Network Working Group
Internet-Draft
Intended status: Informational
Expires: 9 November 2026

A. Menon
Maia Edge
P. MeLampy
Retired
M. Baj
Hewlett Packard Enterprise
P. Timmons
Maia Edge
H. Kaplan
Hewlett Packard Enterprise

8 May 2026

Hewlett Packard Enterprise’s Secure Vector Routing (SVR)
draft-menon-svr-08

Abstract

This document describes Hewlett Packard Enterprise’s Secure Vector Routing (SVR), an overlay inter-networking protocol that operates at the session layer. SVR conveys end-to-end network requirements that cannot be expressed in network-layer headers by carrying application-layer cookies in the first packet of a session, eliminating the need for tunnel encapsulation and for non-overlapping underlay address spaces. SVR traverses middleboxes and address translators between private networks, the IPv4 public Internet, and the IPv6 public Internet, and supports SD-WAN, multi-cloud, multicast, and dual-stack use cases while improving security at the routing plane.

Note

This draft provides information for the Internet community. It does not specify an Internet standard that has IETF consensus, nor is this part of a standards track of the IETF. This document is being published for the purposes of interoperability. The authors request suggestions for improvement.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IP routers need a way to communicate end-to-end network requirements and to select paths whose attributes meet those requirements. Applications increasingly need the same capability as workloads move to multiple public clouds. Standard practice today is to overlay a mesh of tunnels on top of the IP underlay; this addresses address-space and policy issues but at the cost of additional encapsulation, larger packets, fragmentation, and reduced bandwidth.

Secure Vector Routing (SVR) is proposed as an alternative to tunnels. SVR retains a single network layer, transports traffic securely with authentication and adaptive encryption, and expresses network requirements abstractly so that policy can be interworked between different networks and address spaces.

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

An SVR router expresses a network requirement semantically and shares it with a peer as SVR Metadata, carried as an application-layer cookie in the first packet of a session. Every participating router holds session state and installs matching forward and reverse flows. Once the session is bidirectionally established, the cookie is omitted from subsequent packets, eliminating per-packet overhead.

Benefits of this approach include:

* *Tunnel compression*: SVR Metadata removes the need for an outer tunnel header on established sessions, yielding 12% to 100% bandwidth savings versus IPsec depending on packet size.

* *No tunnel "elephant flows"*: Tunnels are long-lived aggregates pinned to a single ECMP hash. Each SVR session has a unique 5-tuple and is hashed independently by the underlay.

* *Per-flow QoS*: Because each SVR session has a unique on-the-wire 5-tuple, standard MPLS routing and DSCP-based QoS work without the entry-marking and copy-policy gymnastics required for tunnels.

* *No re-encryption*: SVR's adaptive encryption encrypts only sessions that require it, avoiding the encrypt-already-encrypted penalty common to IPsec tunnels.

* *Inspection-friendly*: Firewalls and security proxies that are passive to SVR Metadata can still intercept TLS for decryption/ inspection. This is not possible with IPsec by design.

* *Encryption scaling*: Per-session keying allows encryption to scale across cores. Tunnel termination is bounded by a single core per tunnel (at this writing, ~1.5 Gb/s).

1.3. Document Organization

Section 3 describes how SVR works at a high level. Section 4 walks through a worked example. Section 4.7 details first-packet processing. Section 6 covers Peer Pathway management with BFD. Section 7 describes additional procedures. Section 8 and Section 9 cover multicast and IPv6. Section 10 defines the on-the-wire SVR Metadata format and TLV catalog.

1.4. Definitions

The following terms are used throughout this document.

Authority: A textual identifier for the owner of an SVR namespace. The Authority allocates Tenant and Service names within its namespace and acts as a logical administrative-domain boundary. Authority namespaces SHOULD be globally unique when interworking is desired. Naming-dispute resolution is out of scope.

Tenant: A textual identifier for a set of network endpoints that share a common access policy. Endpoints MAY be mapped to a Tenant by source IP/mask, VLAN tag, ingress interface, authentication system, or client assertion; the mapping mechanism is out of scope. Tenant names are scoped to an Authority and MAY use hierarchical domain-style syntax (e.g., location.department.example).

Session: The complete bidirectional sequence of packets representing a single TCP or UDP communication. The initiator is the client; the responder is the server.

Service: A textual identifier for the destination(s) reachable via SVR (e.g., "Zoom", "Office365/Outlook"). The mapping from Service name to concrete endpoints (URLs, IP/protocol/port tuples, CIDR blocks, etc.) is out of scope. Quality and Security policy MAY be associated with a Service in data models not described here.

Context: The original 5-tuple of an IP packet (source IP, source port, destination IP, destination port, protocol). Layer-2 information (MAC, VLAN) MAY also be included for certain use cases.

Signature: An HMAC computed by the source router. SVR Metadata packets SHOULD be HMAC-signed, and all packets traversing an SVR Peer Pathway SHOULD carry an HMAC signature so the next-hop router can authenticate the sender and verify integrity. The signed range MUST NOT include the IP header, since the packet may traverse a NAT or an IPv4/IPv6 translator.

Direction: Inferred per session, not carried as a TLV. "Forward" is client-to-server (the side that sent the first packet); "reverse" is server-to-client. Direction is independent of network topology; traffic between two nodes may be "forward" for some sessions and "reverse" for others.

Peer: A client, server, or router that supports SVR. A Peer MAY be directly adjacent or reachable across an IP network. "Peer" in this document always means SVR Peer; it is unrelated to BGP peering (though SVR Peers are commonly co-located with BGP peers at network edges).

Waypoint: A reachable IP address on an SVR router’s interface. A single interface MAY expose multiple Waypoints, and a Waypoint MAY change dynamically when an interface uses DHCP or PPPoE.

Peer Received IP Address: The destination IP address used to reach a Waypoint. Equal to the Waypoint Address unless a NAT lies between the Peers.

SVR Metadata: A block of TLVs (defined in Section 10) that carries SVR attributes between SVR routers.

BFD Metadata: Data appended to BFD messages to support SVR Peer relationships beyond what standard BFD provides. See Section 6.

Peer Pathway: A unique pair of mutually reachable Waypoint addresses (or domain names that resolve to such addresses). Peer Pathways have availability, performance (jitter, latency, loss), and cost attributes. BFD [RFC5880] is the RECOMMENDED method for monitoring Peer Pathway state.

Router Certificate: An X.509 certificate issued from a Certificate

Signing Request that contains the router's UUID, Authority, and public key. Used to authenticate routers on Peer Pathways. The certificate is long-lived and rarely used directly; subsequent keying uses derived keys.

Peer Key: A shared key established between two authenticated Peers, used to encrypt selected BFD fields between them. Rekeyed as often as required by deriving a new shared key. See Section 6.1.8.

SVR Metadata Key: A per-router key used to encrypt and decrypt SVR Metadata. The router distributes its current Metadata Key to all of its Peers in an encrypted BFD Metadata field. See Section 6.1.9.

Payload Key: A per-session key for payload encryption, generated by the originating router and conveyed inside SVR Metadata. Reusing the same key in both directions avoids glare. Rekey is achieved by generating a new key and conveying it in mid-flow SVR Metadata.

Session HMAC Key: A per-session key used for Time-Based HMAC signatures, used to protect SVR pathways against replay. Set to the Peer Key in effect at session creation, retained for the session's lifetime. Time-Based HMAC effectively rotates the signing input every two seconds without changing this key.

2. Configuration and Management

Provisioning and orchestration are out of scope for this document. This section sketches the minimum set of attributes that any orchestration mechanism MUST be able to deliver to a router in order to bring up an SVR peering relationship; it is neither exhaustive nor normative as to data-model organization.

```
            Adjacency[]
              port-range
              peer[]
              rekey-interval

            Tenant[]
              name
              source-ip[]

            SecurityPolicy[]
              name
              HMAC
                mode
                cipher
              payload-encryption-cipher
              adaptive-encryption
              rekey-interval

            Service[]
              name
              domain[]
              ip-address[]
              port[]
              protocol[]
              permit
                Tenant[]
              deny
                Tenant[]
              ServiceLevelAgreement
                max-jitter
                max-loss
                max-latency
              SecurityPolicy
```

Figure 1

3. Theory of Operation

SVR is a session-stateful routing protocol that runs at network edges, typically co-located with stateful NAT and multipath routing functions: branch routers, data-center edges, and public-cloud gateways. SVR maps local network requirements into administratively defined text strings with Authority-wide meaning, and signals them by inserting an SVR Metadata cookie directly into in-flight IP packets.

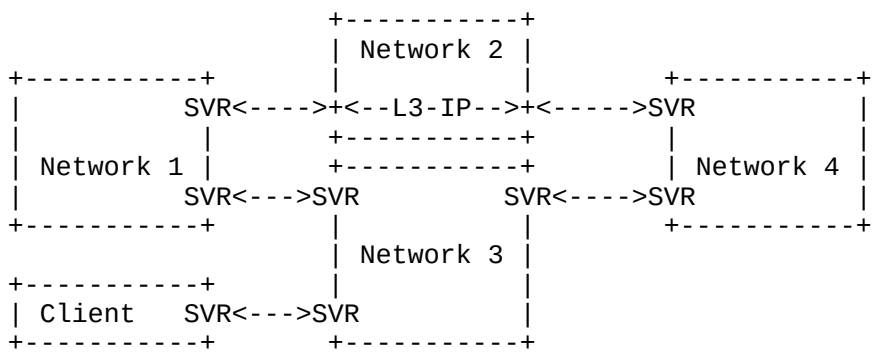


Figure 2

Figure 2 shows typical SVR deployment topologies. SVR may be deployed end-to-end (Network 1 to Network 4 across Network 3, the public Internet), in a chain (Network 1 to Network 3 to Network 4), or directly to SVR-capable clients attached to Network 3.

SVR Metadata is inserted directly after the L4 header (see Section 5.2). Metadata in the first packet of a session is used for path selection and security; subsequent packets MAY carry SVR Metadata to update networking requirements mid-session. Metadata lives in the packet payload to guarantee delivery between SVR routers.

SVR supports TCP, UDP (including UDP unicast), Multicast, MP-TCP, QUIC, point-to-point Ethernet, and ICMP. A session is bracketed by an initial packet with a unique 5-tuple and ends either when the L4 protocol indicates termination (TCP FIN/FIN-ACK or RST) or after a configured inactivity timeout.

3.1. Directionality

Every SVR session has a direction: "forward" is from the sender of the first packet (the client) toward the responder (the server); "reverse" is the opposite. Direction is independent of network topology -- the same Peer Pathway carries forward sessions in both physical directions. Routing policy is expressed as Tenant (source) / Service (destination) pairs that match the forward direction.

SVR Metadata is similarly labeled "forward" (client-to-server) or "reverse" (server-to-client) throughout this document.

3.2. Order of Operations

Processing of a packet at the first (ingress) SVR router proceeds as follows:

1. *Detect new flow*: a packet whose 5-tuple is not in the flow table is treated as the first packet of a new flow.

2. *Parse and route*: parse L2/L3 headers and perform longest-prefix match to determine the next hop.

3. *SVR-capable next hop?*: if the next hop is a known SVR Peer, continue with SVR processing; otherwise forward as a traditional IP router.

4. *Insert SVR Metadata* into the first packet of the new session.

5. *Rewrite transport addresses* to the chosen Peer Pathway's Waypoint pair (analogous to IPv6 Segment Routing); the original addresses are preserved in the Forward Context TLV.

6. *Forward* the SVR-encapsulated packet.

Processing at a subsequent SVR router differs only at the start: a first-packet arrival on a router-interface address from a known SVR Peer is interpreted as carrying SVR Metadata. The router decrypts and authenticates the metadata, and either restores the original addresses (if the next hop is non-SVR) or repeats steps 2-6 above with a new Peer Pathway selection.

3.3. SVR with Other Traffic

SVR coexists with traditional routing and non-SVR traffic. Waypoint addresses MUST remain reachable via traditional networking for every Peer relationship. When the next hop returned by FIB lookup matches a known Peer's Waypoint, the router MAY insert SVR Metadata and apply session-stateful SVR; otherwise it forwards as a traditional IP router.

3.4. SVR Metadata Handshake

For each session, peers perform a metadata handshake to confirm that SVR Metadata has been received and understood. The initiating router inserts forward SVR Metadata into every packet it sends to the recipient until it receives a reverse packet that itself carries SVR Metadata. The recipient inserts reverse SVR Metadata into every packet until it receives a packet from the initiator that no longer carries SVR Metadata. Once both sides have observed metadata

disappear from their counterpart's stream, the handshake is complete and subsequent packets travel without metadata overhead.

3.5. Pathway Obstructions and Changes

Middleboxes (firewalls, NATs, traffic-inspection devices) on a Peer Pathway may drop TCP SYNs that carry payload data, or invalidate TCP streams whose sequence numbers appear inconsistent due to inserted metadata (even though SVR does not change sequence numbers). Peers MAY probe their pathways to detect these conditions; BFD with BFD Metadata is used to detect NATs (see Section 6.1.2). Well-known NAT-traversal protocols -- STUN [RFC8489], TURN [RFC6062], ICE [RFC8445] -- are out of scope for this document.

When a NAT is detected, the affected SVR router records the pathway's externally observed Waypoint address as a pathway attribute and uses NAT keep-alives (see Section 7.3) to preserve the binding.

When a non-NAT middlebox is detected, the transmitting router MAY perform TCP-to-UDP transformation (changing the protocol octet from 0x06 to 0x11), with the receiving router restoring the original protocol. See Section 5.2.7 and Section 5.6.3.

Dynamic Waypoint addresses (DHCP, PPPoE) MAY change mid-session. For active sessions, the new address is detected by inspecting the source address on incoming packets and updating internal references. For idle pathways, BFD with SVR Metadata detects the change; see Section 6.1.3.

3.6. SVR Metadata Removal

SVR Metadata MUST be removed before traffic is delivered to a non-SVR endpoint. A router can be certain that a downstream SVR peer will perform that removal only when the FIB next-hop address exactly matches a known Peer Waypoint address; reachability of the Peer SHOULD be confirmed via BFD (Section 6) before metadata insertion (see also Section 4). If the next hop is not a known reachable Peer, SVR Metadata insertion MUST NOT be performed. SVR Metadata MAY be sent end-to-end when both client and server support SVR.

3.7. Transport Address Modification

To steer a packet to a specific peer, the destination address is rewritten to that Peer's chosen Waypoint and the source address is rewritten to the local egress Waypoint, ensuring symmetric return. The original 5-tuple is preserved in the Forward Context TLV (Section 10.4.1) and restored at egress. This is conceptually similar to IPv6 Segment Routing ([RFC8986]) or to LISP RLOCs

([RFC9300]), except that the original addresses live in SVR Metadata in the packet payload, not in the IP header.

Waypoint selection is implementation-specific. A FIB lookup typically returns one or more Waypoints; when multiple are returned, the router MAY apply pathway quality, session-layer load balancing, or other local logic to choose. See Section 4.

Session state, including translation rules for all five tuple fields in both directions, MUST be held on both the source and destination SVR routers. Once installed, this state replaces tunnel encapsulation/decapsulation -- a substantially cheaper operation than tunneling.

3.8. Optional Tenant- and Service-Name Routing

SVR Metadata carries textual Tenant and Service names alongside IP-level context. A FIB extended with Service-name entries permits policy and route selection based on those names; the egress SVR router typically applies destination NAT to a concrete instance. This is particularly useful for inter-cloud connectivity (e.g., AWS <-> Azure).

When clients support SVR, semantic routing can replace dynamic DNS for service location: the client requests the Service by name and the SVR network resolves and steers the session to the best instance. A local DNS server resolving Service names to a nearby SVR router achieves the same effect for legacy clients.

3.9. Unique 5-Tuples per Session

The ingress SVR router (client side) allocates both source and destination ports for the on-wire 5-tuple. Source ports MUST be even; destination ports MUST be odd. This convention guarantees uniqueness between any two peers without negotiation, and even when a reverse-direction session is allocated the same port pair, the swapped IP addresses keep the on-wire 5-tuple unique. Allocatable port ranges are provisioned per Peer Pathway; a port MUST be free before being reused.

With no NAT between Peers, this scheme provides 2^32 unique sessions per pathway (2^30 per direction); a source NAT reduces this to roughly 63,000 (2^16 minus 1024 reserved). Middleboxes may further restrict the usable range.

Because the original packet’s DSCP and other QoS-related fields are preserved on the wire, underlay QoS continues to operate on a per-session basis.

3.10. Post-Handshake Packet Handling

Once the handshake is complete, every subsequent packet has all five tuple fields rewritten in both directions. Compared to IPsec, this transformation is significantly cheaper: no memory copies, no new header construction, no new checksums, and no mandatory encryption.

3.11. Session State Requirements

Every SVR router MUST maintain per-session state, including the original addresses and translation rules. This state allows peers to stop sending SVR Metadata after the handshake and to forward traffic akin to segment routing. State can be lost in three ways:

* *Ingress router state loss* -- e.g., during failover or power loss at the originating SVR router.

* *Intermediate router state loss* -- at the 2nd-to-Nth SVR router on the path.

* *Middlebox state loss* -- a NAT or firewall between two SVR routers loses or modifies session state.

When state is lost, the affected peer MUST securely reacquire it. If neither peer holds state, the session is processed as a new first packet (see Section 4.7.1).

State-loss detection: Before transmitting each packet, an SVR router compares the elapsed time since the last received packet from the peer to a configurable inactivity timeout (RECOMMENDED 5 seconds). Exceeding this threshold MAY indicate state loss. This logic is independent of session-idle timers used for state expiry.

Unicast / asymmetric flows: For unidirectional or highly asymmetric flows, this timeout will trigger routinely; this is expected.

Health check: On timeout, a Session Health Check SVR Metadata TLV is inserted in the outgoing packet. A peer that holds valid session state responds with a generated packet carrying Forced Drop SVR Metadata, reason 8 ("health check OK"). This bidirectional exchange also confirms middlebox state. It is incidental to normal traffic and does not replace session keep-alive (see Section 3.12).

State not present: A peer that receives a Session Health Check but holds no state for the session responds with a generated packet carrying Forced Drop SVR Metadata, reason 2 ("send full session metadata"). See Section 7.8. If a middlebox has lost state or

mangled the session (e.g., CGNAT) such that no Health Check response arrives, the next forward packet is treated as a new session (Section 4.7.1).

3.12. NATs and Session Keep-Alive

When NAT or a stateful firewall lies between Peers, the source address observed at the receiving SVR router may differ from the sender's Waypoint. Routers MUST store the actual on-wire Waypoint addresses received from each Peer (in addition to provisioned values).

For sessions traversing such middleboxes, the SVR router behind the middlebox SHOULD send periodic SVR Control Messages on idle sessions to keep pinholes open. The RECOMMENDED idle interval is 20 seconds. The Control Message uses the saved source/destination ports of the idle session; see Section 10.3.6 for the message format and Section 7.3 for an example.

3.13. Use of BFD on Peer Pathways

BFD [RFC5880] is the RECOMMENDED mechanism for Peer Pathway management. It is used for reachability, NAT detection, Waypoint-address-change detection, MTU discovery, idle-pathway quality measurement, peer authentication, and key maintenance. Alternative mechanisms MAY be used by mutual agreement of the Peers.

Because standard BFD authentication fields are insufficient for SVR's needs, this document defines BFD Metadata: an extension appended to the end of a BFD control packet (in contrast to SVR Metadata, which follows the L4 header). Some BFD Metadata fields are encrypted. Protocol Buffers, rather than TLVs, are used to encode BFD Metadata for ease of processing. The full BFD Metadata definition is in Section 6.

4. SVR Multi-path Routing Example

The example below shows two SVR capable routers, peering with each other over multiple pathways.

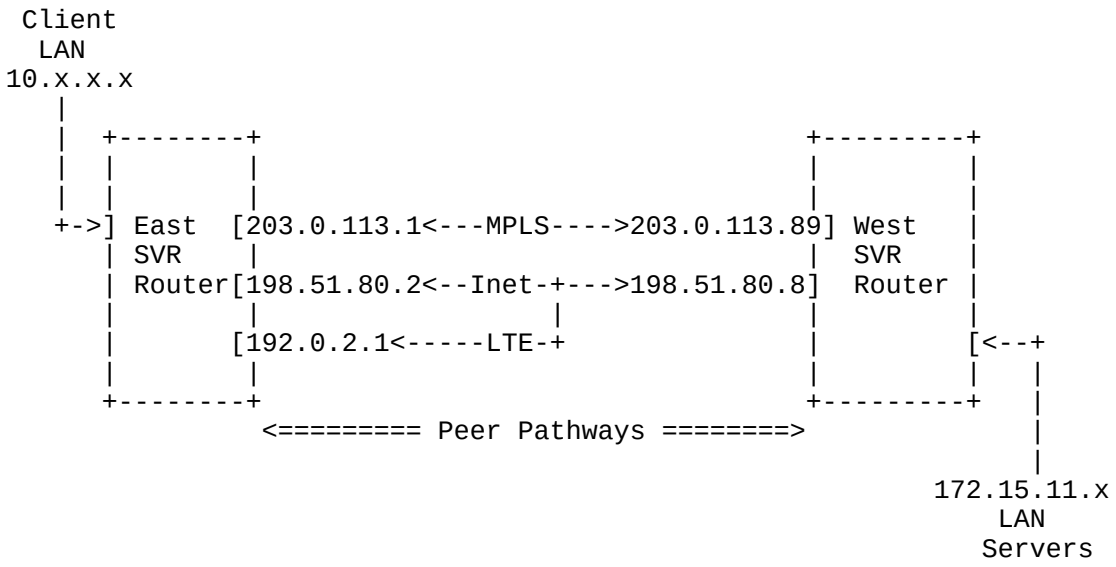


Figure 3

Note: The client, server, and MPLS network can support the private routes in 10.x.x.x and 172.15.11.x address spaces natively, but the internet and LTE networks do not. This is an example of using secure vectors to join networks together.

4.1. Establishing SVR Peering

Peering proceeds in two steps. First, each router applies any local static L3 routes and exchanges routes via standard L3 protocols (BGP, OSPF, etc.) to build a FIB and confirm bidirectional Waypoint reachability. Second, each Peer Pathway between the two routers is authenticated; pathways SHOULD be authenticated bidirectionally before being used for SVR traffic.

4.1.1. Reachability and Peer Authentication

Peer authentication is RECOMMENDED. Sending SVR Metadata without authentication is technically possible but discouraged; either Peer MAY require authentication and reject the relationship if it fails.

Authentication uses a Router Certificate: an X.509 certificate issued from a CSR containing the router's UUID and Authority, signed by a trusted CA. The UUID is administrator-assigned and persists across hostname changes, and avoids exposing router names (typically the hostname) in packet traces. Device registration, certificate signing, and secure installation are out of scope; see [RFC4210].

Elliptic Curve cryptography ([RFC8422]) is RECOMMENDED over RSA. The curve is administratively chosen and MUST be the same across all routers in an SVR network; NIST P-256 is RECOMMENDED.

Each Peer transmits a BFD packet with cleartext BFD Metadata carrying its X.509 Router Certificate in PEM format ([RFC5758]); see Section 6.1.7. The receiver verifies the certificate (chain to a trusted CA), confirms the certificate's common-name UUID matches a configured Peer, and stores the public key for use in subsequent keying (Section 4.1.2). To acknowledge receipt and stop further certificate transmission, the Peer sends a BFD packet without a certificate. A Peer MAY request retransmission (e.g., after reboot) by sending its own certificate again. Certificate updates trigger a repeat of this exchange; existing valid keys remain operational to avoid recertification outages.

4.1.2. Peer Cryptographic Key and Re-keying

In the example (Figure 3), the East-West Peer relationship has three authenticated Peer Pathways. Securely exchanging BFD between East and West requires a Peer Key. Because the Router Certificate's keypair is long-lived, the Peer Key is derived using ECDH-ES with the Concat KDF ([NIST_SP_800-56A]).

Concat KDF inputs are: a Salt from each Peer, the local private key, and both public keys. The resulting key is implicitly authenticated by the certificates, preventing man-in-the-middle attacks. Encrypted BFD Metadata fields then distribute SVR Metadata Keys; see Section 6.1.9.

A nonce in BFD Metadata, tracked at the receiver, prevents replay of old BFD packets. To bound nonce-table size, tracking SHOULD be rate-limited (e.g., one per minute -- ~1440 entries per Peer per day).

Either Peer MAY rekey at any time by re-running the same exchange with new Salt values. A single Peer Key is shared across all pathways between two Peers, which is efficient when many parallel pathways exist.

4.1.3. Metadata Cryptographic Key and Re-keying

SVR routers may receive metadata-bearing packets on any interface, from any Peer, and source addresses can change due to NAT or rerouting. Each router therefore needs to know the cryptographic key for each Peer in advance.

Each SVR router generates (or obtains via a quantum-safe mechanism) a single SVR Metadata Key and distributes it to all Peers via encrypted BFD Metadata, along with a version identifier. All SVR Metadata sent from this router uses this key until rotation. See Section 6.1.9.

