of a session.

6.4.6. Tenant Name

An alphanumeric ASCII string which dictates what tenancy the session belongs to.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 7           |       Length = variable       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Name (1 - n bytes) ....                       |
\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 22

TLV: Type 7, Length variable.

Name (variable length): The tenant name represented as a string.

6.4.7. Service Name

An alphanumeric string which dictates what service the session belongs to.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 10          |       Length = variable       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Service Name (1-n bytes) .....               |
\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 23

TLV: Type 10, Length variable.

Name (variable length): The service name represented as a string.

6.4.8. Session Encrypted

Indicates if the session is having its payload encrypted by the SVR router. This is different from having the metadata encrypted. The keys management and ciphers used are outside the scope of this document. The keys used for payload encryption may be different than the keys used for metadata encryption as the security associations are different. The keys selected will be based on the Tenant and Service metadata fields permitting end user specified cryptography.

This field is necessary because often traffic is already encrypted before arriving at an SVR router. Also in certain use cases, re-encryption may be required. This metadata TLV is always added when an SVR is going to encrypt the payload.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 11          |           Length = 0          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 24

TLV: Type 11, Length 0.

6.4.9. TCP SYN Packet

Indicates if the session is being converted from TCP to UDP to enable passing through middle boxes that are TCP session stateful. A SVR implementation must verify that metadata can be sent inside TCP packets through testing the Peer Pathway. If the data is blocked, then all TCP sessions must be converted to UDP sessions, and restored on the destination peer.

Although this may seem redundant with the Forward Context that also has the same originating protocol, this refers to a specific peer pathway. In a multi-hop network, the TCP conversion to UDP could occur at the second hop. It's important to restore the TCP session as soon as possible after passing through the obstructive middlebox.

When TCP to UDP conversion occurs, no bytes are changed other than the protocol value (TCP->UDP). Because the UDP message length and checksum sit directly on top of the TCP Sequence Number, the Sequence number is overwritten. The Sequence number is saved by copying it to the TCP Checksum. The Checksum is recalculated upon restoration of the packet. The packet integrity against bit loss or malicious activity is provided through the HMAC signature.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 12          |           Length = 0          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 25

TLV: Type 12, Length 0.

Note: This type does not contain any value as its existence in metadata indicates a value.

6.4.10. Source Router Name

An alphanumeric string which dictates which source router the packet is originating from. This attribute may be present in all forward metadata packets if needed.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 14          |       Length = variable       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Router Name (1-n bytes) ....                    |
\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 26

TLV: Type 14, Length variable.

Name (variable length): The router name represented as a string.

6.4.11. Security KEY

An alphanumeric string containing the session key to use from this packet onward for encryption/decryption.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 15          |       Length = variable       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        SESSION KEY                            |
\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 27

TLV: Type 15, Length variable.

KEY (variable length): The session key to use for encryption/ decryption for this packet and future packets in a session.

6.4.12. Peer Pathway ID

An ASCII string which dictates which router peer pathway has been chosen for a packet. This name is the hostname or IP address of the egress interface of the originating router. This can be used to determine the peer pathway used exactly when there may be multiple possibilities. This enables association of policies with specific paths.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 19          |       Length = variable       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Peer Pathway Name (1-n bytes) ....                 |
\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
```

Figure 28

TLV: Type 19, Length variable.

Name (variable length): The peer pathway name which is represented as a string.

6.4.13. IPv4 Source NAT Address

Routers may be provisioned to perform source NAT functions while routing packets. When a source NAT is performed by an SVR Peer, this metadata TLV MUST be included. When the far end router reconstructs the packet, it will use this address as the source address for packets exiting the SVR.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 25            |           Length = 4          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      IPv4 Source Nat Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 29

TLV: Type 25, Length 4.

Source Address (4 bytes): Source NAT address of the packet.

7. Security Considerations

7.1. HMAC Authentication

HMAC signatures are REQUIRED for the packets that contain metadata to guarantee the contents were not changed, and that the router sending it is known to the receiver. Any HMAC algorithm can be used, from SHA128, or SHA256 as long as both sides agree. HMAC is always performed on the layer 4 payload of the packet. The signature is placed at the end of the existing packet.

7.2. Replay Prevention

Optional HMAC signatures are RECOMMENDED for every packet. This prevents any mid-stream attempts to corrupt or impact sessions that are ongoing. This also helps detect and correct lost state at egress SVR routers. See Section 2.10. The signature must include all of the packet after Layer 4, and include a current time of day to prevent replay attacks. The signature is placed at the end of the existing packet.

Both the sending and receiving routers must agree on these optional HMAC signatures, details of which are outside the scope of this document.

7.3. Payload Encryption

Payload encryption can use AES-CBC-128 or AES-CBC-256 ciphers which can be configured. Since these are block-ciphers, the payload should be divisible by 16. If the actual payload length is divisible by 16, then the last 16 bytes will be all 0s. If the actual payload is not divisible by 16, then the remaining data will be padded and the last byte will indicate the length.

7.4. DDoS and Unexpected Traffic on Waypoint Addresses

Waypoint addresses could be addressed by any client at any time. IP packets that arrive on the router's interface will be processed with the assumption that they MUST contain metadata OR be part of an existing established routing protocol.

Routers will only accept metadata from routers that they are provisioned to speak with. As such an ACL on incoming source addresses is limited to routers provisioned to communicate. All other packets are dropped.

When a packet is received the "cookie" in the metadata header is reviewed first. If the cookie isn't correct, the packet is dropped.

The HMAC signature is checked. If the signature validates, the packet is assumed to be good, and processing continues. If the HMAC fails, the packet is dropped.

These methods prevent distributed denial of service attacks on the waypoint addresses of routers.

8. IANA Considerations

This document does not require any IANA involvement.

9. Acknowledgements

The authors would like to thank Anya Yungelson, and Scott McCulley for their input into this document.

The authors would like to thank Tony Li for his extensive support and help with all aspects of this document.

The authors want to thank Ron Bonica, Kireeti Kompella, and other IETFers at Juniper Networks for their support and guidance.

10. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792] Postel, J., "Internet Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI

10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC5580] Tschofenig, H., Ed., Adrangi, F., Jones, M., Lior, A., and B. Aboba, "Carrying Location Objects in RADIUS and Diameter", RFC 5580, DOI 10.17487/RFC5580, August 2009, <https://www.rfc-editor.org/info/rfc5580>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6062] Perreault, S., Ed. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", RFC 6062, DOI 10.17487/RFC6062, November 2010, <https://www.rfc-editor.org/info/rfc6062>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8321]

Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFC8489]
Petit-Huguenin, M., Salgueiro, G., Rosenberg, J., Wing, D., Mahy, R., and P. Matthews, "Session Traversal Utilities for NAT (STUN)", RFC 8489, DOI 10.17487/RFC8489, February 2020, <https://www.rfc-editor.org/info/rfc8489>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

Authors' Addresses

Abilash Menon
Juniper Networks
10 Technology Part Dr.
Westford, MA 01886
United States of America

Email: abilashm@juniper.net

Patrick MeLampy
Juniper Networks
10 Technology Part Dr.
Westford, MA 01886
United States of America

Email: pmelampy@juniper.net

Michael Baj
Juniper Networks
10 Technology Part Dr.
Westford, MA 01886
United States of America

Email: mbaj@juniper.net

Patrick Timmons
Juniper Networks
10 Technology Part Dr.
Westford, MA 01886
United States of America

Email: ptimmons@juniper.net

Hadriel Kaplan
Juniper Networks
10 Technology Park Dr.

Westford, MA 01886
United States of America

Email: hkaplan@juniper.net