4.1.4. Bringing a Peer Into Service

A Peer is declared operational and in service when at least one authenticated pathway exists and an Elliptic Curve Peer Key (ECPK) has been established. It is then ready to carry bidirectional traffic.

4.1.5. Resulting Peer Relationship

In service, East and West run BFD on each pathway to monitor operational status and measure jitter, latency, and packet loss. For the running example, assuming all pathways are healthy:

```
PEER: East -> West Authenticated/In Service
  Name      Description                    Characteristics
  MPLS      203.0.113.1->203.0.113.89      20ms Lat, 0 Loss,  2 Jit
  Internet  198.51.80.2->198.51.80.8       30ms Lat, 0 Loss,  3 Jit
  LTE       192.0.2.1->198.51.80.8         50ms Lat, 0 Loss, 15 Jit

PEER: West -> East Authenticated/In Service
  Name      Description                    Characteristics
  MPLS      203.0.113.89->203.0.113.1      20ms Lat, 0 Loss,  2 Jit
  Internet  198.51.80.8->198.51.80.2       30ms Lat, 0 Loss,  3 Jit
  LTE       198.51.80.8->192.0.2.1         50ms Lat, 0 Loss, 15 Jit
```

Figure 4

BFD also runs on in-service Peer Pathways for MTU discovery, address-change detection, and idle-pathway quality monitoring.

4.2. CIDR-based SVR Peer FIB Entries

Forwarding sessions onto an SVR Peer Pathway requires a route lookup that resolves to an SVR Peer (Waypoint) next-hop. Continuing the running example, assume servers in 172.15.11.0/24 sit behind West and West advertises the prefix to East over each pathway. East's FIB then contains:

```
East's Forward Information Base (FIB)
   Route             Next-Hop IP Addr
   ----------------  -----------------
   172.15.11.0/24    203.0.113.89
   172.15.11.0/24    198.51.80.8
   ....
   [FIB Entries to reach Waypoints omitted]
```

Figure 5

Assume an Authority "example" defines Tenant "engineering" as 10.0.0.0/25 on VLAN 2 and Service "github.example" as 172.15.11.23 TCP/22. (Policy provisioning is out of scope.) When an engineering client initiates a session toward github at the East router, FIB lookup yields two next-hop Waypoints, which East cross-references against its Peer Pathway list to identify three usable pathways:

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

Figure 6

East selects a pathway based on quality SLAs and/or load-balancing policy, then constructs SVR Metadata, inserts it into the first packet, and forwards it down the chosen pathway to West. The running example uses a private LAN behind East with overlapping addresses (typical of branch deployments), so East applies source NAT.

Assuming MPLS is chosen, East performs first-packet processing (Section 4.7.1) and emits the packet on its MPLS interface with source 203.0.113.1 and destination 203.0.113.89 -- the MPLS pathway's Waypoint pair. The bidirectional session uses the same address pair (reversed for return); per-session uniqueness on the wire is provided entirely by the SVR-allocated source/dest ports.

4.3. Optional Service-Name FIB

Because SVR Metadata carries Service names as text, an SVR FIB MAY be extended with name-based entries to enable routing decisions on Service name rather than (or in addition to) CIDR. Two use cases motivate this:

Avoiding Dynamic DNS: Dynamic DNS is often used to answer "which IP is the best instance right now?". It can be slow to update, costly, and oblivious to private-network path state. SVR can route by Service name directly and resolve the destination at egress.

Multi-cloud networking: Public clouds publish accurate per-instance DNS, but only inside the cloud. SVR routers can resolve Service names at egress to bridge clouds without exposing those DNS responders externally.

An example FIB combining named and CIDR entries:

East's Extended SVR Forward Information Base (OPTIONAL)

Service Name	Route	Waypoint	Egress Action
github.example	172.15.11.23:TCP:22	203.0.113.89	FWD
github.example	172.15.11.23:TCP:22	198.51.80.8	FWD
logsvc.example	172.15.11.20:UDP:514	203.0.113.89	DNS
logsvc.example	172.15.11.20:UDP:514	198.51.80.8	DNS
https.example	172.15.11.24:TCP:443	203.0.113.89	DEST NAT -196.168.1.1 -196.168.1.2 -196.168.1.3

[FIB Entries to reach Waypoints omitted]

Figure 7

Longest-prefix matching on destination address combined with exact protocol/port match is used at ingress to select the route. The Service name string (e.g., "github.example") then travels in SVR Metadata across the SVR network until egress. Implicit default-deny: in this example only SSH, Syslog, and HTTPS are permitted; everything else is dropped.

The egress action MAY be one of:

Forward (default): Restore the original IP addresses and forward; apply source NAT if a Source NAT TLV is present.

DNS: Resolve the Service name locally via DNS at the egress router.

Destination NAT: Rewrite the destination to a chosen address (or load-balance across a pool); equivalent to a load balancer.

Named routes coexist with CIDR FIB entries; named routes are matched first, with CIDR as fallback.

4.4. SVR Security Definitions

An Authority MUST provision a common set of security parameters across its peers:

HMAC algorithm: SHA1, SHA256, or SHA256-128.

Time-Based HMAC: YES or NO.

HMAC scope: NONE, SVR Metadata only, or ALL packets.

SVR Metadata block cipher: NONE, AES128, or AES256.

Other ciphers MAY be used, provided they produce fixed, well-known block sizes for both signing and encryption.

Per-session payload-encryption Security Policies are negotiated in the first-packet SVR Metadata. The example uses:

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

Figure 8

4.5. Time-Based HMAC Details

SVR uses Time-Based HMAC (per [RFC2104]) for authentication and integrity; see Section 5.5.1. The running example uses SHA256-128 (16-octet signature).

4.6. Payload Encryption

Every SVR Metadata transaction includes a Security ID header TLV (Section 10.3.2).

A unique Payload Key is used per session, for each direction. Keys are generated by the originating router and conveyed in the encrypted portion of metadata. Per-flow keying eliminates glare and simplifies key exchange. Payload Keys and IVs SHOULD be generated using a FIPS 140-3-approved DRBG.

Long-lived sessions MAY require rekey if their duration exceeds the configurable rekey interval. Rekey is performed by generating a new key, conveying it in mid-flow SVR Metadata, and applying it to all subsequent packets. Downstream SVR routers MUST monitor for mid-flow metadata and update their keying state accordingly.

4.7. New Session Initiation in Detail

The ladder diagram below shows the example github SSH session traversing the East and West routers.

Ladder Diagram for SSH Example:

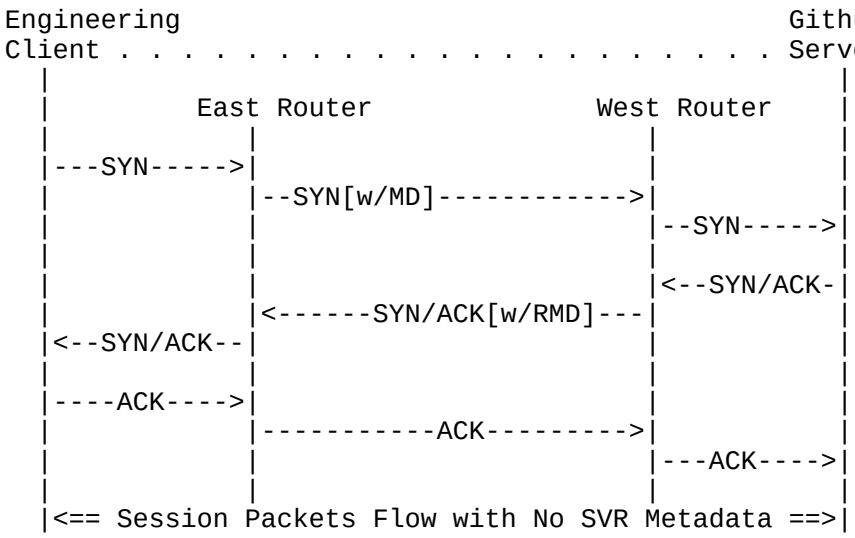


Figure 9

East MUST construct and insert SVR Metadata into the first packet of the SSH session (the TCP SYN). West MUST remove the metadata, forward the SYN, and on receipt of the SYN/ACK insert reverse SVR Metadata into it. Once both directions have seen metadata from their counterpart, the handshake is complete; metadata is carried in existing packets when possible.

Detecting a first packet is protocol-specific: for TCP, a new 5-tuple with only the SYN flag set; for UDP, a new 5-tuple not currently active.

4.7.1. East: First-Packet Processing

Assume the SSH packet arrives at East from the client LAN as shown:

Arriving Packet at East Router

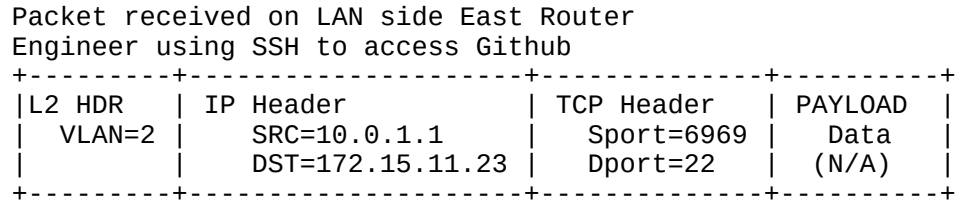


Figure 10

4.7.1.1. Determine Tenant

A Tenant is a textual identifier for a group of source endpoints with common access policy (akin to a security zone). In the example, the "engineer" Tenant is mapped from VLAN 2 in the Authority "example". The attribute-to-Tenant mapping is implementation-specific; routers SHOULD associate a default Tenant with each logical interface.

4.7.1.2. Determine Service

Service identification is out of scope for this document but is the mechanism by which a session is bound to policy. Common techniques include IP/port ranges, TLS SNI, certificate Common Name, and HTTP URL extraction; SaaS vendors typically publish their CIDRs and ports. For the example, 172.15.11.23 TCP/22 is taken to be the "github" Service in Authority "example".

4.7.1.3. Determine Network Requirements

With Tenant and Service known, network requirements are looked up:

Example Network Requirements

```
SERVICE: github
  Access Policies:
    Tenants Allowed: engineering
    Tenants Denied: release.engineering
  Quality Policy: latency < 40ms
  Security Policy: None
```

Figure 11

Access policies determine which tenants are allowed, and if any are specifically denied. The Quality policy defines the service level experience requirements. Secure Vector Routing exchanges tenants, services, and security policies using character strings in SVR Metadata. Access and quality policies are defined and used locally within a router and logically associated with 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.

4.7.1.4. Picking a Peer Pathway

Of East’s three reachable Peer Pathways, the 40 ms latency budget eliminates LTE. MPLS and Internet are both within SLA; MPLS is chosen for its lower latency.

Pathway selection criteria are implementation-specific and may include pathway utilization and capacity, cost (e.g., reserving LTE/5G for backup), and operator-defined load-balancing policies. The selection algorithm is out of scope for this document.

4.7.1.5. Allocate Source NAT (if Required)

The example uses source NAT at the East router’s MPLS interface so that overlapping branch address spaces can coexist. A NATing router may reuse the interface address with a new source port, or allocate from an IP pool. Either way, the chosen address is placed in SVR Metadata; the actual translation is applied at the egress (West) router.

4.7.1.6. Allocate SVR Ports

The router allocates new source and destination ports for the session. Ports MUST NOT be currently in use and SHOULD NOT have been recently freed (to avoid middlebox state confusion); see Section 5.2. The allocatable range is provisioned per site to accommodate upstream firewall and middlebox restrictions. East has 8000-24000 available; the example uses source 8000 (even) and destination 8001 (odd). Both ports are allocated by the router that initiates SVR Metadata.

4.7.1.7. Build Session State and SVR Metadata

With requirements, pathway, and ports in hand, East creates session state (including a Session UUID for end-to-end tracking) and builds the SVR Metadata block. The table below maps state to SVR Metadata TLVs (full TLV definitions in Section 10):

Session State Table Entry

State Information & Mappings to SVR Metadata Fields

Category	SVR Metadata TLV -Field	VALUE	TLV Type	TLV Len	TLV 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	e9b083df-d922.....	6	16	4
	Source Router Name	East Router	14	11	4
	Source NAT Address	203.0.113.1	25	4	4
	Security Policy	NONE	15	4	4
	Peer Path		19	22	4
	- Source Addr	203.0.113.1			
	- Dest Addr	203.0.113.89			
				---	---
	Total Payload Length = 119 (87+32)			87	32

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

Session HMAC Key [Peer Key at session start]

Figure 12

The required first-packet TLVs are defined in Section 5.3.1: Security ID, Forward Context, Tenant Name, Service Name, Session UUID, Source Router Name, Security Policy, and Peer Pathway ID. The example also includes two optional TLVs: Path Metrics (Section 10.3.7) and IPv4 Source NAT Address (Section 10.4.13).

TLV order is arbitrary, but Header TLVs MUST precede Payload TLVs. A peer that receives an unknown TLV MUST ignore it. In this example the header (with two Header TLVs) is 34 octets and the eight Payload TLVs total 119 octets.

The Session HMAC Key is router-local state, conveyed between Peers in BFD Metadata, and used for the life of the session.

4.7.1.8. Encrypt SVR Metadata

The Payload TLVs are encrypted per Section 5.4. The example provisioning uses AES-256, which has a 128-bit (16-octet) block size. With 119 octets of payload, 9 octets of padding bring the encrypted region to 128 octets; a 16-octet IV is appended. The full SVR Metadata block is 34 (header) + 128 (encrypted) + 16 (IV) = 178 octets:

SVR Metadata Block

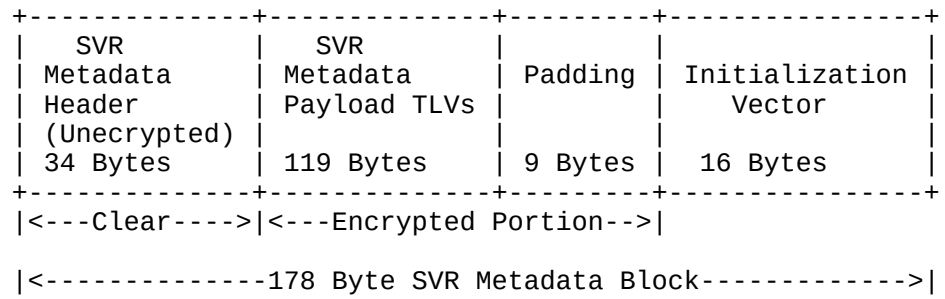


Figure 13

4.7.1.9. Insert SVR Metadata

The 178-octet block is inserted directly after the L4 header. Any existing payload data is shifted to make room:

SVR Metadata Added

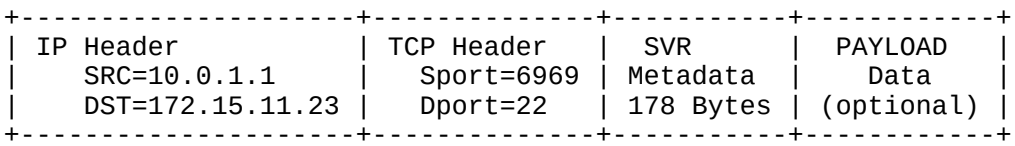


Figure 14

The transport addresses are then rewritten to the chosen Peer Pathway's Waypoints, and the SVR-allocated ports are installed:

Transport Addresses Updated

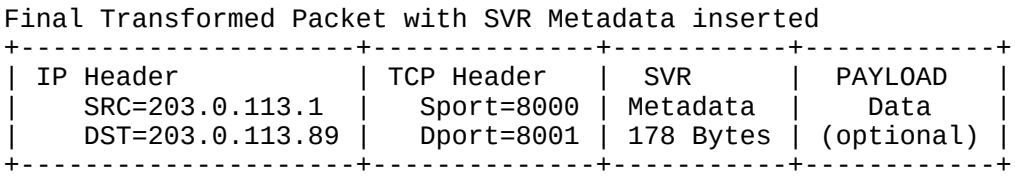


Figure 15

4.7.1.10. Sign SVR Packet

The packet is then HMAC-signed (Section 4.5) using the current SVR Metadata Key (Section 6.1.9). The signature is appended to the end of the packet, extending its length by the signature size (16 octets in this example). The IP header is excluded from the signed range.

HMAC Signature Added

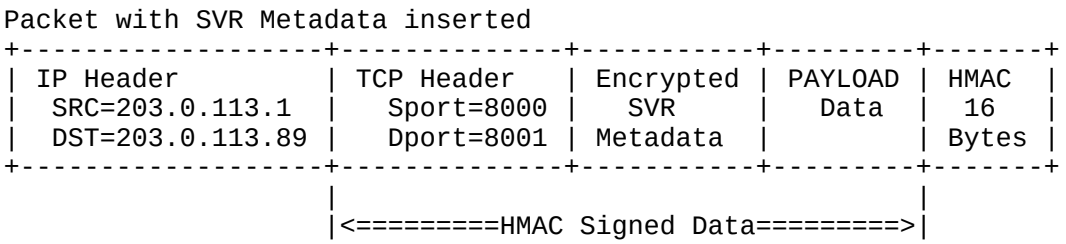


Figure 16

4.7.1.11. Send the First Packet

The IP length and checksum are recomputed and the packet is transmitted. East continues to include the same SVR Metadata in every outbound packet of the session until it sees a reverse packet carrying SVR Metadata (handshake complete). For TCP this is typically the SYN/ACK; thereafter no further metadata is inserted for the lifetime of the session unless a path-migration event occurs.

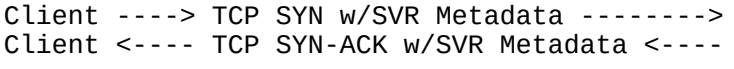


Figure 17

For UDP, metadata is inserted on every packet until a reverse packet carrying metadata is observed, except for unidirectional flows; see Section 4.7.7.

4.7.2. West: First-Packet Processing

A packet arriving at West with West’s own Waypoint as the IP destination (i.e., addressed to the router rather than transiting it) likely carries SVR Metadata and is processed as follows.

4.7.2.1. Verify Source Address is a Waypoint

The packet MUST be validated before processing (Section 5.6.2). On authentication or validation failure, the packet MAY be dropped or returned with an ICMP Destination Unreachable. In the example, only three source addresses are valid:

Possible Source Addresses

203.0.113.1	MPLS Peer Pathway
198.51.80.2	Internet Peer Pathway
169.254.231.106	LTE Peer Pathway

Figure 18

4.7.2.2. Verify SVR Metadata Block

The most efficient first test is checking for the SVR header magic number (Section 5.6.1). The HMAC signature is then verified (Section 5.5.1); on failure the packet is dropped and a security event recorded. The unencrypted portions of the SVR Metadata header MUST be sanity-checked (header and payload lengths must each be less than the total block size).

4.7.2.3. Decrypt, Parse, and Save State

The metadata block is decrypted (Section 5.6.2.2); any decryption failure causes the packet to be dropped. The Payload TLVs are parsed and the resulting state and translation rules are installed; any parse failure causes the packet to be dropped. The SVR Metadata block and the HMAC signature are then removed from the packet.

4.7.2.4. Restore Addresses and Route

The original 5-tuple is restored from the Forward Context TLV and the packet is processed recursively as if it were a new first packet (Section 4.7.1), with one difference: the Tenant, Service, Security Policy, and Session UUID are taken from the received metadata rather than locally derived. The packet may then traverse another Peer Pathway or be delivered via standard forwarding -- in the example, West delivers the packet to the server LAN.

When forwarding to another SVR Peer, Tenant, Service, Session UUID, Security Policy, and the original 5-tuple are cloned into the new metadata so that semantics remain consistent across multi-hop SVR paths. SVR Metadata MUST be decrypted and re-encrypted at every hop (because each hop uses different Waypoint addresses), but payload encryption is end-to-end -- applied at the first SVR router and decrypted at the last.

4.7.2.5. Loop Detection

Because every SVR hop preserves the Session UUID, a looping first packet is trivial to detect: there MUST never be two sessions with the same UUID, and any duplicate MUST be dropped. Detecting the loop on the first packet allows subsequent packets to be dropped at ingress. SVR routers MUST also decrement TTL/Hop Limit so that loops not caught by SVR are still bounded by traditional IP forwarding rules.

A packet bearing SVR Metadata that arrives after the handshake is treated as an in-session update, not a loop. Updates may modify any attribute; most commonly they change the Peer Pathway for the session. See Section 7.1.

4.7.3. Pre-established Return-Path State

After East and West each process the first forward packet, both directions of forwarding state and any required translations are installed. Subsequent packets in either direction are matched on 5-tuple and forwarded without further routing-table consultation ("fast path").

4.7.4. Sending Reverse SVR Metadata

Each SVR Router tracks the metadata-handshake status per session. If a forward packet arrives for a session whose handshake is incomplete, the router MUST insert SVR Metadata; it continues until it sees evidence (a reverse packet bearing metadata) that the peer received it. For TCP, the first reverse packet is normally the SYN/ACK. Reverse metadata serves to:

* signal the sender to stop inserting metadata (handshake complete); and

* convey return-direction Service information for future routing decisions.

The reverse SVR Metadata for the example contains:

Reverse SVR Metadata Response

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

Category	SVR Metadata TLV -Field	VALUE	TLV Type	TLV Len	TLV 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	203.0.113.1			
	- Dest IP Addr	172.15.11.23			
	- Protocol	TCP			
	- Source Port	7891			
	- Dest Port	6969			
	Peer Path		19	22	4
	- Source Addr	203.0.113.89			
	- Dest Addr	203.0.113.1			
				---	---
	Total Payload Length = 43 (35+8)			35	8

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

Session HMAC Key [Peer key used by remote peer]

Figure 19

See Section 5.3 for required and optional reverse metadata TLVs. The example also includes the optional Path Metrics TLV (Section 10.3.7).

The Session HMAC Key is router-local state communicated to peers in BFD Metadata; its lifetime tracks the Peer Metadata key. The key is used for the life of the session.

The Forward and Reverse Contexts together give SVR end-to-end visibility of every session: pre-NAT 5-tuple at ingress, post-NAT 5-tuple at egress, and the Waypoint addresses/ports used on each pathway.

This SVR 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 octets of header data, and 43 octets of payload data, 5 octets of padding, and a 16 octets initialization vector resulting in an SVR Metadata block that is 98 octets long.

4.7.5. Subsequent Packet Processing

A peer that receives a session packet without SVR Metadata treats the handshake as complete and stops inserting metadata in its own outbound direction. This applies symmetrically to East and West.

4.7.6. Session Termination

No SVR Metadata is exchanged on normal termination. For TCP, the router monitors FIN/ACK or RST and removes session state after a guard timer; a new SYN matching the same 5-tuple during that window immediately tears down the prior session. All protocols additionally have an inactivity timeout; a packet arriving after the timeout is treated as a new session.

4.7.7. Unidirectional and Asymmetric Flows

When a flow is unidirectional or asymmetric (e.g., TCP sequence numbers advance with no observed reverse packets but end-to-end delivery is confirmed), the sender stops inserting SVR Metadata. For UDP asymmetry, the sender inserts metadata in at most ~20 packets; if no reverse packet appears in that window, the receiving peer generates a "disable metadata" packet to complete the handshake. The exact packet count is implementation-specific.

4.7.8. Multi-Hop Session Ladder Diagram

The diagram below illustrates a TCP session traversing three SVR routers:

Ladder Diagram for Session Initiation with SVR Metadata:

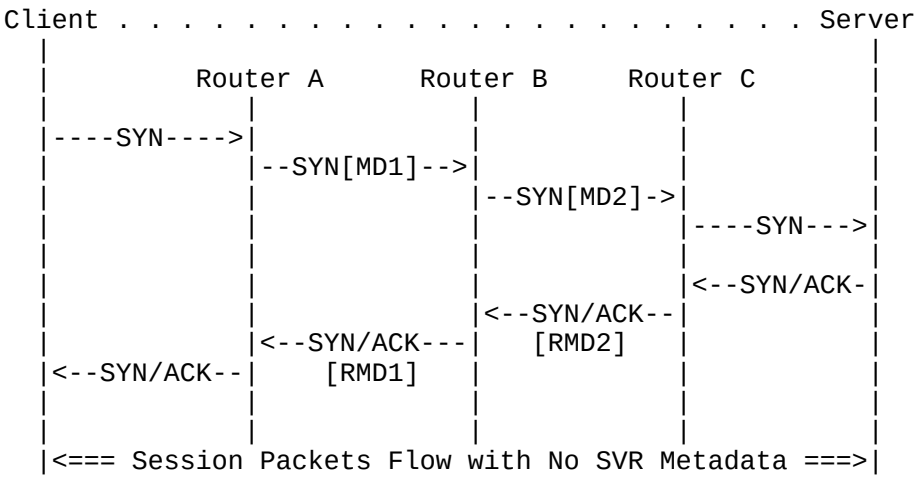


Figure 20

Each router builds metadata (MD1, MD2) for the next chosen Peer; on the first reverse packet each router inserts reverse metadata (RMD1, RMD2). Each router allocates its own Waypoints and ports. The Forward Context, Tenant, Service, and Session UUID are preserved across all hops, allowing consistent policy application end-to-end. The Session UUID is identical in MD1, MD2, RMD1, and RMD2.

A TCP teardown is similarly direct -- no SVR Metadata is required:

Ladder Diagram for Session Teardown SVR Metadata:

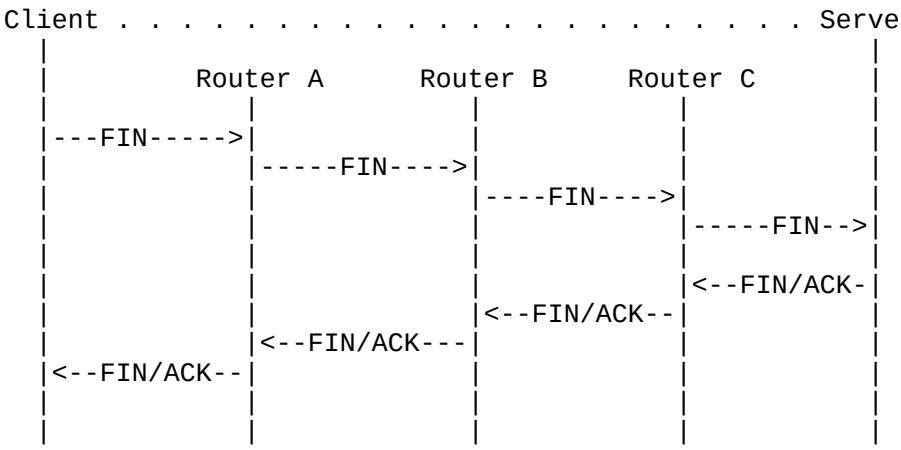


Figure 21

No SVR Metadata is sent on session termination. Each router retains state for a configurable interval (covering FIN/ACK loss) before removing it.

5. SVR Protocol Definition

This section provides the normative requirements for SVR Metadata.

5.1. Sessions and Types

SVR implementations MUST support TCP, UDP, and ICMP. SVR implementations SHOULD support UDP Unicast. A session is identified by a 5-tuple unique to the SVR router; it begins on the first packet and ends when either the L4 protocol signals completion (TCP FIN/FIN-ACK or RST) or after a protocol-specific inactivity timeout (UDP, ICMP, UDP Unicast, point-to-point Ethernet).

SVR is always OPTIONAL: implementations MAY choose per session whether to apply SVR, and SVR implementations MUST also support non-SVR traffic.

5.2. SVR Metadata Insertion

5.2.1. Metadata Location in the Packet

SVR implementations MUST insert SVR Metadata directly after the L4 header, even if doing so causes IP fragmentation. For Ethernet point-to-point and ICMP error messages, IP and L4 headers MUST be created; if associated with an existing session, the created packet MUST share the exact 5-tuple (Waypoints and Ports) of that session. SVR Metadata MUST appear in the very first packet of a new session (TCP or UDP bidirectional flow) to influence path selection or security. SVR Metadata SHALL be inserted in any subsequent packet in either direction to update networking requirements. Metadata is carried in the L4 payload to ensure end-to-end transparency. Packet lengths and L3/L4 checksums MUST be adjusted; TCP sequence numbers MUST NOT be adjusted.

5.2.2. Prerequisites for Insertion

A Peer Pathway MUST be selected before SVR Metadata is inserted. The pathway’s Waypoint addresses serve as the L3 source/destination for every packet bearing SVR Metadata.

5.2.3. Session Port Allocation

The originating SVR peer (client side) MUST allocate both source and destination ports. The ingress side MUST use even ports for the local (source) port and odd ports for the remote (destination) port, guaranteeing uniqueness between any peer pair without negotiation. The allocatable range is provisioned. Ports in use MUST be excluded from allocation; ports MUST be released when the session is removed; and a freed port MUST observe a 60-second guard time before reallocation.

5.2.4. Metadata on Idle Sessions

An SVR implementation MAY need to send metadata to a peer when no data packets are flowing (see Section 7.3). In that case it MUST synthesize an IP packet matching the session’s 5-tuple, marked to be dropped after processing; the directly adjacent peer MUST process and discard it without forwarding to any other SVR peer.

5.2.5. Packet Structure

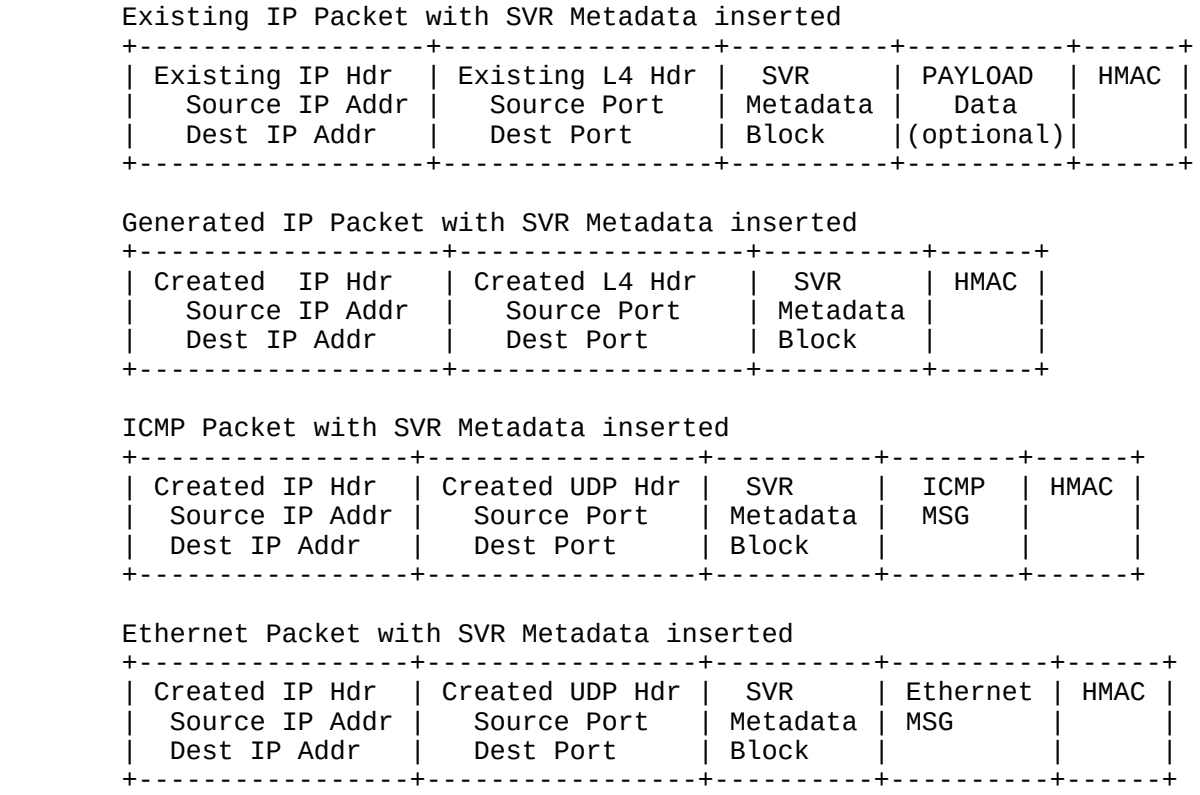


Figure 22

For UDP, the UDP header length field MUST be updated. The L4 checksum (TCP or UDP) MUST be recalculated. The IP total-length field MUST be updated to account for the inserted metadata block and the appended HMAC, and the IP header checksum MUST then be recomputed. For TCP, sequence numbers MUST NOT be modified.

5.2.6. Prevention of False Positives

Because SVR Metadata is carried in the L4 payload, an existing payload could coincidentally begin with the same 8-octet SVR magic number. False-positive logic prevents downstream routers from misinterpreting such payloads.

False positives SHALL NOT occur on first packets, since valid metadata is unconditionally inserted there. They can only arise on mid-session packets of an established SVR session.

When a mid-session packet's payload begins with the SVR magic number, the implementation MUST insert an empty SVR Metadata header (12 octets, zero TLVs). This establishes the contract that any appearance of the magic number on the wire indicates valid metadata that downstream routers MUST process and remove. The inserted header carries no TLVs and is not encrypted.

SVR Metadata Location

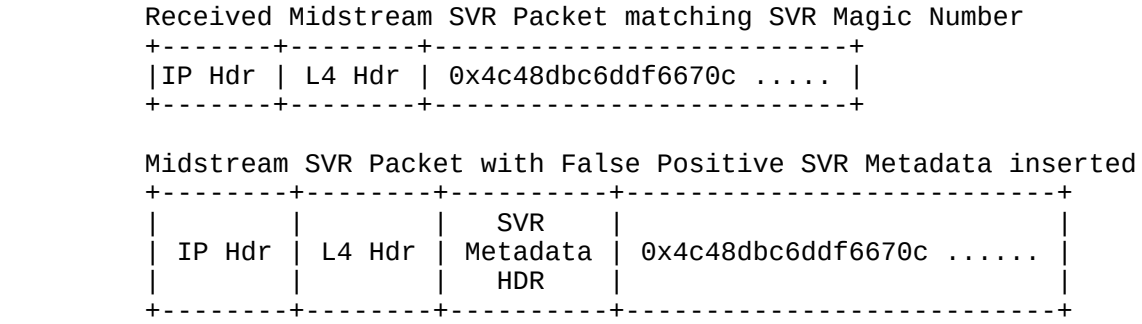


Figure 23

Header or payload TLVs MAY be added at the implementation's discretion; if added, standard processing rules apply (including encryption when payload TLVs are present).

5.2.7. TCP-to-UDP Transformation

When a middlebox blocks TCP packets that carry SVR Metadata, implementations transform the affected TCP session to UDP for the duration it crosses the middlebox and restore TCP at the egress. Pathway-test traffic typically detects the need for this. The transform proceeds as follows:

The IP protocol field MUST be changed from 0x06 (TCP) to 0x11 (UDP). The original TCP sequence number is preserved by copying it to the TCP checksum/urgent-pointer field before the UDP checksum overlays the sequence number location. TLV 12 (Section 10.4.9) MUST be added to the SVR Metadata to flag the transformation.

Once engaged, every packet of the session is transformed -- not just those carrying metadata. Restoration is described in Section 5.6.3.

5.3. Required and Optional TLVs

5.3.1. IP Session TLVs

The first forward SVR Metadata of a new session MUST contain:

* Header: Security ID: see Section 10.3.2.
* Payload: Forward Context: see Section 10.4.1, Section 10.4.2.
* Payload: Tenant Name: see Section 10.4.6.
* Payload: Service Name: see Section 10.4.7.
* Payload: Session UUID: see Section 10.4.5.
* Payload: Source Router Name: see Section 10.4.10.
* Payload: Security Policy: see Section 10.4.11.
* Payload: Peer Pathway ID: see Section 10.4.12.

Forward SVR Metadata MAY also include the following optional TLVs:

* Header: Path Metrics: see Section 10.3.7.
* Header: SVR Control Message: see Section 10.3.6.
* Payload: Session Encrypted: see Section 10.4.8.
* Payload: TCP Syn Packet: see Section 10.4.9.
* Payload: IPv4 Source NAT Address: see Section 10.4.13.
* Payload: Remaining Session Time: see Section 10.4.14.

TLV order is arbitrary, but Header TLVs MUST precede Payload TLVs. A peer MUST ignore any TLV it does not recognize.

The first reverse packet of a new session MUST contain:

* Header: Security ID: see Section 10.3.2.
* Payload: Reverse Context: see Section 10.4.3, Section 10.4.4.
* Payload: Peer Pathway ID: see Section 10.4.12.

Reverse SVR Metadata MAY also include:

* Payload: Path Metrics: see Section 10.3.7.

5.3.2. ICMP TLVs

A returned ICMP error MUST contain:

* Header: ICMP Error Location Address: see Section 10.3.4, Section 10.3.5.

It MAY also include:

* Header: Path Metrics: see Section 10.3.7.

5.4. SVR Metadata Encryption

SVR Metadata encryption uses block ciphers with a fixed block size. The cipher and its block size MUST be provisioned and known to peers in advance (provisioning is out of scope). The SVR Metadata Key is common to all Peer Pathways between two peers and is obtained via BFD with SVR Metadata (Section 6.1.9). Payload TLVs are zero-padded (0x00) up to a multiple of the block size, and a per-block IV makes decryption stateless.

SVR Metadata Block

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

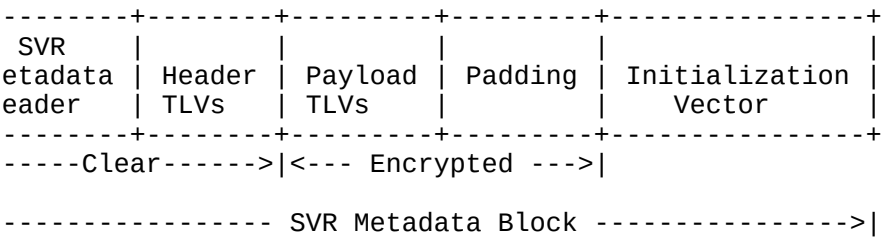


Figure 24

The required pad length is (block_size - (payload_len mod block_size)) mod block_size.

5.5. Packet Authentication

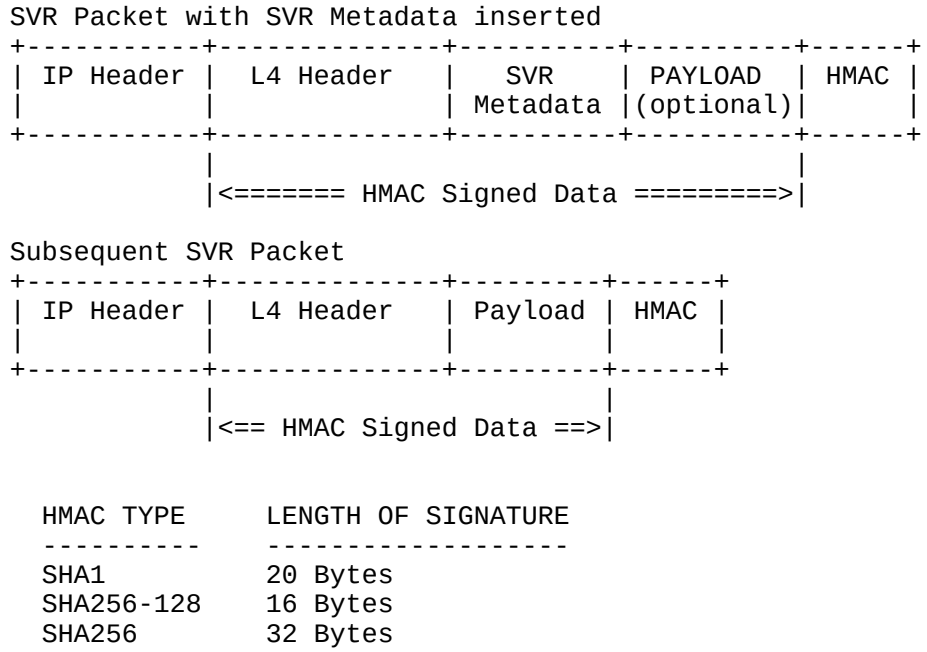
5.5.1. HMAC Signatures

An SVR Authority MUST provision (out of scope here): whether HMAC signatures are used; whether Time-Based HMAC is used; and whether ALL packets are signed or only those carrying SVR Metadata. Time-Based HMAC on ALL packets is RECOMMENDED to mitigate replay attacks. The Session HMAC Key is conveyed between peers in BFD Metadata (Section 6.1.9) and is used for the life of the session.

SVR peers SHOULD sign all packets per [RFC2104] using the Session HMAC Key. When present, an IP packet MUST contain at most one HMAC signature, even if it is fragmented. For Time-Based HMAC, the implementation appends (current_epoch_seconds / 2) to the signed data; clocks MUST be synchronized accurately, with NTP ([RFC5905]) as a baseline. Disabling Time-Based HMAC in favor of standard HMAC is NOT RECOMMENDED.

The HMAC signature is appended to the end of the packet. Its length depends on the algorithm; supported algorithms include SHA1, SHA256-128, and SHA256.

Location of HMAC Checksum



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

Figure 25

5.5.2. HMAC Verification

When HMAC signatures are in use, SVR implementations MUST verify and remove the signature on receipt. Verification provides both peer authentication and integrity protection across the previous hop. Provisioning rules (signatures present, Time-Based, ALL vs. metadata-only) are as in Section 5.5.1. Verification regenerates the signature locally and compares it byte-wise to the one in the packet, using the Session HMAC Key from the matching session state.

HMAC Signature Removed

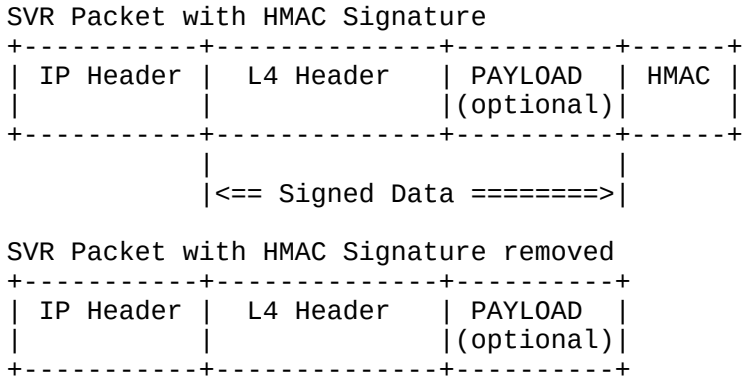


Figure 26

For efficiency with Time-Based HMAC, implementations SHOULD compute the partial HMAC over the packet body once (excluding the IP header), then attempt finalization with the current time window; on mismatch, retry with the next window (current+2 seconds) and then the previous (current-2 seconds). The active window for a given peer SHOULD be cached and advanced as time progresses.

If no time window and no recently-issued key produces a match, the packet MUST be dropped and a security event recorded.

On match, the packet is authenticated; the HMAC signature MUST be removed, the IP total-length field MUST be updated, and the IP header checksum MUST then be recomputed.

5.6. Processing Packets That May Carry SVR Metadata

SVR routers MUST process both SVR and non-SVR traffic and MUST track which sessions are using SVR. SVR-bearing traffic always uses Waypoint addresses, providing efficient ingress separation between SVR and non-SVR traffic. Packets arriving on a known Peer Pathway MUST be presumed to either contain SVR Metadata or belong to an established SVR session.

5.6.1. Detection of SVR Metadata

DPI MUST be used on every packet to detect SVR Metadata. For first packets, metadata is required and its absence causes the packet to be dropped. The HMAC verification step (above) MUST run before any further metadata processing, to prevent on-path tampering.

If the first 8 octets of the L4 payload (TCP or UDP) exactly match the SVR magic number (0x4c48dbc6ddf6670c), the packet MUST be treated as bearing SVR Metadata. If they do not match, the packet does not contain metadata; if it belongs to an existing session it SHOULD be routed (Section 5.6.4), otherwise it MUST be dropped and a security event recorded.

5.6.2. Verification of SVR Metadata

5.6.2.1. TLV Parsing

The SVR Metadata header is parsed (Section 10.1). If both the header length and payload length are zero, the metadata is simply removed and the packet forwarded -- this is the false-positive case (Section 5.2.6). The implementation then walks any header TLVs to validate them; with payload length zero, no decryption is required.

An unknown TLV SHOULD be skipped and MUST be forwarded unchanged. If any TLV value is unreasonable, the packet MUST be dropped and a security event recorded.

5.6.2.2. Decryption of SVR Metadata Blocks

If the peers have been provisioned to encrypt SVR Metadata and the payload length is non-zero, the implementation MUST treat the payload region as an encrypted block. The pre-provisioned cipher, block size, and IV size, combined with the header’s payload length, fully determine the block layout.

Encrypted SVR Metadata Block

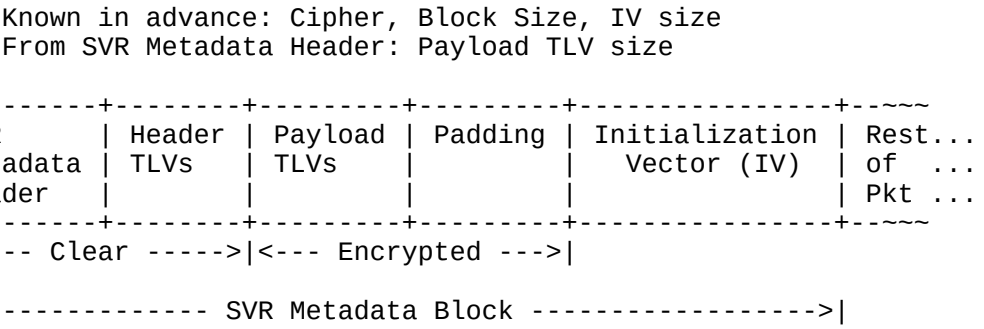


Figure 27

The pad length is (block_size - (payload_len mod block_size)) mod block_size; the IV immediately follows the encrypted block.

If decryption fails, the packet MUST be dropped and a security event recorded. On success, the payload TLVs MUST be checked for completeness against Section 5.3; insufficient or unreasonable TLVs cause the packet to be dropped with an error recorded. The metadata block is then removed and the IP total-length and header checksum MUST be updated.

5.6.3. UDP-to-TCP Restoration

If the received metadata contains a TCP SYN Packet TLV (Section 10.4.9), the following MUST be performed on every packet of the session, in both directions (see Section 5.2.7):

The IP protocol field MUST be changed from 0x11 (UDP) back to 0x06 (TCP). The 32-bit value at the TCP checksum/urgent-pointer location is copied back to the sequence-number field; the urgent pointer is zeroed and its flag cleared. The TCP checksum MUST then be recomputed.

5.6.4. SVR Session Packets

A packet whose source and destination addresses both map to a Peer Pathway is an SVR packet. Such packets without metadata are in-session traffic and MUST have matching session state; if no state exists, the packet MUST be dropped or session state MUST be restored (Section 3.11).

Ingressing in-session packets MUST be translated bidirectionally on all 5-tuple fields: source address to the local Waypoint, destination address to the chosen peer’s Waypoint, ports to the allocated SVR ports, and protocol modified if UDP transformation applies (Section 5.2.7). Implementations SHOULD cache a single checksum delta per session, since the rewrite is identical for every packet.

Egressing in-session packets MUST have the original 5-tuple restored from saved session state (source/destination addresses, ports, and protocol -- with UDP-to-TCP restoration per Section 5.6.3 if applicable).

5.6.5. Tenant and Service Overview

A provisioned SVR Policy SHOULD include both a Tenant and a Service. Absence of an applicable SVR Policy SHOULD prevent SVR session establishment; traditional IP routing MAY be used when no SVR policy applies.

5.6.5.1. Service Interpretation

A Service is a textual name for a set of CIDR blocks, protocols, and ports (e.g., "Zoom", "Office365").

Service Definition

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

Figure 28

A packet arriving with SVR Metadata MUST carry the Service name in the first-packet metadata. A packet arriving without SVR Metadata is classified by destination address/port/protocol lookup; if that fails, application-recognition techniques (HTTP request inspection, TLS SNI, certificate Common Name) MAY be used. These techniques are out of scope for this document.

Services MAY have associated quality and security policies provisioned (out of scope here).

At SVR egress, the Service name MAY be used to forward to another SVR peer (in which case it MUST be carried unchanged) or to resolve to a final IP destination by one of:

Use Destination from Context (default): Restore the original destination address and forward.

Destination NAT: Map the Service to one or more local IP addresses (load-balancing to Service instances; common in public clouds).

DNS Resolution: 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 useful in public clouds.

Services SHOULD be provisioned with allow/deny lists of Tenants; these access controls are RECOMMENDED.

5.6.5.2. Tenant Determination and Interpretation

A Tenant is a period-delimited textual hierarchy, logically akin to a VLAN, CIDR subnet, or security zone. Policy match is performed right-to-left on full segments, with the longest segment match winning.

A Tenant SHOULD be paired with a Service to form a from-to vector (allowing ACLs to be tied directly to destinations). A provisioned SVR Policy SHOULD include both; absence of an applicable policy prevents session establishment. Default-deny is RECOMMENDED.

It is RECOMMENDED that Tenants be associated with physical and logical (VLAN) interfaces as defaults; CIDR-block-based Tenants SHOULD override these defaults; client self-asserted Tenants SHOULD override all other definitions.

Interface-based Tenant assignments are local to each SVR router; ingress and egress Tenant identities need not match, allowing heterogeneous segmentation across networks.

5.6.6. Payload Encryption

When payload encryption is required, a Security Policy specifies the agreed methods. The policy semantics MUST be valid and identical at the points of encryption and decryption (relevant for multi-hop SVR routes).

An SVR router generates a Payload Key locally on the first packet requiring encryption per the policy. Keys and IVs MUST be generated using a FIPS 140-3-approved DRBG; following NIST SP 800-90B for entropy is strongly RECOMMENDED (see [NIST_SP_800-90B]). OpenSSL’s RAND_bytes() is one suitable option. The key is conveyed in the first encrypted packet’s SVR Metadata. Future implementations MAY source key material from quantum entropy.

The metadata is forwarded hop-by-hop until SVR egress; the egress router MUST extract the Payload Key and store it in session state for decrypting all subsequent payload packets in that direction. The reverse direction uses an independently generated key, carried in reverse metadata. Asymmetric per-direction keying simplifies key management and avoids glare.

The originator MAY rekey at any time by inserting new SVR Metadata bearing a new key; the new key takes effect on the carrying packet.

The Security KEY TLV (Section 10.4.15) carries the encryption/ decryption key in the first-packet metadata for each direction. End-to-end key conveyance is safe because SVR Metadata is encrypted hop-by-hop; payload decryption occurs only at the final SVR hop. Named Security Policies allow any cipher; the default is AES-256 with a 256-bit key.

Payload Encryption:

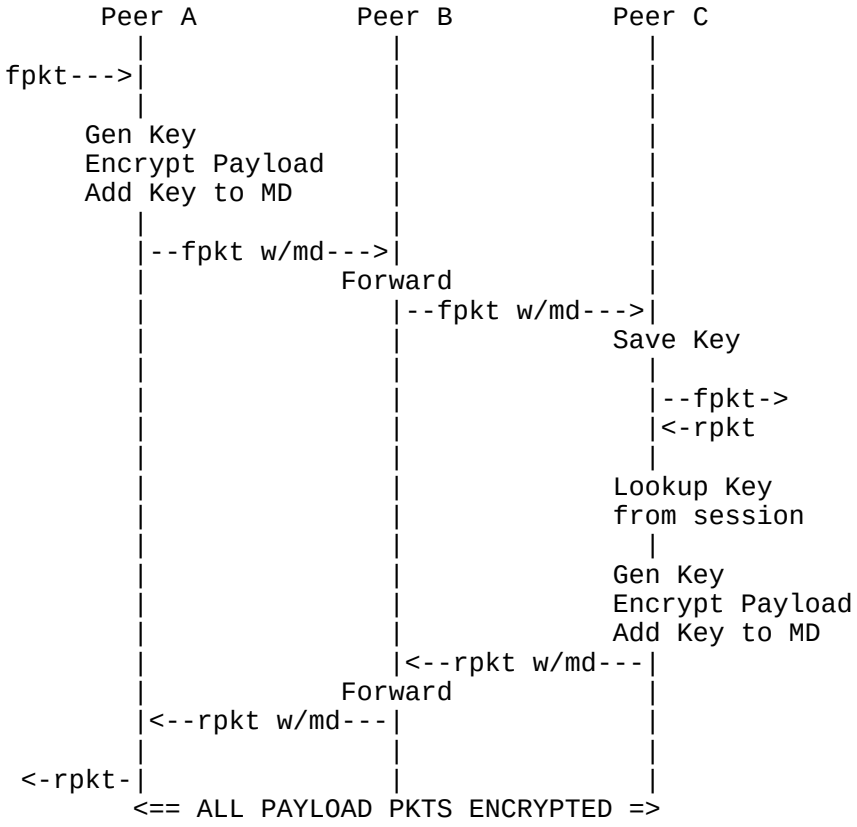


Figure 29

6. BFD for Peer Pathways

Peer Pathways are virtual transport links between routers, analogous to tunnels. SVR uses BFD for reachability, quality measurement, peer authentication, and key management.

6.1. SVR Peering and BFD

It is RECOMMENDED that every configured or discovered Peer Pathway run a UDP BFD session for liveness and quality measurement.

BFD timers per pathway are administratively determined. Because BFD ([RFC5880]) does not natively support certificates or public-key exchange, SVR carries this information in BFD Metadata appended to BFD control messages. BFD Metadata is used to:

* discover the Peer Received IP address;

* detect NATs on a Peer Pathway;

* detect changes to a router’s Peer Received IP address;

* determine pathway MTU;

* measure path quality during idle periods (active-flow measurement uses Path Metrics, Section 10.3.7);

* detect failover to a redundant link;

* authenticate peers via certificate exchange; and

* derive a Peer Key for SVR Metadata Key encryption.

BFD Metadata is appended after the BFD control message; the IP, UDP, and BFD length fields MUST be adjusted accordingly.

BFD Metadata Location:

BFD Control Packet with BFD Metadata

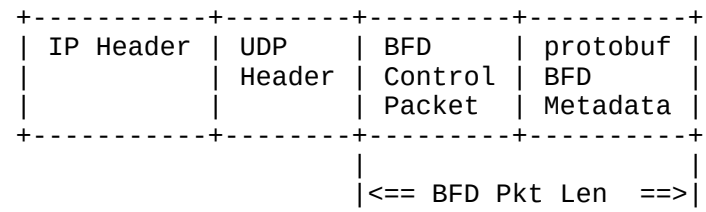


Figure 30

All messages are BFD Control packets. "MeasureData" messages behave like BFD Echo packets and require Required Min Echo RX Interval ([RFC5880]) to be greater than zero.

BFD Metadata is encoded in protobuf as follows:

BFD Metadata Protobuf Definition:

```
syntax = "proto2";
package pb.bfd;
import "ip.proto";

message SessionData {
    required ip.Tuple original_ipTuple = 1;
    required ip.Tuple received_ipTuple = 2;
    optional string peername           = 3;
    optional string routername         = 4;
    optional string routerID           = 5;
}

message MeasureData {
    message Request {
        required uint32 transId = 1;
    }
    message Response {
        required uint32 request_transId  = 1;
        required uint32 response_transId = 2;
    }
    oneof type {
        Request  request  = 1;
        Response response = 2;
    }
```

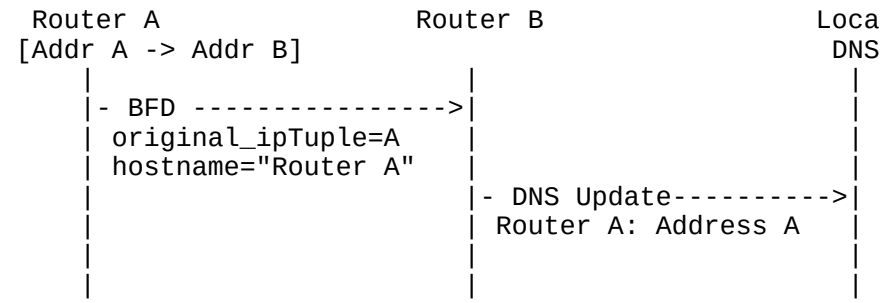
```
        optional bool mtu_discovery = 3;
    }

    message NodeInfo {
        required uint32 id               = 1;
        required uint64 create_timestamp = 2;
        optional uint64 time_value       = 3;
        optional string nonce            = 4;
        optional string public_key       = 5;
        optional uint32 salt             = 6;
    }

    message Encrypted {
      optional NodeInfo node_info          = 1;
      optional string   metadata_key       = 2;
      optional uint32   metadata_key_index = 3;
      optional string   hmac_key           = 4;
    }

    message Metadata {
        optional SessionData sessionData = 1;
        optional MeasureData measure     = 2;
        optional NodeInfo    nodeInfo    = 3;
        optional Encrypted   encrypted   = 4;
    }
```

Figure 31

6.1.1. Discovering the Peer’s Received IP Address

SessionData carries the source address a remote peer observes for this router on a Peer Pathway. This is required for pathway establishment because configuration references router IDs rather than dynamic addresses; remote peers maintain a local resolution table (e.g., /etc/hosts) keyed by the configured hostname. This step can be combined with NAT detection (below).

Determination of Peer Received Address:



Router B has hostname lookup for Router A

Figure 32

6.1.2. NAT Detection

SessionData also detects NATs on a pathway, typically during initial peer establishment (and often combined with certificate exchange). Like STUN, the originating interface address is placed in SessionData.original_ipTuple; on receipt, a router stores the observed source address from the IP header in SessionData.received_ipTuple. Comparing the wire address against the peer's claimed original_ipTuple reveals any intervening NAT. The peername field MAY also be set.

BFD NAT Detection on Pathway:

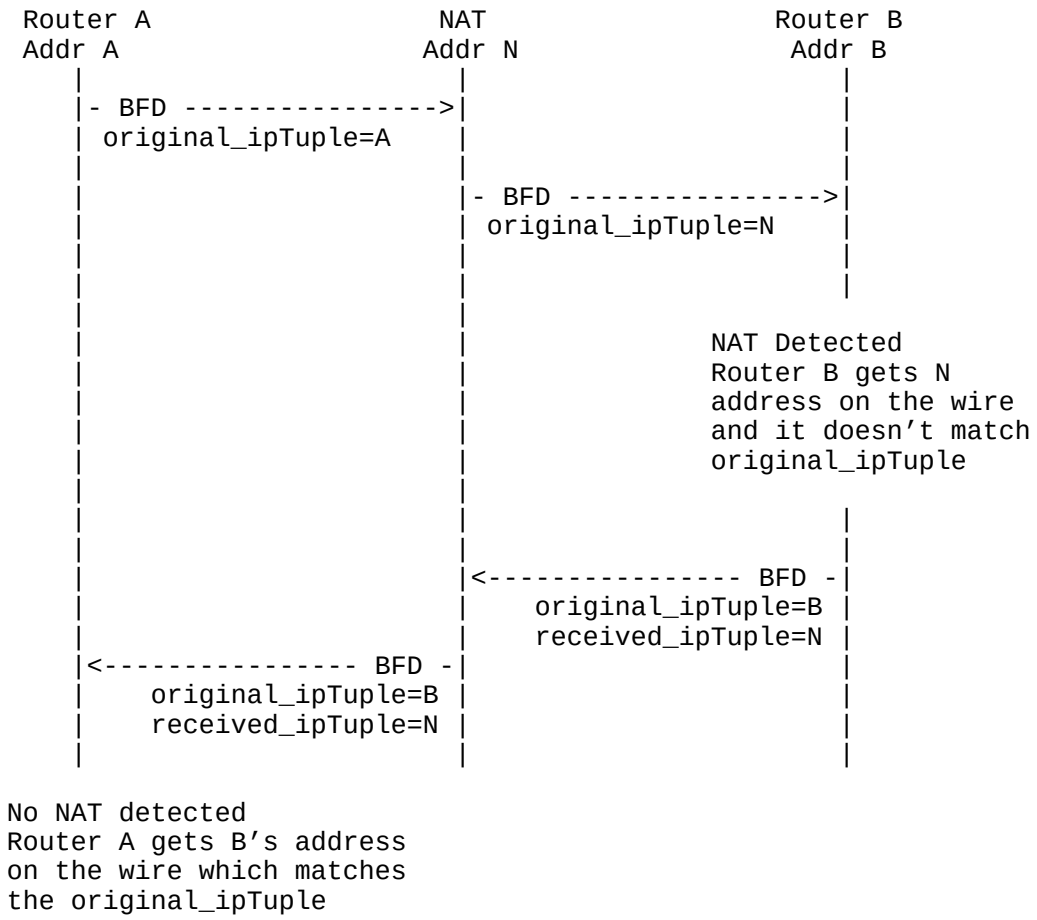


Figure 33

When NAT is detected, the NAT-side address MUST be associated with the pathway to the far peer. Sessions traversing such a pathway may require NAT keep-alive processing (Section 7.3).

6.1.3. Detecting Router Address Changes

Branch routers commonly receive their addresses dynamically (DHCP, LTE, PPPoE) and the address may change unexpectedly (e.g., lease expiry or reconnection). Without intervention, peers using the old address would lose connectivity. SessionData BFD Metadata makes learning the new address and recovering connectivity fast.

BFD Detection on Router Address Change:

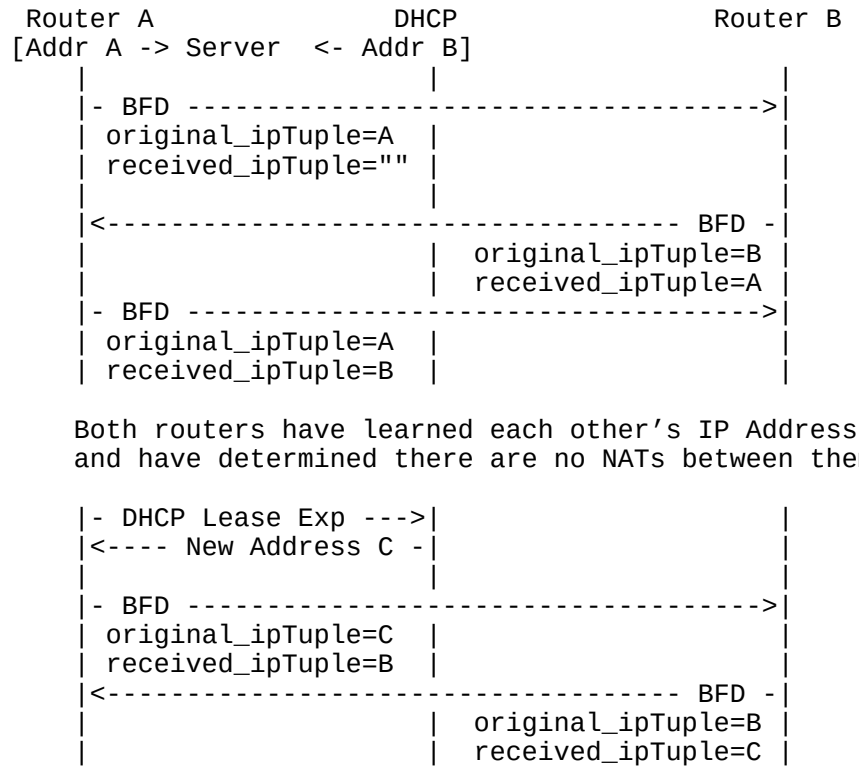


Figure 34

6.1.4. MTU Discovery

Knowing the pathway MTU lets routers fragment when necessary. After a pathway is established, BFD MeasureData packets of increasing size probe for the limit; the IP, UDP, and BFD length fields are adjusted to enlarge the BFD packet. A peer that receives a fragmented MeasureData request with mtu_discovery=TRUE simply does not respond, signaling that the size exceeds the path MTU.

Because networks change, MTU SHOULD be remeasured periodically.

BFD MeasureData for Determining Pathway MTU:

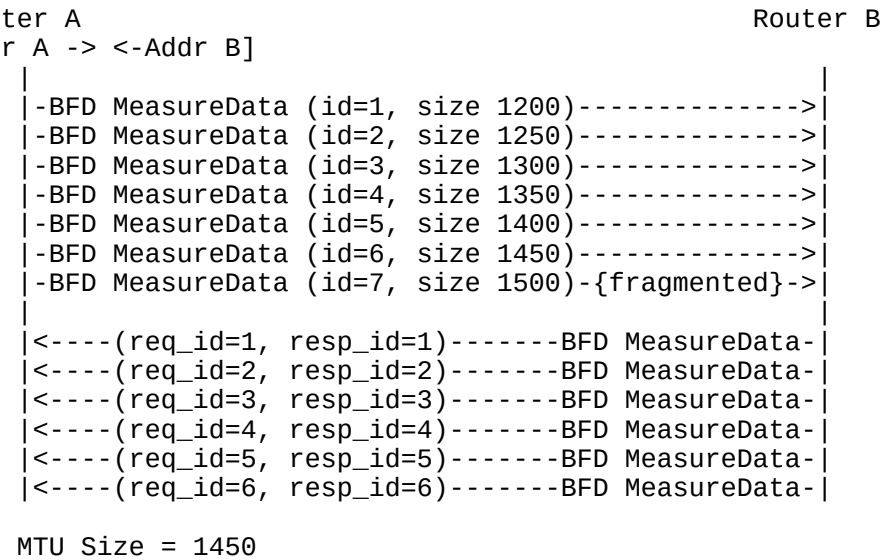


Figure 35

6.1.5. Path-Quality Measurement

Once a pathway is operational, BFD MeasureData packets are used to measure latency, jitter, and loss. Either side MAY measure; burst size and frequency are configuration. Hub routers with many pathways often rely on spoke-side measurements rather than measuring themselves.

BFD-based measurement is useful only when a pathway is idle. For pathways carrying live sessions, SVR Path Metrics are used (Section 10.3.7).

The receiver responds to each request by echoing it with its own transaction ID. Each request and each response carry a transaction ID, eliminating any ambiguity between requests and responses.

BFD MeasureData for Measuring Pathway Quality:

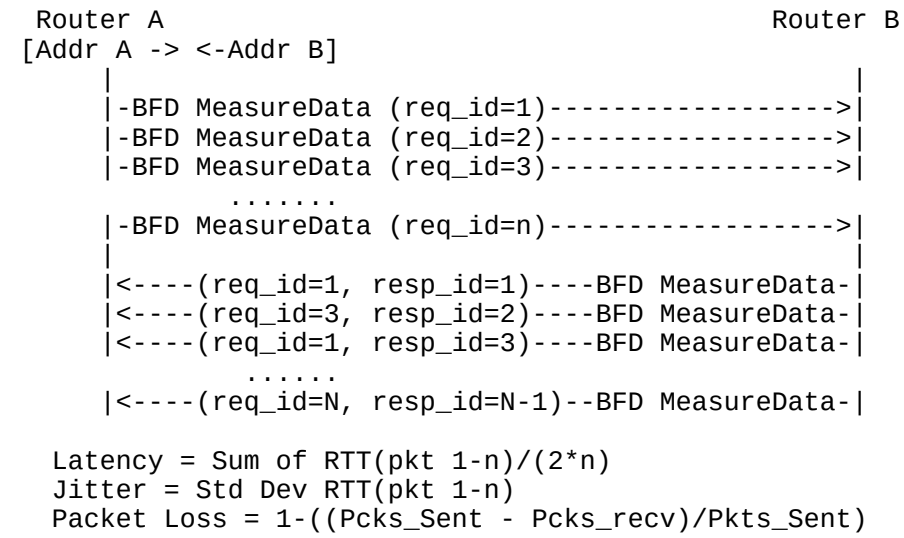


Figure 36

The sender derives latency from the round-trip times of matching request/response IDs; missing responses count as packet loss; the distribution of RTTs gives jitter; MoS scores follow from latency, loss, and jitter together. Each direction is measured independently because network behavior may be asymmetric.

6.1.6. Failover Detection

BFD NodeInfo carries a node ID (cluster member instance) and a peer start timestamp. Changes to either signal cluster failover or peer-side restart, allowing remote peers to react to redundancy events on the far side of a pathway. Inclusion of this information is OPTIONAL.

6.1.7. Peer Authentication

A router lacking a valid signed certificate MUST obtain one from a CA. The router generates an elliptic-curve key pair ([RFC8422]) -- elliptic curves are used to keep the X.509 certificate small -- and submits an X.509 CSR with the router’s UUID as the Common Name. The CA returns an ECDSA-signed certificate. Detailed enrollment procedures are out of scope but SHOULD follow [RFC4210]. Certificates and public keys are stored locally in PEM format ([RFC7468]).

Router Authentication:

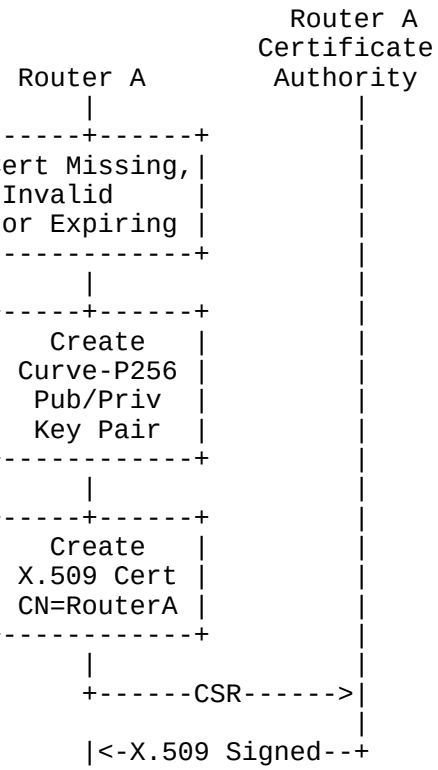


Figure 37

The certificate is persisted on the router in PEM format. The associated private key SHOULD be created and stored in secure non-volatile storage such as a TPM.

During pathway establishment, peers authenticate using their UUIDs and public keys, then derive a symmetric Peer Key from their X.509 key material. UUIDs are used (rather than IP addresses) because router addresses commonly change (e.g., DHCP lease expiry on branch routers).

The diagram below shows two routers with two pathways each exchanging X.509 certificates in BFD NodeInfo public_key fields. Certificates are exchanged on every pathway but validated only once per peer.

Router Authentication:

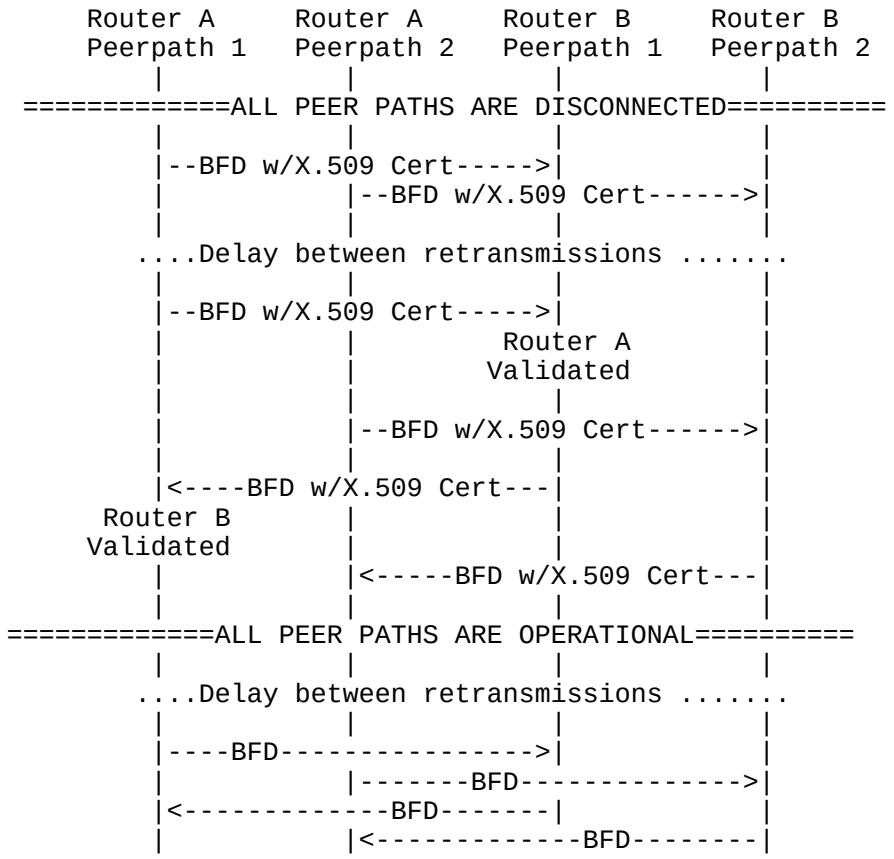


Figure 38

A received certificate MUST be validated:

* dates are within validity;

* CA signature verifies;

* if a CRL is available, the certificate is not revoked; and

* the router name appears in the local configuration (administrative revocation is the primary control).

Validation runs once per peer; subsequent receipts of the same certificate on other pathways MAY use a cached result. After validation, the receiver extracts and stores the peer's public key for use in Peer Key/rekey authentication.

A router SHOULD renew its certificate using the same procedure before expiry.

6.1.8. Peer Key and Rekey

A single Peer Key is used across all pathways between two router peers and remains valid until replaced. The key persists across network outages. If the key is lost or fails, a new key MUST be established before encrypted BFD traffic can resume.

To establish or replace a key, the initiator generates a salt and sends it in BFD NodeInfo; the peer responds with its own salt in BFD NodeInfo. With both salts, each side runs Concat KDF to derive a symmetric Peer Key.

Concat KDF ([NIST_SP_800-56A]) uses the routers' authenticated certificate private keys, providing man-in-the-middle resistance.

OpenSSL provides ConcatKDF() with the following parameters:

ConcatKDF Function (Part of OpenSSL):

```
Peer Key = ConcatKDF(SharedSecret,
                     AlgorithmID,
                     PartyUInfo,
                     PartyVInfo,
                     SuppPubInfo,
                     SuppPrivInfo,
                     KeyDataLen)
```

Here's what each parameter represents:

SharedSecret: The result of an ECDH calculation with the peer
AlgorithmID: "ECDH"
PartyUInfo: UUID of the Router
PartyVInfo: UUID of the Peer Router
SuppPubInfo: Initiator Salt Concatenated with Responder Salt
SuppPrivInfo: ""
KeyDataLen: 256

SuppPubInfo is the initiator salt concatenated with the responder salt; SharedSecret is the result of an ECDH exchange ([ECDH_Key_Exchange]).

After a brief guard period (1-2 s) to allow both sides to complete their computation, the Peer Key is active across all pathways between the two peers and replaces any prior key.

The key MAY be used immediately to encrypt BFD Metadata.

Peer Key-Rekeying:

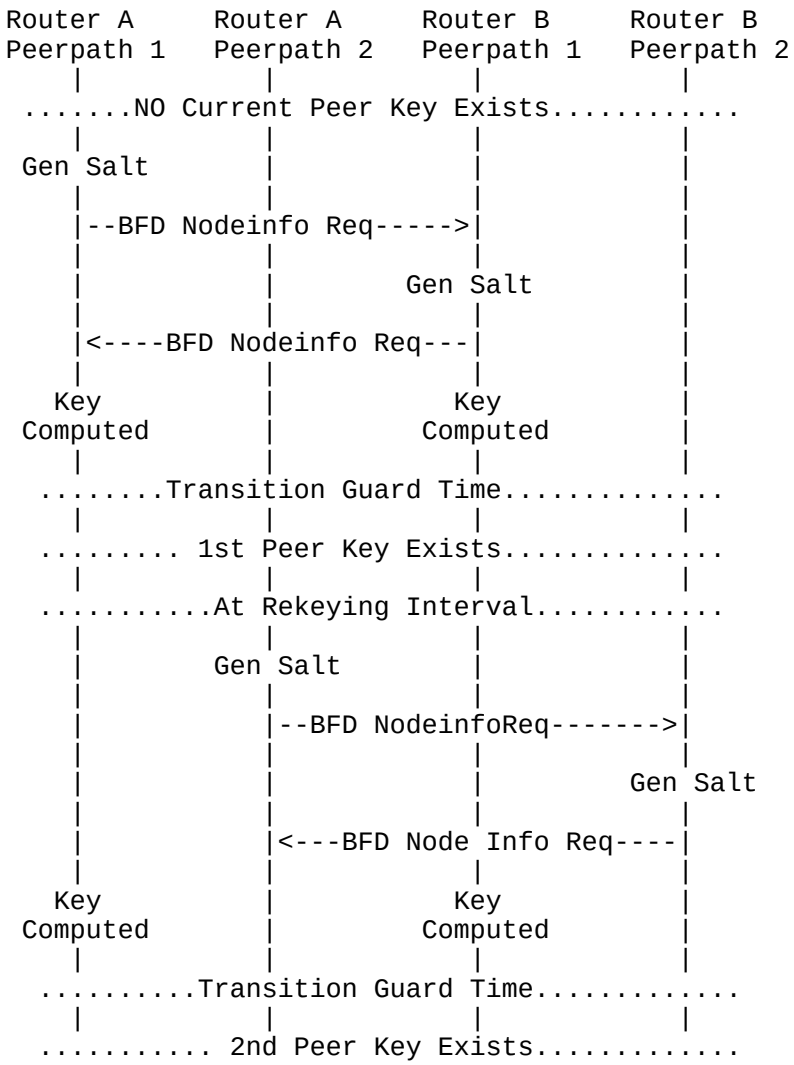


Figure 39

The Peer Key is symmetric and is used to encrypt the SVR Metadata Key exchange. Concat KDF (a form of ECDH-ES) yields a symmetric key only if there is no man-in-the-middle; if peers cannot decrypt each other's messages, an MITM is likely and the affected pathway SHOULD be removed from service.

If a BFD NodeInfo elicits no response, the sender retries periodically; after an extended interval (e.g., 1 hour) with no response, the pathway MAY be declared invalid and removed from service per administrative timers.

6.1.9. SVR Metadata Key and Rekey

The SVR Metadata security association is not pathway-specific: multiple pathways may share an interface or source address (e.g., over the public Internet), so peer identity can only be established by decrypting the metadata. Each SVR router MUST therefore be able to decrypt SVR Metadata arriving on any interface.

SVR Metadata is encrypted for the chosen next-hop SVR router; no other router should be able to decrypt it. Senders MUST select the key associated with the chosen next-hop peer.

Keys are generated locally per the security policy using a FIPS 140-3-approved DRBG (NIST SP 800-90B for entropy is RECOMMENDED; see [NIST_SP_800-90B]). OpenSSL’s RAND_bytes() is suitable for producing a 256-bit key.

The key and its index are distributed to all known peers in an Encrypted BFD NodeInfo message. The 256-bit SVR Metadata Key is encrypted with the current Peer Key, producing a 32-octet ciphertext plus a 16-octet IV (48 octets total). The key index is incremented on each rekey. Encryption follows the SVR Metadata encryption procedure (Section 4.7.1.8) but uses the Peer Key. SVR Metadata keys are asymmetric: forward metadata is encrypted with the destination router’s key, reverse metadata with the originating router’s key.

SVR Metadata Key/Rekeying:

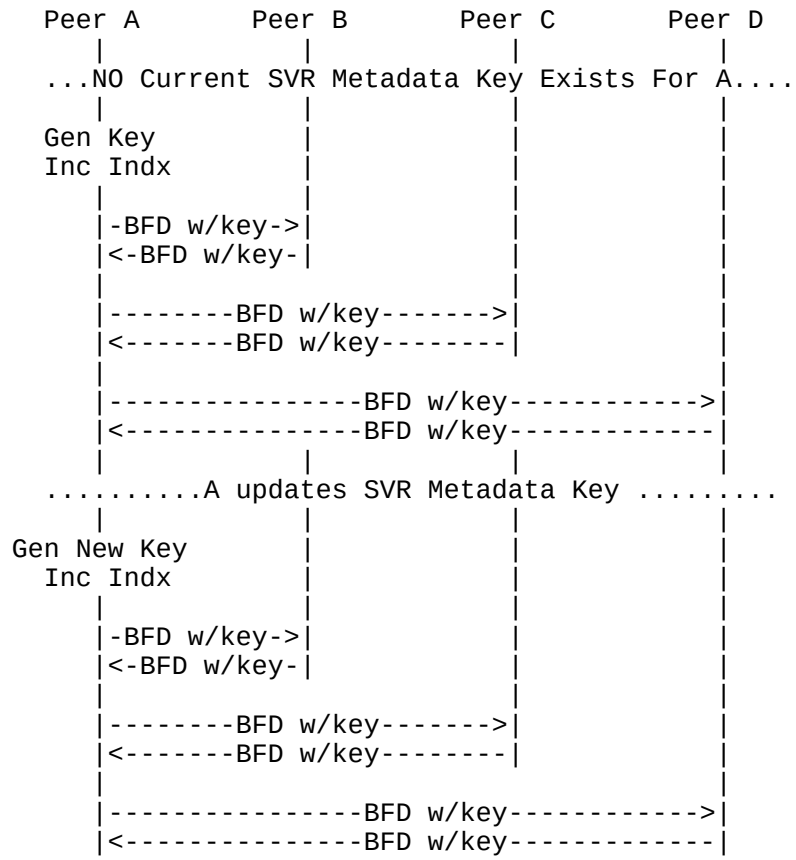


Figure 40

The diagram shows Peer A distributing its key/index to Peers B, C, and D, then later rekeying. Each peer responds with its own current key/index, providing a handshake; this is also how a restarting router quickly acquires every key it needs. A router that needs to send SVR Metadata to a peer for which it lacks a key MAY use this procedure to obtain it. Whenever a peer rekeys, it MUST update all of its peers. Stored peer state pairs each shared key with its metadata_key_index; the Security ID field in incoming SVR Metadata identifies which key to use for decryption (Section 10.3.2).

6.1.10. Certificate Revocation and Replacement

A new certificate loaded onto the router triggers re-execution of the peer authentication procedure (Section 6.1.7). The mechanism for loading the certificate is out of scope.

If a system is compromised, its certificate SHOULD be revoked. The router's management platform SHOULD periodically check the CRL, or use OCSP to validate certificates directly. On revocation, the router consults configured policy to choose its behavior.

A policy SHOULD govern behavior on certificate expiry or revocation. The default SHOULD be fail-soft (signal that action is required). A fail-hard configuration MUST tear down all peering relationships, removing the router from SVR participation.

7. Additional SVR Metadata Use Cases

The following examples illustrate additional uses of SVR Metadata. They are not exhaustive.

7.1. Moving a Session

Any SVR router MAY move an existing session onto a different Peer Pathway by re-emitting the session-establishment metadata of Section 4.7.1.7 on the new pathway while preserving the Session UUID. It simultaneously updates fast-path forwarding for the new Waypoints/ports; everything else about the session is unchanged. Both endpoints MUST then redo NAT detection on the new pathway and update wire addresses/ports as needed.

Old and new fast-path entries are kept for ~5 seconds to avoid dropping in-flight packets, after which the old state is removed.

Ladder Diagram for Existing Session Reroute with SVR Metadata:

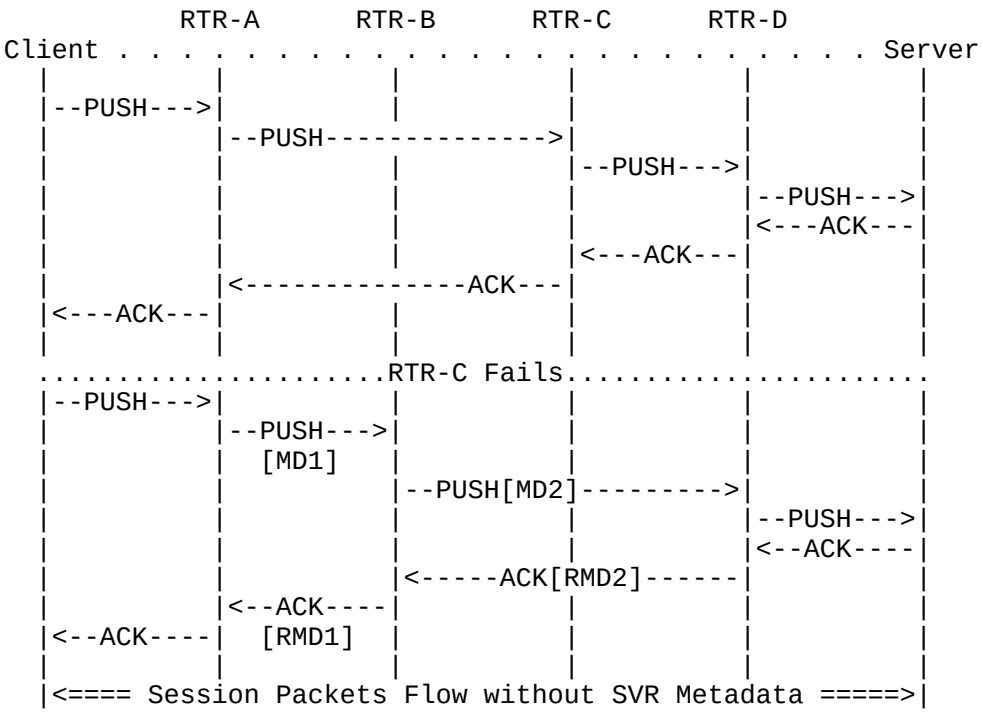


Figure 41

When Router C fails, Router B inserts MD1/MD2 in the next packet in either direction; RMD1/RMD2 confirm the move. For TCP this can be a PUSH (shown) or an ACK. The technique installs SVR session state on Router B even though it had no prior involvement in the session, and can equally be used to migrate a session between transports (e.g., MPLS to LTE). A move can be initiated by any router at any time.

Ladder Diagram for Session Reroute Between Peers with SVR Metadata:

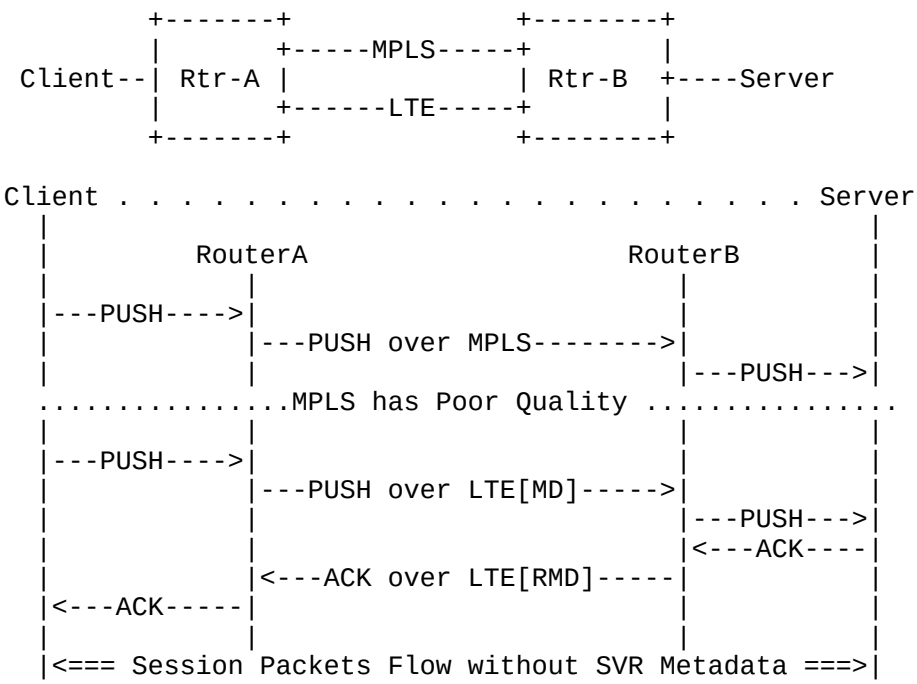


Figure 42

The diagram shows an active TCP session migrated from MPLS to LTE by inserting metadata on any in-flight packet; reverse meta data on the reverse direction confirms the move.

7.2. Moving Idle or Unidirectional Sessions

Idle sessions and one-way flows (TCP pushes with bare ACKs) may not offer a packet into which metadata can be piggybacked.

If, after a move, no packets are seen for 1 second on an active session, the SVR router MUST emit an SVR Control Message to its peer.

Ladder Diagram for One Way Media Move with SVR Metadata:

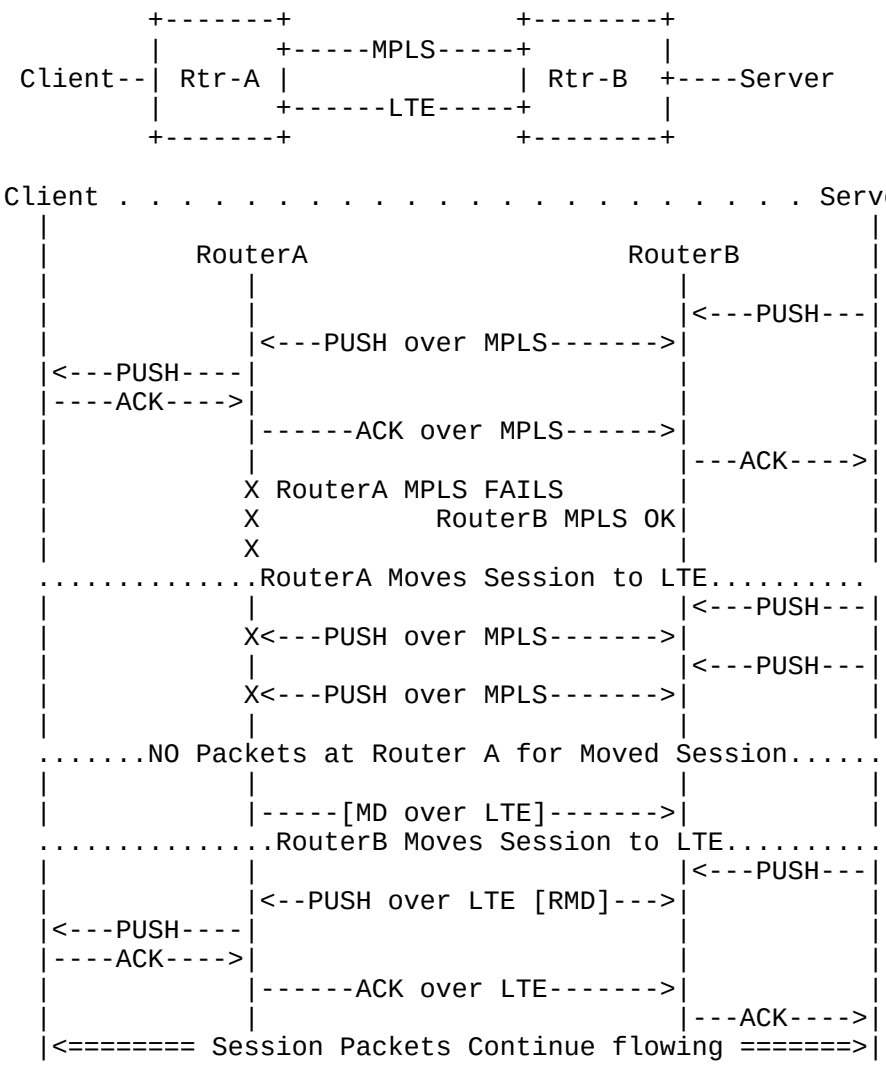

Rtr-A
MPLS
Rtr-B
LTE
Client
Server
RouterA
RouterB
PUSH over MPLS
ACK over MPLS
X RouterA MPLS FAILS
RouterB MPLS OK
RouterA Moves Session to LTE
NO Packets at Router A for Moved Session
[MD over LTE]
RouterB Moves Session to LTE
PUSH over LTE [RMD]
ACK over LTE
Session Packets Continue flowing

Figure 43

The SVR Control Message uses the new Waypoint addresses and carries the standard first-packet TLVs plus an SVR Control Message TLV with drop reason "FLOW MOVED" and a Remaining Session Time TLV (so that intermediate routers age the session out roughly together). It is sent once per second for up to 5 seconds, or until reverse metadata arrives; if no reverse metadata arrives within 5 seconds, the session is torn down. For an idle flow the reverse metadata is itself a generated SVR Control Message:

Ladder Diagram for Quiescent Moved Session with SVR Metadata:

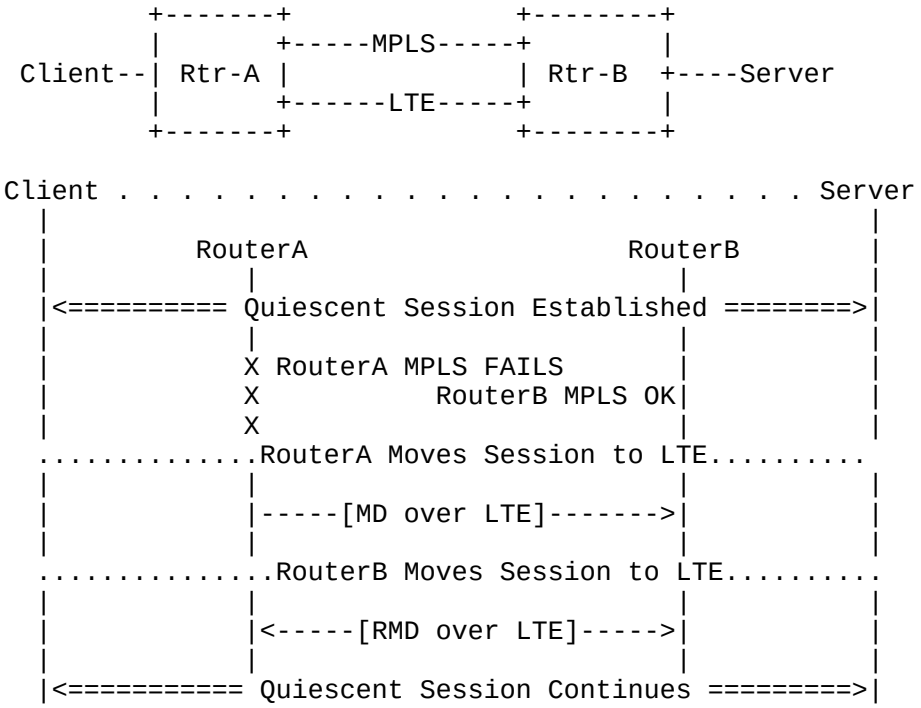


Figure 44

7.3. NAT Keep-Alive

When a Peer Pathway has one or more NATs (Section 3.5), each side MUST refresh the NAT bindings for each active session by sending a keep-alive packet that matches the idle session’s L4 header and carries SVR Metadata with TLV 24 and drop reason "Keep Alive".

Ladder Diagram for NAT Keep Alive with SVR Metadata:

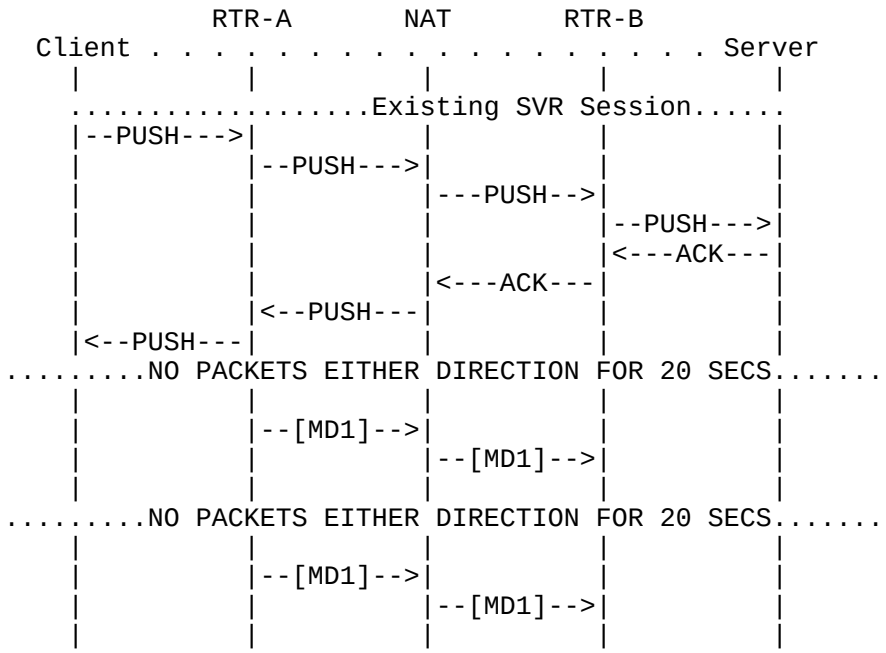


Figure 45

A keep-alive MUST contain:

* Header: SVR Control Message: see Section 10.3.6.

* Header: Security ID: see Section 10.3.2.

* Payload: Session UUID: see Section 10.4.5.

* Payload: Source Router Name: see Section 10.4.10.

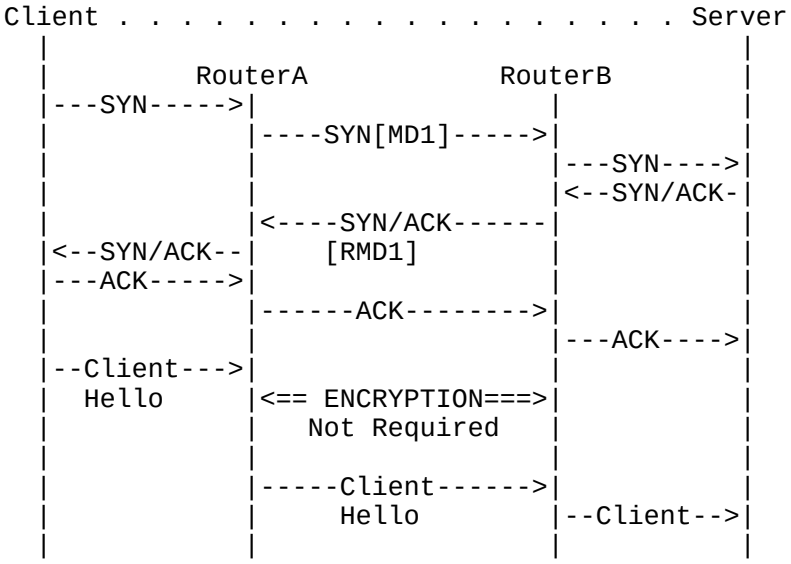
* Payload: Peer Pathway ID: see Section 10.4.12.

This minimal set lets the receiver verify the session (by UUID) and update any NAT-state changes.

7.4. Adaptive Encryption

Unlike a tunnel, each SVR session is independent, so SVR can encrypt only those sessions that are not already encrypted by the application. With adaptive encryption every session begins unencrypted; by inspecting the first four packets, the router determines whether application-layer encryption is in use (e.g., a TLS Client Hello). If yes, no SVR encryption is applied; otherwise SVR encryption is enabled bidirectionally.

Ladder Diagram of Adaptive Encryption with Client Hello:



If the fourth packet does not indicate transport-layer encryption, the ingress and egress SVR routers MUST encrypt and decrypt the session bidirectionally, ensuring all data between SVR routers is encrypted.

Ladder Diagram of Adaptive Encryption with data:

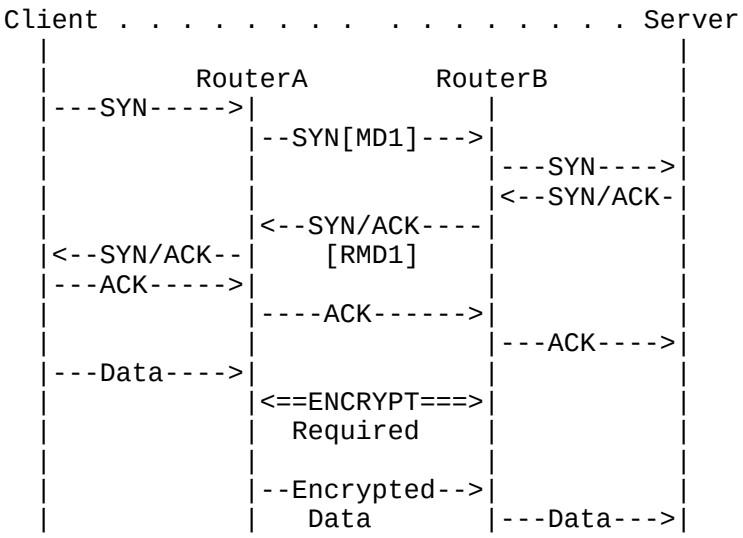


Figure 47

Adaptive encryption is configured via Security Policy. Policies are bound to Services, so different Services may mandate, allow, or disallow encryption.

7.5. IPv4 Packet Fragmentation

A fragmented packet arriving at an SVR router MUST be fully reassembled before processing, since HMAC and routing apply to the whole packet. The router then re-fragments as needed onto the Peer Pathway, copying the Peer Pathway’s L4 header onto every fragment and inserting SVR Metadata identifying the fragment to the downstream peer. The Time-Based HMAC covers the entire packet and is appended to the last fragment.

Ladder Diagram Fragmented Packets:

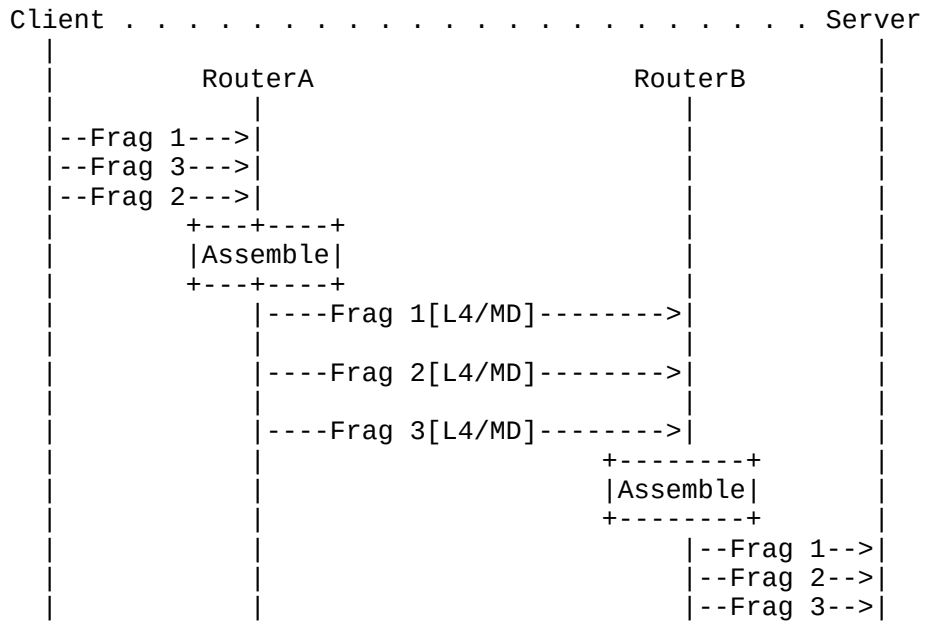


Figure 48

Router A reassembles, recording the original ID and largest fragment size. It then transmits the reassembled jumbo packet, re-fragmenting onto the chosen Peer Pathway with the pathway's L4 header on each fragment. The fragment size is min(path-MTU, largest-fragment-seen). The SVR Metadata header and header TLVs are not encrypted; the per-fragment layout is:

SVR Packet Layout

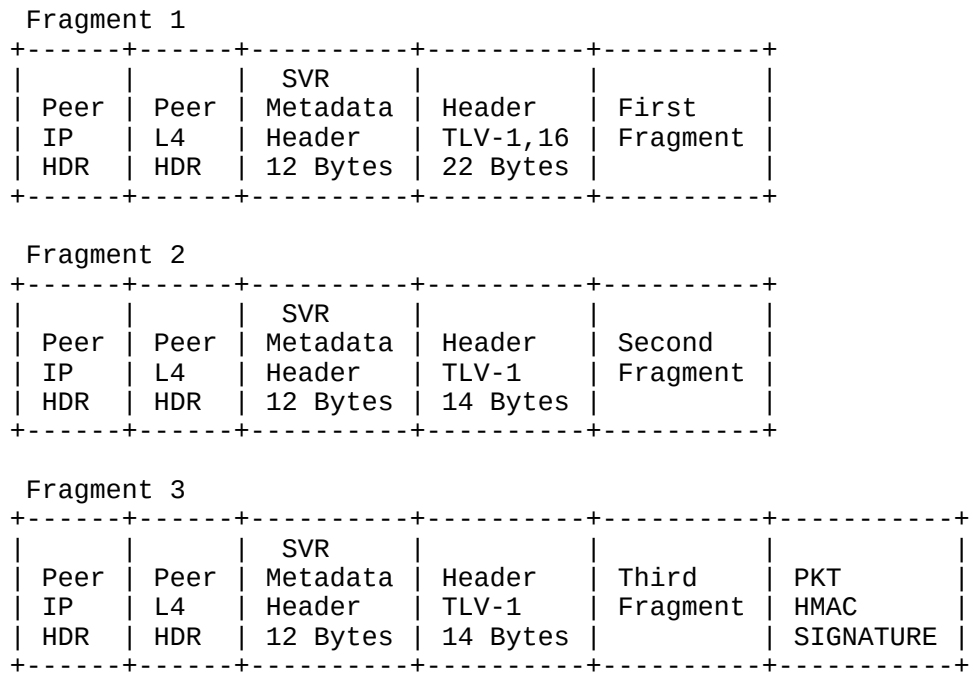


Figure 49

The SVR Metadata Type 1 header gets its own extended ID, allowing the number of fragments between Router A and Router B to differ from the client/server fragmentation. Router B re-fragments to its egress MTU using the Largest Fragment Seen value carried in the metadata.

No other SVR Metadata fields are required; session state is keyed by the Peer Pathway 5-tuple. The fragmentation mechanics are otherwise standard IP fragmentation.

If a router itself needs to fragment a packet (e.g., to make room for SVR Metadata), it inserts SVR Metadata on the first fragment, duplicates the L4 header on the remaining fragments, and inserts SVR Metadata on each. In this case, Largest Fragment Seen and Original ID are left blank.

Ladder Diagram Fragmented Packets:

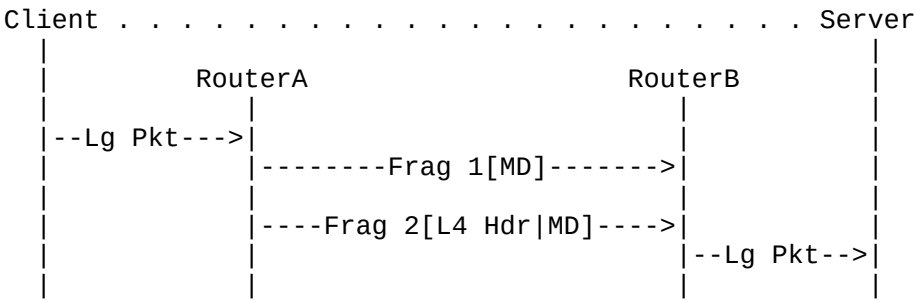


Figure 50

7.6. IPv6 Packet Fragmentation

IPv6 fragments only at the source, so SVR routers perform no special handling. An oversized packet elicits an ICMPv6 "Packet Too Big" from the destination, which the routers forward back along the reverse flow.

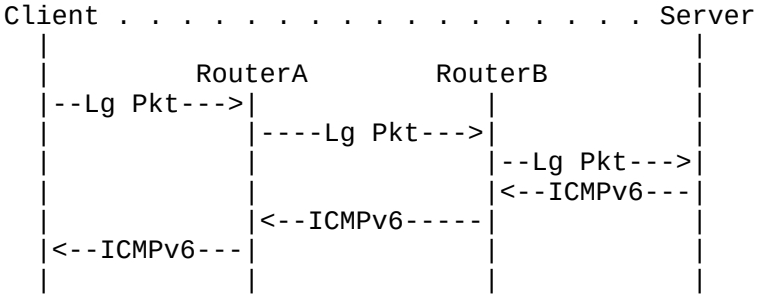


Figure 51

7.7. ICMP and SVR

ICMP messages fall into two categories. The first is session-associated network errors:

* Type 3: Destination Unreachable
* Type 11: Time Exceeded

These carry the original IP header plus 8 octets ([RFC0792]) and MUST be returned to the session originator. The router parses the embedded IP header to identify the session, then forwards the ICMP message back across the SVR network as reverse SVR Metadata, with the destination set to the upstream peer's Waypoint. SVR Metadata Types 20/21 (Section 10.3.4, Section 10.3.5) carry the original ICMP source address hop-by-hop until the ingress router recreates the ICMP message with the correct source address.

SVR Fragment Packet Layout

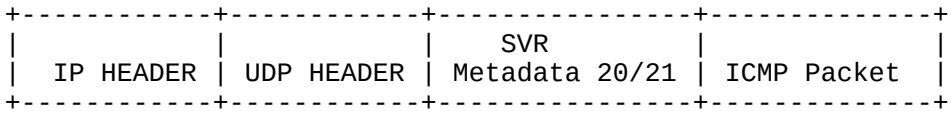


Figure 52

ICMP over SVR for Network Failures

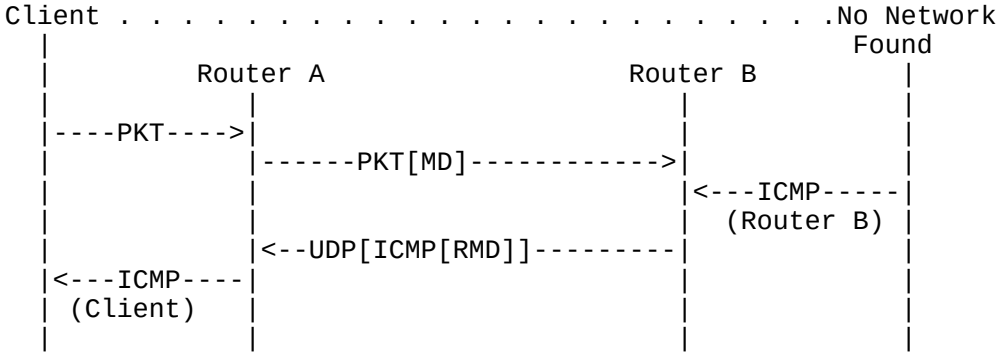


Figure 53

Router B receives the ICMP error, looks up the session, and forwards back to Router A's Waypoint; Router A recreates the ICMP message and delivers it to the client.

The second category is session-independent ICMP (e.g., ICMP Echo). SVR encapsulates these as UDP and creates a session keyed by the ICMP identifier and IP addresses:

* Type 8: Echo Request (Ping)

ICMP over SVR for Information

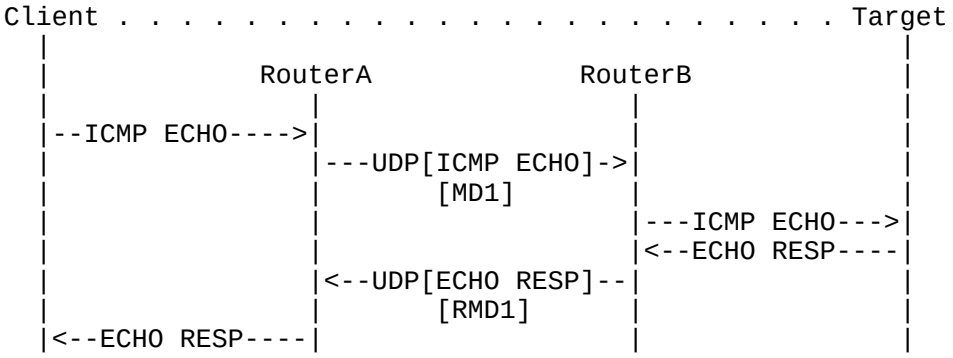


Figure 54

The first Echo Request creates a UDP-encapsulated session at Router A; MD1/RMD1 establish the bidirectional path. Subsequent ECHO messages reuse the path without further metadata. Session state MUST be aged with an inactivity timer.

ICMPv6 is handled identically; only header sizes/types differ.

7.8. State Recovery

Session state can occasionally be lost. Most applications recover on their own, but long-lived idle sessions (e.g., SIP on idle handsets) may need explicit state recovery.

Every SVR session has at least one router holding full state. Recovery techniques include obtaining state from a peer or regenerating it from session-establishment metadata.

The simplest case is loss at the ingress router: a fresh session is created using the original parameters; first-packet metadata then reaches the egress, which updates state, restoring the bidirectional flow. The first-packet metadata is signed and encrypted, so this recovery is secure.

State Recovering Ingress Router with Active Session

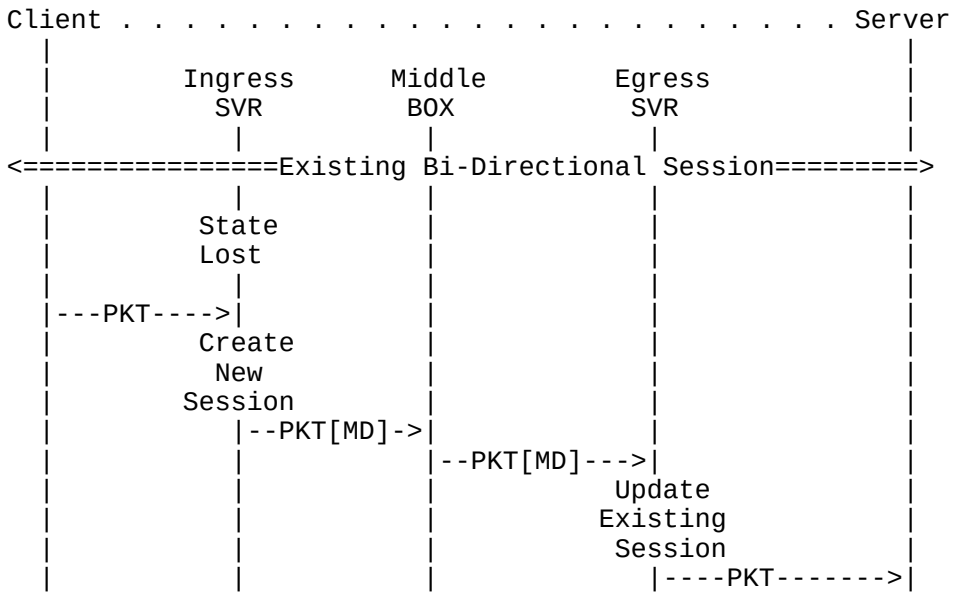


Figure 55

If the ingress loses state while the client is idle (server-side data cannot be delivered), the egress checks the client inactivity timer (default 5 s) on the next server packet, and if exceeded inserts Session Health Check metadata (Section 10.3.8). The ingress responds by generating a UDP packet matching the session's L3/L4 with an SVR Control Message (Section 10.3.6) carrying drop reason 6 ("delete session"). The egress then clears state, and the next server packet carries first-packet SVR Metadata, treated as a new session for state restoration.

State Recovering Ingress Router Client Inactive

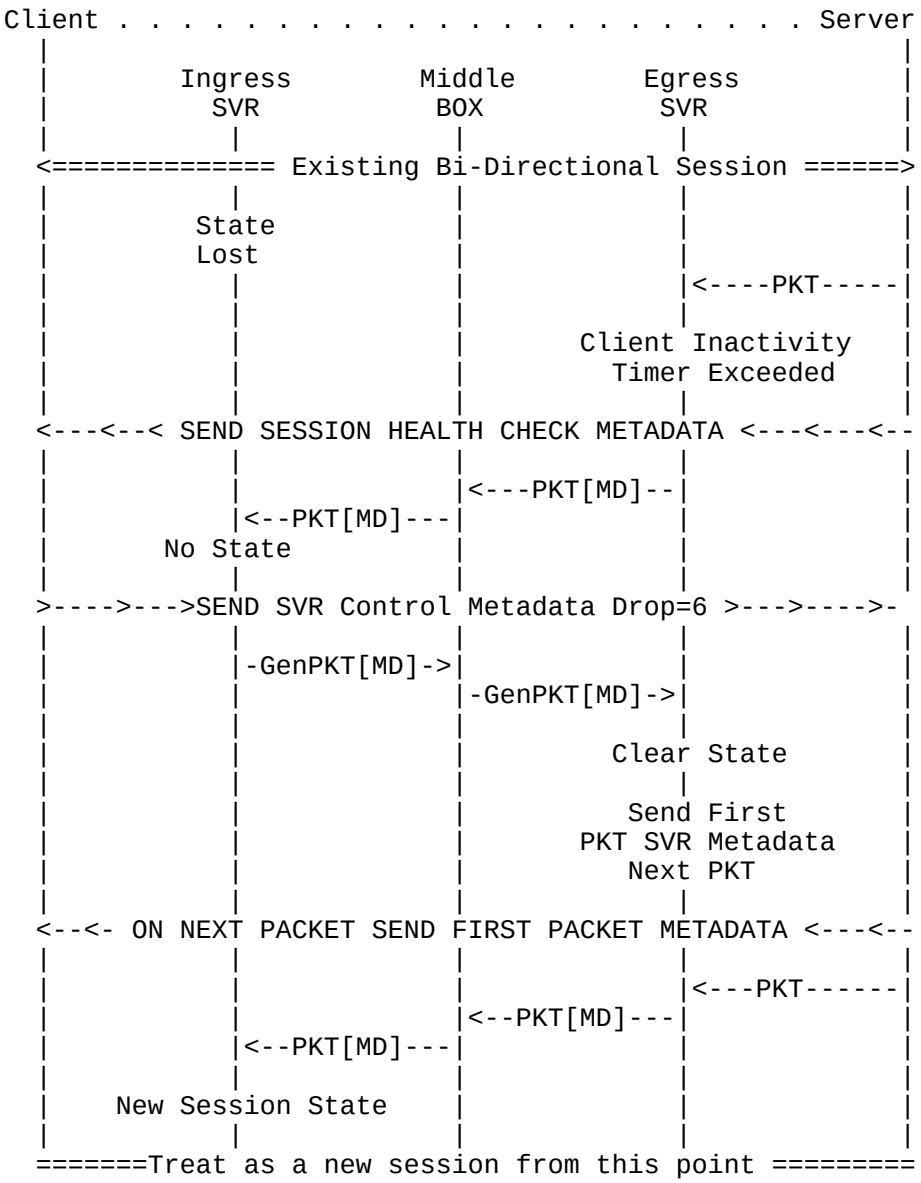


Figure 56

If the egress loses state, it pulls state from a peer. The ingress recognizes the recovery condition when it receives a packet for a session the server has not responded to within the inactivity timer;

it augments the next packet with Session Health Check metadata (Section 10.3.8). The egress MUST respond with a UDP packet carrying an SVR Control Message (Section 10.3.6) using drop reason 2 ("send metadata"). The client’s next packet will carry full first-packet SVR Metadata, restoring egress state.

State Recovering Egress Router

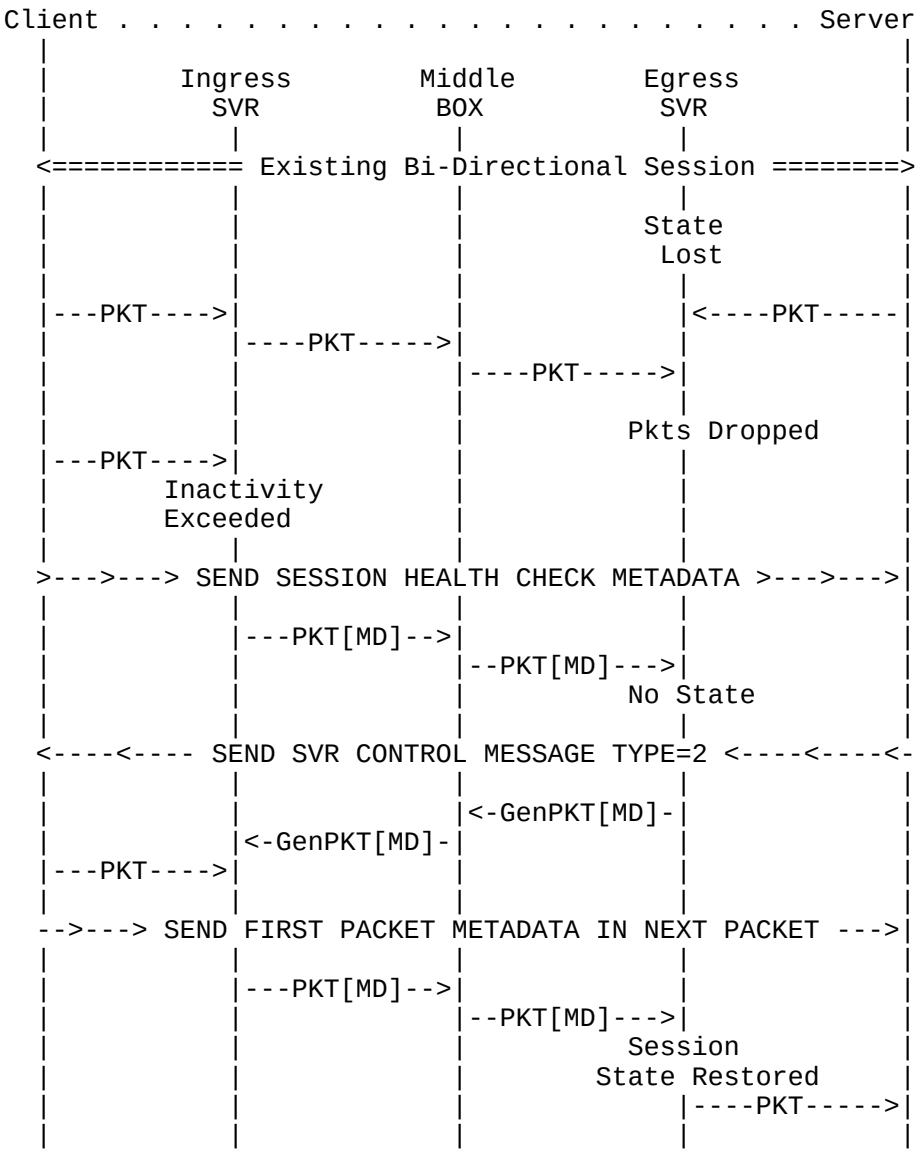


Figure 57

Middleboxes are the most common cause of state loss; they may stop forwarding in one or both directions, or silently rewrite ports.

Because the location of the loss is unknown, the router includes Session Health Check metadata in a packet of the session. SVR peers MUST respond; absence of a response indicates a middlebox or network problem.

Recovery is performed by clearing session state and treating the next packet as a first packet (full SVR Metadata exchange per Section 4.7.1). Both ingress and egress detect the matching 5-tuple and replace the existing session state.

State Recovering of Middlebox

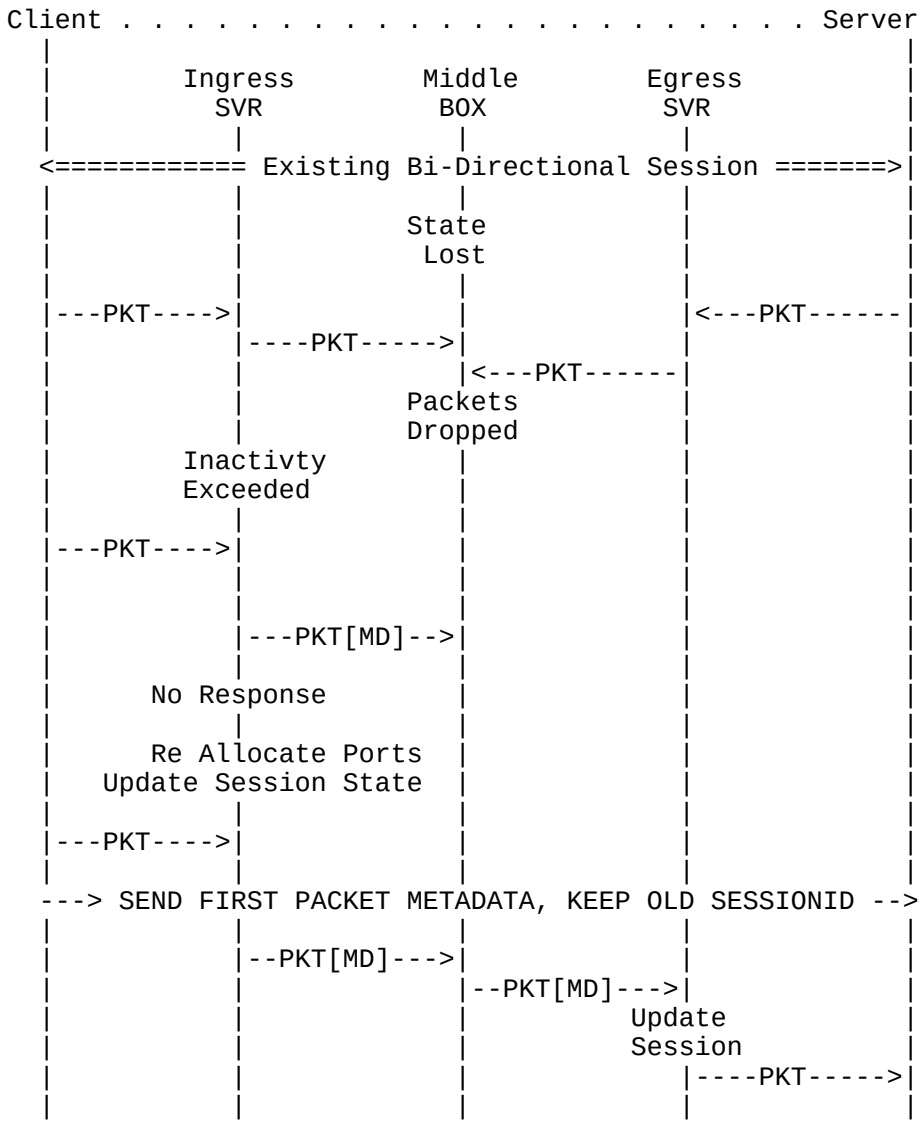

Client
Server
Ingress SVR
Middle BOX
Egress SVR
<========== Existing Bi-Directional Session =======>
State Lost
---PKT---->
----PKT----->
<---PKT------
Packets Dropped
Inactivty Exceeded
---PKT---->
---PKT[MD]-->
No Response
Re Allocate Ports
Update Session State
---PKT---->
---> SEND FIRST PACKET METADATA, KEEP OLD SESSIONID -->
--PKT[MD]--->
--PKT[MD]--->
Update Session
----PKT---->

Figure 58

8. SVR Multicast Support

This section describes how SVR transports IPv4 and IPv6 multicast traffic across an SVR overlay. SVR's session model, designed for unicast TCP/UDP, is extended to multicast by treating each (S,G) flow as a long-lived, unidirectional SVR session whose set of egress peers is derived from receiver-side group membership.

8.1. Architectural Model

SVR multicast is a hybrid distribution model:

* Where two or more SVR Peer Pathways share an underlay that supports native IP multicast (a common LAN segment, an IP/MPLS core with PIM, or an overlay that already provides multicast), the ingress SVR router MAY transmit a single replicated copy of the packet on a shared multicast Waypoint and rely on the underlay to fan out to egress SVR routers. This is the "native multicast" mode.

* Where the underlay does not support multicast end-to-end (the public IPv4 or IPv6 Internet, most SD-WAN transports, NATed paths), the ingress SVR router MUST fall back to ingress replication: a separate unicast SVR session is established to each egress SVR peer that has at least one interested receiver.

A single (S,G) flow MAY use a mix of both modes simultaneously (e.g., native multicast on an MPLS pathway and ingress replication over the Internet) without coordination beyond what is described below.

Multicast control-plane interaction with hosts uses standard IGMPv2 ([RFC2236]), IGMPv3 ([RFC3376]), and MLDv2 ([RFC3810]) at the SVR edge only. SVR does NOT run PIM, MSDP, or any multicast routing protocol over Peer Pathways; receiver state is distributed inside the SVR control channel as described in Section 8.3.

8.2. Multicast Terminology

Multicast Session: A unidirectional SVR session keyed by (Source IP, Group IP, Protocol) for SSM, or (*, Group IP, Protocol) for ASM. Identified by a Session UUID like any other SVR session.

First-Hop SVR Router (FHR): The SVR router on whose interface the multicast source is reachable. The FHR originates the multicast SVR session and inserts SVR Metadata into the first packet seen for an (S,G).

Last-Hop SVR Router (LHR): An SVR router that has one or more

downstream receivers (learned via IGMP/MLD) for an (S,G) and that emits native multicast onto the receiver-facing interface.

Egress Set: The set of LHRs (identified by Router Name) currently interested in an (S,G). Maintained by the FHR.

8.3. Group Membership Distribution

When an LHR receives an IGMP/MLD Report or Done/Leave message that changes its receiver state for an (S,G), it MUST signal the change to every potential FHR using an SVR Control Message (Section 10.3.6) carrying a Multicast Group Context TLV (Section 10.4.16).

The set of potential FHRs for a group is determined by routing policy (typically, peers that advertise reachability to the source prefix in ASM, or to the explicit source in SSM). Distribution of this policy is outside the scope of this document.

An FHR maintains the Egress Set per (S,G) as the union of all LHRs that have signaled non-empty receiver interest and have not subsequently signaled withdrawal. Egress Set entries SHOULD be soft state, refreshed at an administratively configured interval (default 60 seconds, RECOMMENDED upper bound 240 seconds, to align with IGMPv3/MLDv2 query intervals).

8.4. First-Packet Processing for Multicast

When the FHR receives the first packet of an (S,G) flow whose Egress Set is non-empty, it MUST construct forward SVR Metadata identical in structure to a unicast first packet (see Section 4.7.1) with the following differences:

* The Forward Context (Section 10.4.1 or Section 10.4.2) carries the original (S,G) addresses and the original L4 ports. The Destination Address in the Forward Context is the multicast group address.

* A Multicast Egress List TLV (Section 10.4.17) MUST be included, enumerating the Router Names of every LHR in the Egress Set.

* No Reverse Context is created; multicast SVR sessions are unidirectional. The SVR Metadata handshake described in Section 3.4 is replaced by the keep-alive procedure in Section 8.6.

* Port allocation (Section 5.2.3) is performed once per (S,G), not per egress peer; the same allocated ports are reused across all replicated copies so that a single (Group, port) tuple identifies the multicast SVR session on the wire.

For each entry in the Egress Set, the FHR then chooses a Peer Pathway to that LHR using the same selection logic as for unicast (Section 4.7.1.4). If two or more chosen pathways share a common multicast-capable Waypoint, the FHR MAY transmit a single copy onto that Waypoint with a multicast destination address; this is the native-multicast optimization. Otherwise, the FHR transmits one unicast copy per chosen pathway (ingress replication).

8.5. Egress (LHR) Processing

An LHR that receives a multicast SVR packet:

1. Validates and decrypts SVR Metadata exactly as for unicast (Section 5.6).

2. Restores the original (S,G) addresses and L4 ports from the Forward Context.

3. Uses local IGMP/MLD state to determine which receiver-facing interfaces have interested receivers for that (S,G), and emits a native multicast packet on each such interface. The TTL/Hop Limit MUST be decremented per [RFC1112] / [RFC4604].

4. If the LHR has no remaining receivers when the packet arrives, it MUST drop the packet and SHOULD send an SVR Control Message with reason code "NO_MULTICAST_RECEIVERS" back to the FHR. On receipt, the FHR removes the LHR from the Egress Set.

8.6. Keep-Alive and Session Lifetime

Because multicast SVR sessions are unidirectional, the standard bidirectional handshake cannot be used to confirm receipt of SVR Metadata. Instead:

* The FHR includes SVR Metadata in the first packet of every Egress Set refresh interval (default 60 seconds), and on every change to the Egress Set.

* LHRs MUST treat absence of multicast SVR data for the source-active timer (default 210 seconds, configurable, aligned with IGMPv3/MLDv2) as session termination and withdraw their Egress Set entry.

* If an LHR has not received a refresh after the source-active timer but still has interested receivers, it MUST re-signal its Group Context to potential FHRs as described in Section 8.3.

8.7. Multicast Security Considerations

All multicast SVR packets MUST be authenticated with the same Time-Based HMAC mechanism as unicast SVR (Section 5.5.1). The Session HMAC Key is established from the Peer Key of the FHR-to-LHR Peer Pathway being used; when a single replicated copy traverses a native-multicast underlay to multiple LHRs, every LHR in the Egress Set MUST share a common Metadata Key derived as described in Section 6.1.9, and that key MUST NOT be reused outside the Egress Set.

Payload encryption (Section 7.4) for multicast uses a single Payload Key generated by the FHR and distributed to every LHR in the Egress Set inside the encrypted portion of the unicast SVR Metadata sent at session start (or at rekey). When native-multicast replication is in use, the Payload Key is additionally distributed to all LHRs out-of-band over their respective unicast Peer Pathways before it is first applied to the shared multicast packet stream.

LHR removal from the Egress Set MUST trigger a Payload Key rekey if payload encryption is in use; otherwise a departing receiver could decrypt traffic for which it is no longer authorized.

9. IPv6 Considerations

SVR is dual-stack by design. This section consolidates IPv6-specific behavior that applies in addition to, or in place of, the procedures described elsewhere for IPv4. Where an existing section is IP-version-agnostic it is referenced; where it is not, the IPv6 behavior is stated here.

9.1. Addressing and Waypoints

An SVR Waypoint MAY be an IPv6 Global Unicast Address (GUA, [RFC4291]) or a Unique Local Address (ULA, [RFC4193]). Link-local addresses (fe80::/10) MUST NOT be used as Waypoints because they are not routable beyond a single link and therefore cannot anchor an SVR Peer Pathway. IPv4-mapped IPv6 addresses (::ffff:0:0/96) MUST NOT be used as Waypoints; an IPv4 Waypoint is configured as an IPv4 address.

A single SVR router MAY simultaneously expose IPv4 and IPv6 Waypoints on the same physical interface; each is treated as an independent Peer Pathway. A given SVR Peer relationship MAY span IPv4-only, IPv6-only, and dual-stack pathways; the Peer Pathway selected for a session (Section 4.7.1.4) determines the IP version of the encapsulating transport, independently of the IP version of the original packet.

9.2. IPv4 / IPv6 Interworking

Because the original 5-tuple is preserved in the Forward Context TLV and reconstructed at the egress SVR router, an SVR session whose original packet is IPv4 MAY traverse an IPv6 Peer Pathway, and vice versa. Concretely:

* The FHR selects the Forward Context TLV based on the original packet's IP version: Section 10.4.1 for IPv4 sources, Section 10.4.2 for IPv6 sources.

* The transport IP header used between SVR routers is independent and is built from the chosen Peer Pathway's Waypoint addresses.

* At the egress SVR router, the original packet is reconstructed in its original address family.

This behavior provides a NAT64-like service ([RFC6146]) for SVR-managed sessions without requiring a NAT64 gateway in the data plane: an IPv4-only client behind an SVR router can reach an IPv6-only server behind another SVR router, provided routing policy at the egress router resolves the destination Service to an IPv6 address and applies destination NAT as in Section 3.8. Stateful per-flow translation as described in [RFC6146] is the RECOMMENDED model for SVR egress when interworking address families.

SVR routers MUST NOT silently translate between IPv4 and IPv6 for non-SVR (transit) traffic; address-family interworking is provided only for sessions in which the FHR has constructed SVR Metadata.

9.3. IPv6 Extension Headers

SVR Metadata is inserted directly after the L4 (TCP or UDP) header (Section 5.2.1). For IPv6 packets, "directly after the L4 header" means after any chain of IPv6 extension headers ([RFC8200]) that precede the L4 header.

When building the transport IPv6 header for an SVR-encapsulated packet, an SVR router:

* MUST NOT copy Hop-by-Hop or Destination extension headers from the original IPv6 packet onto the SVR transport header. The original extension header chain is preserved as part of the original payload but is not visible to the underlay.

* MUST NOT insert a Routing extension header on its own initiative. If SR-over-v6 ([RFC8986]) policy applies to the chosen Peer Pathway, the SR Header is constructed per [RFC8986] and is treated as part of the underlay transport.

* MUST handle a received Fragment header per Section 7.6; SVR Metadata MUST be present only in the first fragment.

9.4. Hop Limit, Traffic Class, and Flow Label

The original IPv6 header fields are preserved in SVR Metadata via the Forward and Reverse Context TLVs, and on egress are restored to their original values. On the SVR transport header, the following rules apply:

* *Hop Limit*: SVR routers act as IP routers and MUST decrement the original packet’s Hop Limit by 1 at the FHR (or by N at an N-hop SVR path, one per SVR router that the original packet logically traverses). The transport Hop Limit is independent and set by the originating SVR router according to local policy (default 64).

* *Traffic Class*: The original Traffic Class (including DSCP) MUST be copied to the transport IPv6 header by default, so that DSCP-based QoS in the underlay applies. An administrator MAY override this by policy.

* *Flow Label*: The transport Flow Label is set per session and MUST be derived from the SVR-allocated source/dest port pair so that ECMP hashing in the IPv6 underlay ([RFC6438]) treats each SVR session as a distinct flow, mirroring the unique-5-tuple property of SVR over IPv4 (Section 3.9). The original packet’s Flow Label is preserved in the Forward Context TLV.

9.5. ICMPv6 Interaction

ICMPv6 ([RFC4443]) replaces ICMP for IPv6 SVR sessions. The ICMP procedures in Section 7.7 apply with the following IPv6-specific details:

* A Packet Too Big (Type 2) message received on an SVR transport IPv6 packet MUST be honored by the receiving SVR router for Path MTU Discovery ([RFC8201]) on that Peer Pathway, in addition to being relayed back to the original sender as described in Section 7.6.

* Neighbor Discovery ([RFC4861]) traffic MUST NOT be encapsulated in SVR. NS, NA, RS, RA, and Redirect messages are link-local and are forwarded by the underlay only.

* MLDv1/MLDv2 ([RFC3810]) at the LHR is the IPv6 analog of IGMPv2/v3 in Section 8.3; the same distribution mechanism and timers apply.

9.6. BFD over IPv6

BFD ([RFC5880]) and BFD Metadata (Section 6) operate identically over IPv6 Peer Pathways, with the encapsulation defined by [RFC5881] for single-hop and [RFC5883] for multihop. The BFD Metadata payload is unchanged. When a Peer Pathway uses an IPv6 Waypoint, BFD packets MUST use a matching IPv6 source address; IPv4 BFD MUST NOT be used to monitor an IPv6 pathway.

10. SVR Metadata Format

SVR Metadata consists of Header attributes and Payload attributes. Header attributes are always unencrypted, are router/pathway scoped (no forward/reverse distinction), and MAY be inspected by intermediate elements but MUST NOT be modified. Payload attributes are session scoped (with forward/reverse semantics) and MAY be encrypted by the sender. Each TLV defined below is exclusively either a header or payload attribute.

10.1. SVR Metadata Header

The SVR Metadata header begins with a well-known 8-octet "cookie" (0x4c48dbc6ddf6670c, network byte order) immediately following the L4 header, used by receivers to detect SVR Metadata. Normal IP traffic never targets a Waypoint address; any packet arriving at a Waypoint MUST either carry SVR Metadata or belong to an active SVR session (see Section 3.11 for state recovery).

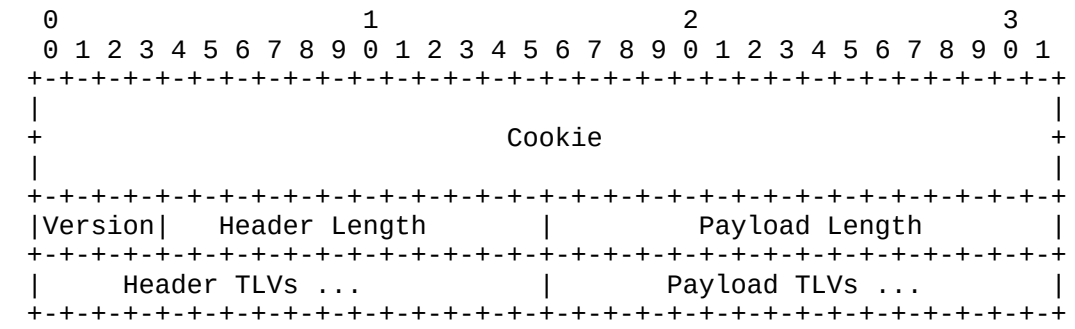


Figure 59

Cookie (8 octets): The fingerprint of SVR Metadata. This value is used to determine the existence of SVR Metadata within a packet.

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

Header Length (12-bits): Length of the SVR 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 octets): Length of data following the SVR Metadata header, not including the size of the header. This data may be encrypted. The value of this field is the number of octets in the Payload TLVs. If there are no TLVs the value is zero.

10.1.1. False Positives

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

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

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

10.1.2. Forward and Reverse Attributes

Payload SVR 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.

10.2. TLVs for Attributes

All SVR Metadata attributes are expressed as Tag Length Values or TLVs. 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 TLVs, they can be transmitted in clear text. The entire SVR 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 SVR Metadata. Doing so will invalidate the signature, and the SVR Metadata will be dropped.

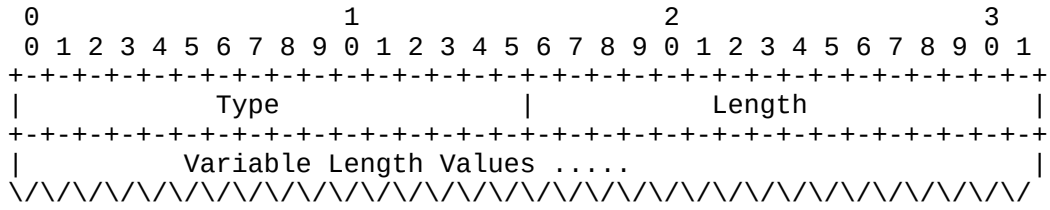


Figure 60

Type (2 octets): Type of data that follows. Each of different Header and Payload TLVs are defined below.

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

10.3. Header Attributes

10.3.1. Fragment

When a packet is fragmented to insert SVR 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 an SVR Metadata signaled pathway, it must also have this SVR Metadata attached to properly bind the fragment with the correct session.

All fragments will have an SVR Metadata header and the fragment TLV added to the guaranteed unencrypted portion of the SVR Metadata header. If the original packet already has an SVR 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 7.5

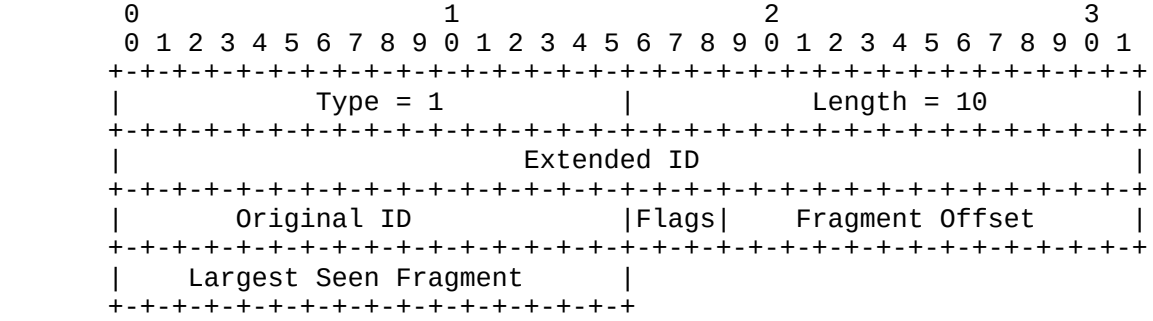


Figure 61

TLV: Type 1, Length 10.

Extended ID (4 octets): 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. The same algorithm for fragmenting packets is used for both IPv6 and IPv4, therefore a 32-Bit Extended ID is used.

Original ID (2 octets): 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

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-octet segments the fragment payload contains.

Largest Seen Fragment (2 octets): 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.

10.3.2. Security ID

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

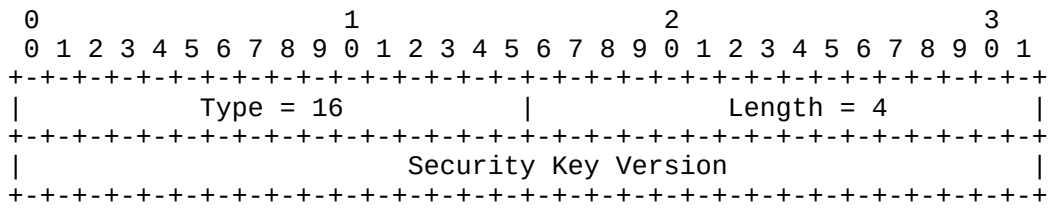


Figure 62

TLV: Type 16, Length 4.

Security Key Version (4 octets): This is a four-octet security key version identifier. This is used to identify the algorithmic version used for SVR Metadata authentication and encryption.

10.3.3. Disable Forward SVR Metadata

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

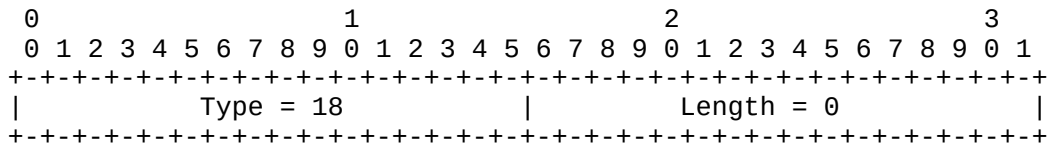


Figure 63

TLV: Type 18, Length 0.

No other data is required. The specific session referenced is looked up based on the 5-tuple address of the packet. See SVR Metadata handshake in Section 3.4.

10.3.4. IPv4 ICMP Error Location Address

This is exclusively used to implement ICMP messages that need to travel backwards through SVR pathways. See Section 7.7 for more information. The IPv4 address of the source of the ICMP message is placed into SVR Metadata. This SVR 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.

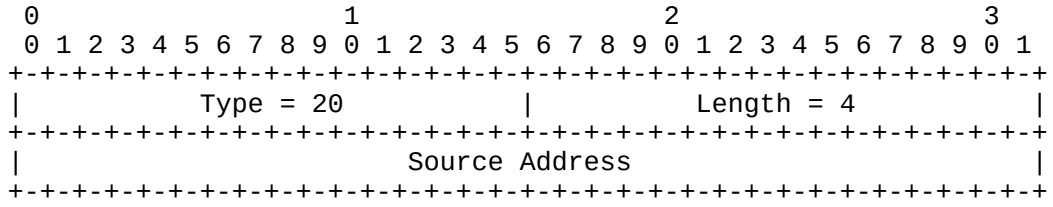


Figure 64

TLV: Type 20, Length 4.

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

10.3.5. IPv6 ICMP Error Location Address

This is exclusively used to implement ICMP messages that need to travel backwards through SVR pathways. See Section 7.7 for more information. The IPv6 address of the source of the ICMP message is placed into SVR Metadata. This SVR 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.

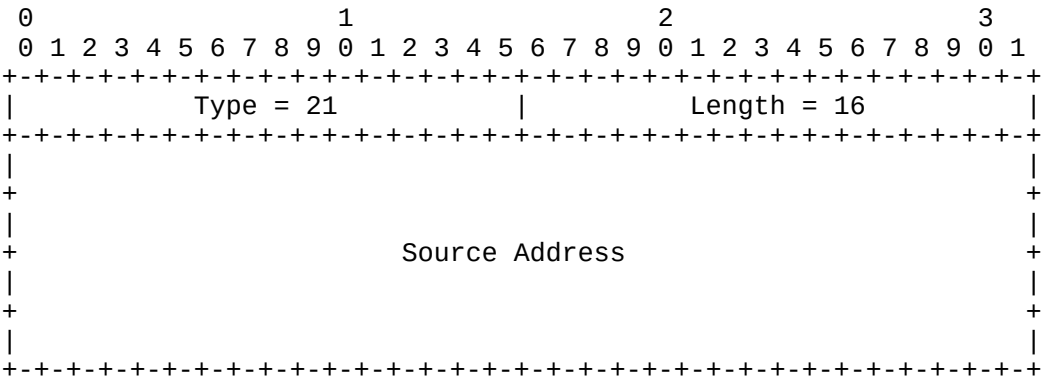


Figure 65

TLV: Type 21, Length 16.

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

10.3.6. 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 SVR 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" at a rate (e.g., one message every 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 SVR 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 3.11 for more information.

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

Delete Session: The session associated with the flow spec used by this generated packet should be deleted. This provides an administrative and error correcting capability to remove a session when required.

Session State Exists: In response to a Session Health Check request (see Section 10.3.8 to indicate that state for a session exists.

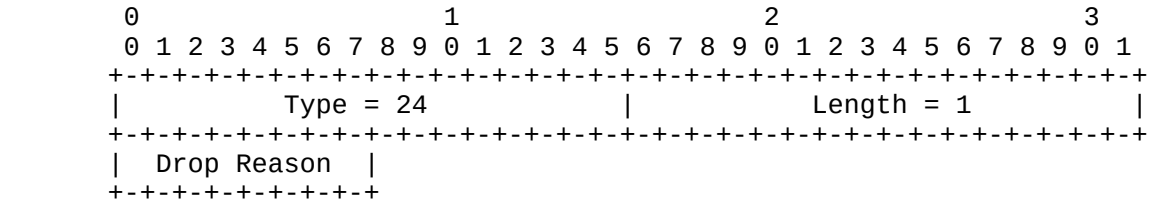


Figure 66

TLV: Type 24, Length 1.

Drop Reason (1 octet): 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 SVR Metadata. Begin sending SVR Metadata on the peer pathway for the 5-tuple matching this control packet.

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

* 6 = Delete Session. Delete any state for the session associated with this SVR Metadata.

* 8 = Session Health Check indicates state exists, and is valid.

10.3.7. Path Metrics

This SVR 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 SVR Metadata type.

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

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

After several exchanges, each side can compute accurate path metrics for the traffic included. This SVR Metadata can be sent at any time, but is normally sent when SVR Metadata is being sent for other reasons. The SVR 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 [RFC9341]

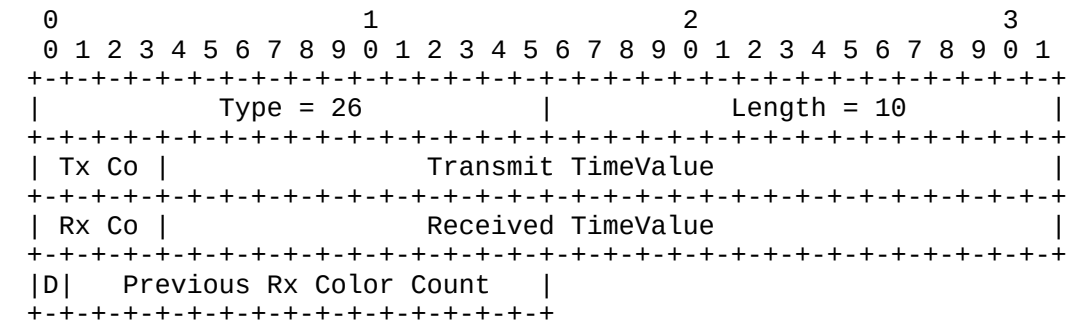


Figure 67

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 that 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.

10.3.8. Session Health Check

This SVR Metadata is used to request a session state check by a peer. The peer responds upon receipt with a generated packet with SVR Metadata confirming presence of SVR Metadata. This SVR Metadata type can be included in an existing packet to check that the peer has session state. The peer will always respond with a generated packet that includes a forced drop SVR Metadata attribute. See Section 7.8 for an example.

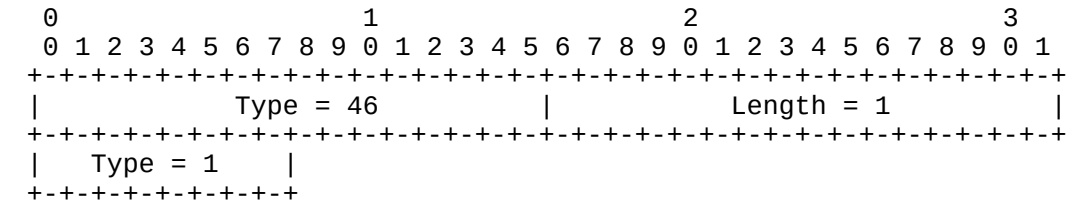


Figure 68

TLV: Type 46, Length 1.

TYPE: (1=Request, 2=Request/Timeout) Request to verify session state with backward SVR Metadata. Type 1 indicates session state is available, Type 2 indicates session state is available but will be cleared and replaced upon receipt of state from a peer. Type 2 is used when a middle box closes pinholes that must be recovered.

10.4. Payload Attributes

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

10.4.1. 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.

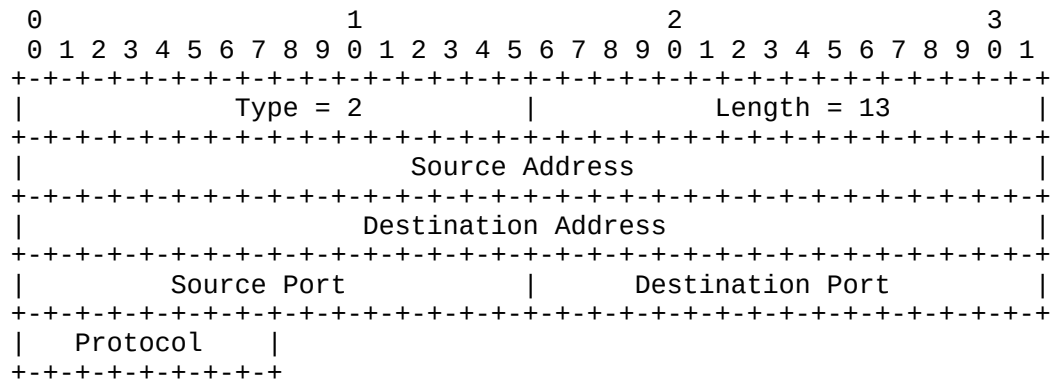


Figure 69

TLV: Type 2, Length 13.

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

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

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

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

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

10.4.2. Forward Context IPv6

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

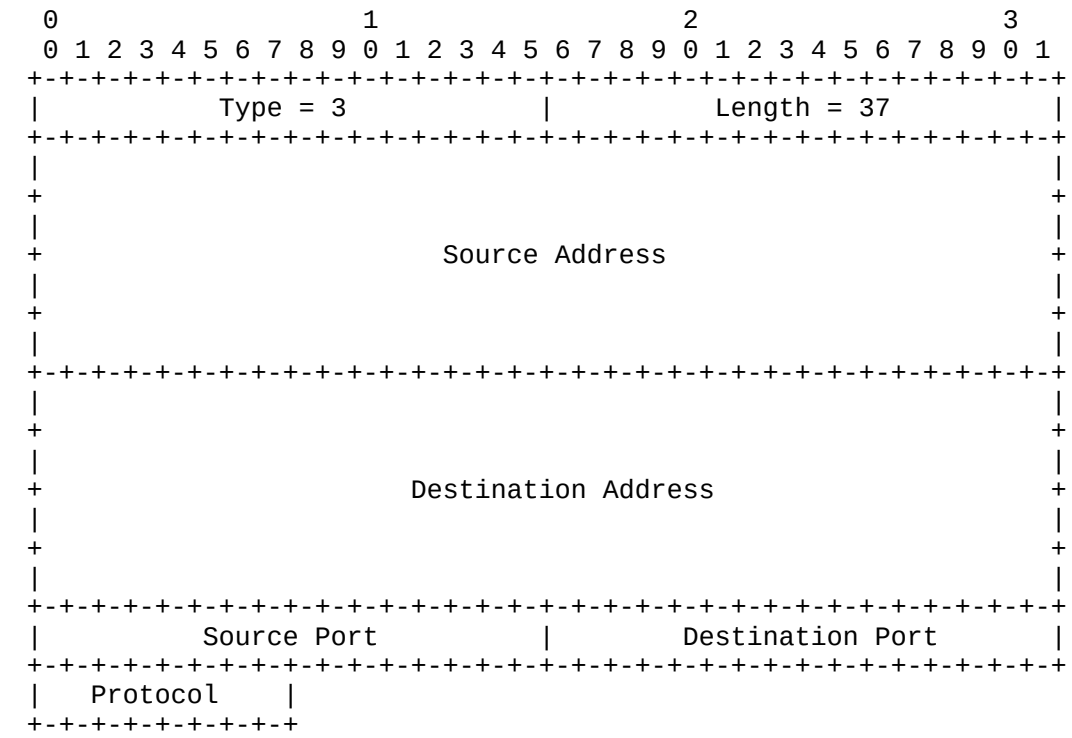


Figure 70

TLV: Type 3, Length 37.

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

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

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

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

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

10.4.3. 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.

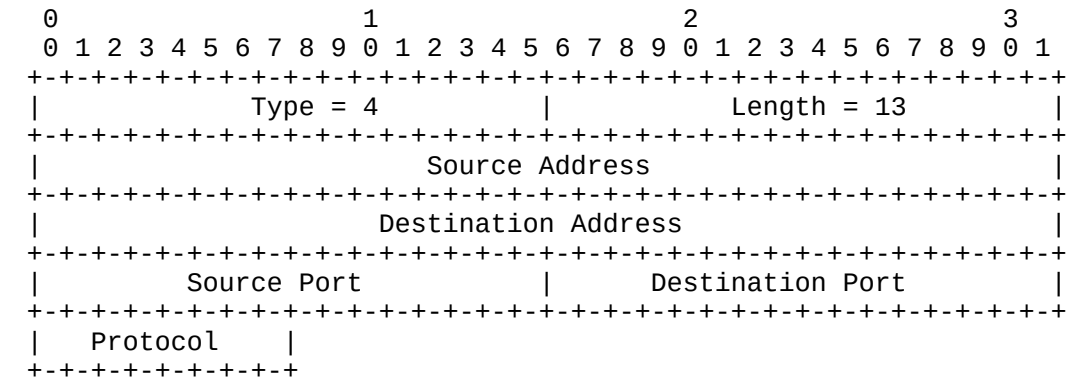


Figure 71

TLV: Type 4, Length 13.

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

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

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

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

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

10.4.4. 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.

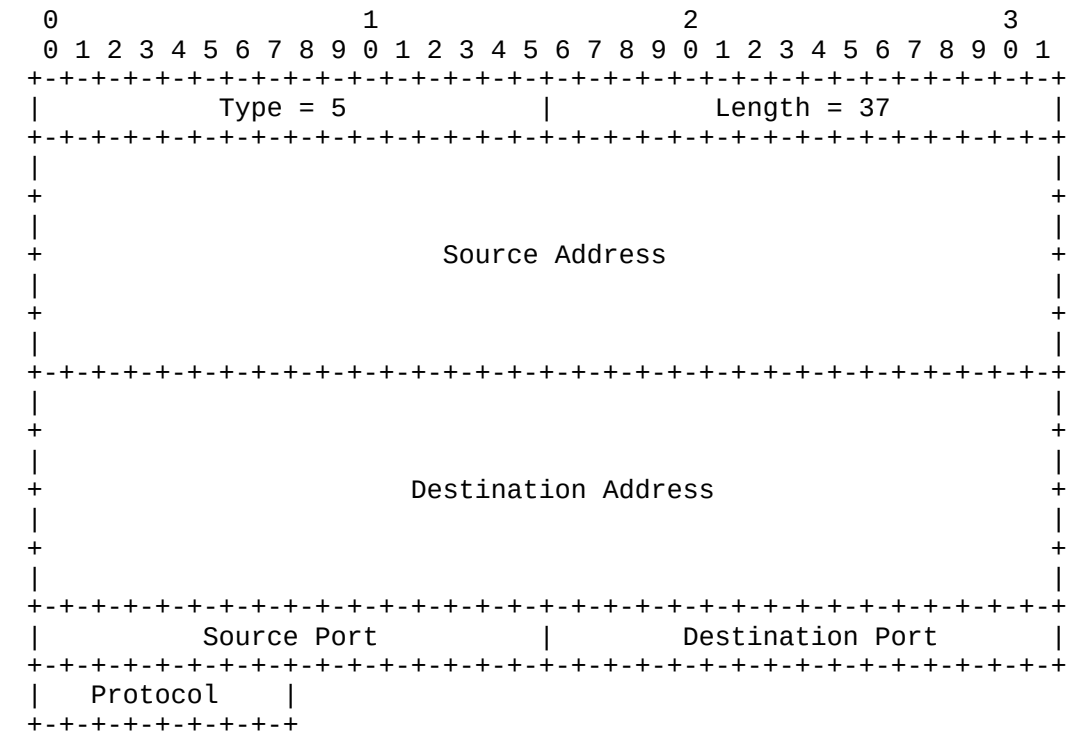


Figure 72

TLV: Type 5, Length 37.

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

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

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

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

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

10.4.5. Session UUID

Unique identifier of a session. The UUID MUST be conformant to [RFC9562]. 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 SVR Metadata field is required for all session establishment.

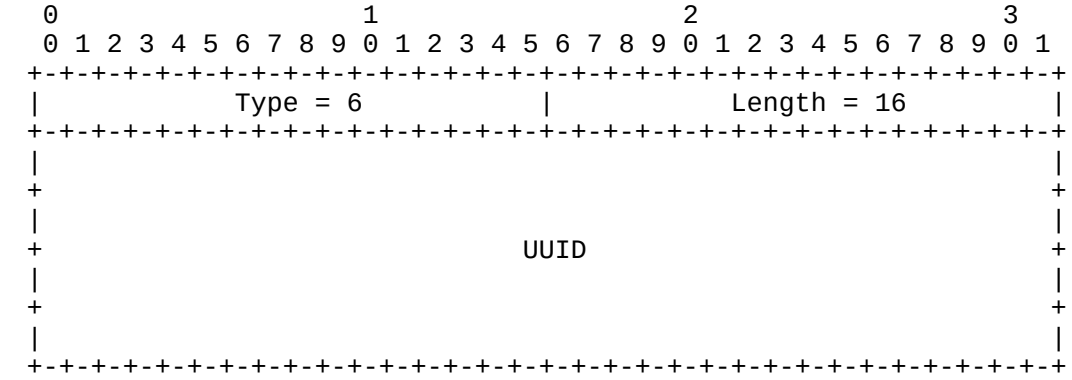


Figure 73

TLV: Type 6, Length 16.

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

10.4.6. Tenant Name

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

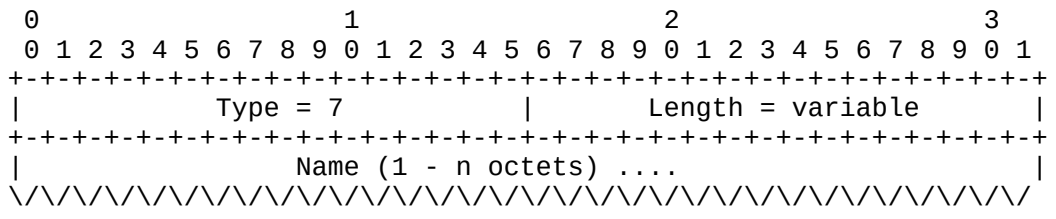


Figure 74

TLV: Type 7, Length variable.

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

10.4.7. Service Name

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

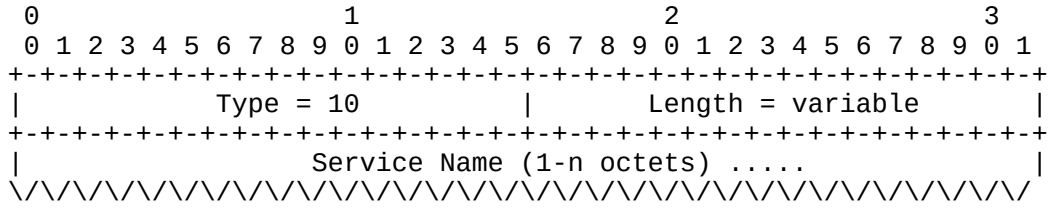


Figure 75

TLV: Type 10, Length variable.

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

10.4.8. Session Encrypted

Indicates if the session is having its payload encrypted by the SVR router. This is different from having the SVR Metadata encrypted. The keys used for payload encryption are dependent on the Security Policy defined for a session.

This field is necessary because often traffic is already encrypted before arriving at an SVR router (making DPI a poor choice). Also in certain use cases, re-encryption may be required. This SVR Metadata TLV is always added when SVR encrypts the payload.

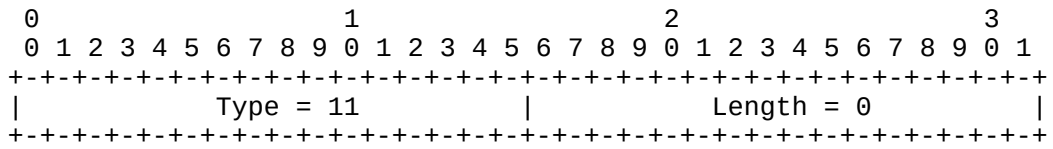


Figure 76

TLV: Type 11, Length 0.

10.4.9. 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 SVR 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 octets are changed other than the protocol value (TCP->UDP). Because the UDP message length and checksum overlap with the TCP Sequence Number, the Sequence number is overwritten. The Sequence number is saved by copying it to the TCP Checksum and Urgent Pointer portion of the packet. The Checksum is recalculated upon restoration of the packet. The urgent pointer is zeroed out and urgent flag is cleared.

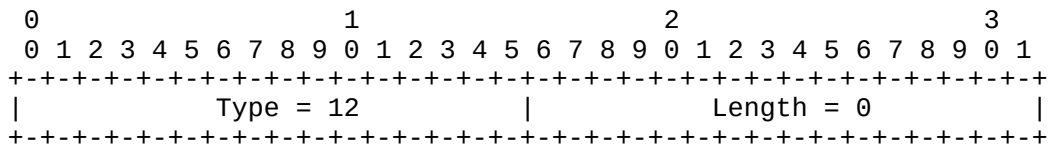


Figure 77

TLV: Type 12, Length 0.

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

10.4.10. Source Router Name

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

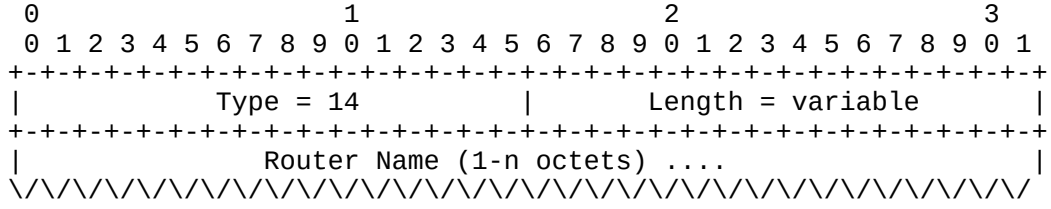


Figure 78

TLV: Type 14, Length variable.

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

10.4.11. Security Policy

An alphanumeric string containing the Security Policy to use for a particular session. This is used only when payload encryption is being performed. The Security Policy describes the specifics about ciphers used for payload encryption.

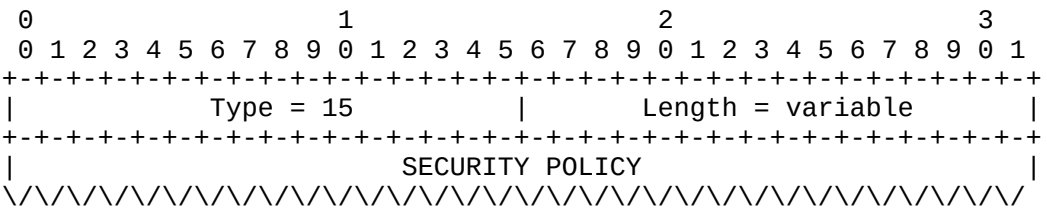


Figure 79

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.

10.4.12. 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.

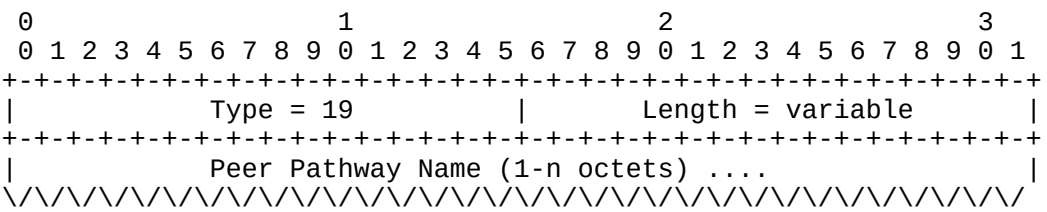


Figure 80

TLV: Type 19, Length variable.

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

10.4.13. 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 SVR 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.

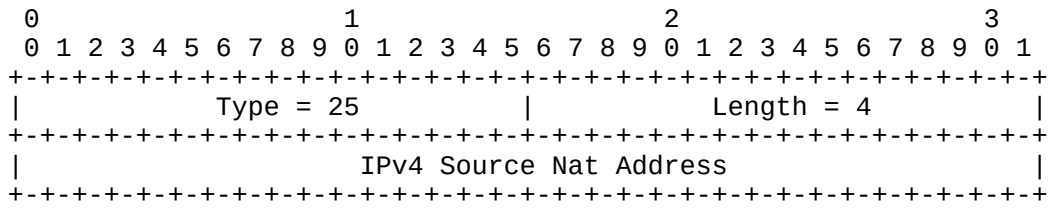


Figure 81

TLV: Type 25, Length 4.

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

10.4.14. Remaining Session Time

After a path failure, it may become necessary to transmit an SVR Control Message when there are one-way flows waiting for a packet to be transmitted. In these cases, the SVR Metadata includes an attribute defining the remaining session time so intermediate routers creating new session entries will expire the session at the correct time.

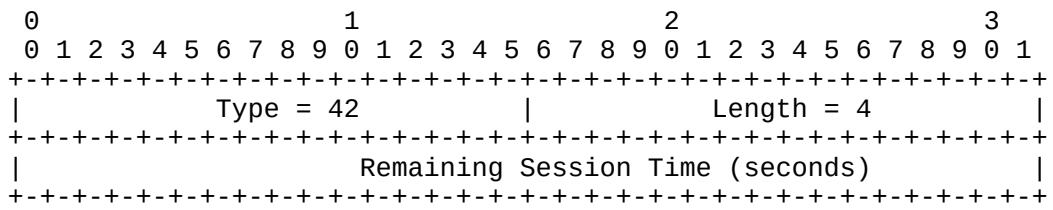


Figure 82

TLV: Type 42, Length 4.

Remaining Session Time (4 octets): Number of seconds remaining on a session packet guard time. This ensures accurate guarding of sessions that have been moved.

10.4.15. Security Encryption Key

An alphanumeric string containing the cryptographic key to use for a payload encryption of a particular session. This is used only when payload encryption is being performed. The key is encrypted in SVR Metadata hop-by-hop through a network, preventing any party from obtaining the key. The router terminating the session utilizes this key to decrypt payload portions of packets. This prevents re-encryption penalties associated with multi-hop routing scenarios.

To rekey a session, this SVR Metadata can be included in any subsequent packet with the new key to use. When rekeying, the SVR that initiated the encrypted session must compute a new key, and include the key as SVR Metadata.

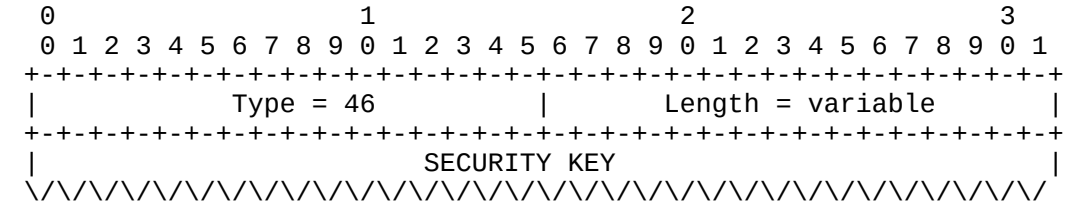


Figure 83

TLV: Type 46, Length variable.

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

10.4.16. Multicast Group Context

Identifies the (Source, Group, Protocol) of a multicast SVR session. Used both in SVR Control Messages signaling group membership (Section 8.3) and as a payload TLV in the first packet of a multicast session. The address family field selects between IPv4 (lengths shown) and IPv6 (16-octet addresses).

TLV: Type 50, Length variable.

Address Family (1 octet): 0x04 = IPv4, 0x06 = IPv6.

Flags (1 octet): Bit 0 = SSM (1) or ASM (0). Bit 1 = Join (1) / Leave (0) when used in a Control Message; ignored when used as session metadata. Remaining bits reserved, MUST be sent as 0 and ignored on receipt.

Source Address (4 or 16 octets): The multicast source address. For ASM, the all-zero address of the appropriate family.

Group Address (4 or 16 octets): The multicast group address.

Protocol (1 octet): L4 protocol of the multicast flow (typically 17, UDP).

10.4.17. Multicast Egress List

Enumerates the LHRs currently in the Egress Set of a multicast SVR session. Carried in forward SVR Metadata sent by the FHR.

TLV: Type 51, Length variable.

Count (2 octets): Number of router-name entries that follow.

Entries (variable): A sequence of Count entries, each consisting of a 1-octet length followed by that many octets of UTF-8 router name. Total TLV length MUST match the Length field.

11. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist. According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this

information as they see fit".

11.1. Session Smart Router

* Organization: Hewlett Packard Enterprise
* Name: Session Smart Router
* Description: The Session Smart Router exists as a SDWAN router following the implementation defined in this document.
* Level of maturity: Production
* Coverage: All aspects of the protocol are implemented.
* Licensing: Proprietary
* Contact: michael.baj@hpe.com
* URL: https://www.juniper.net/documentation/us/en/software/session-smart-router/

12. Security Considerations

12.1. HMAC Authentication

Packets carrying SVR Metadata are HMAC-signed to authenticate the sender and protect integrity. Any HMAC algorithm agreed by both peers MAY be used (e.g., SHA1, SHA256, SHA256-128). The signature covers the L4 payload and is appended to the end of the packet.

12.2. Replay Prevention

Time-Based HMAC on every packet is RECOMMENDED. It prevents in-stream tampering, supports egress state-loss detection (Section 3.11), and binds each signature to a time window to prevent replay. Provisioning of HMAC parameters is out of scope.

12.3. Payload Encryption

Payload encryption uses AES-CBC-128 or AES-CBC-256. Plaintext is padded to a 16-octet boundary; if the plaintext is already a multiple of 16, an additional 16-octet pad block is appended whose last octet indicates the pad length.

12.4. DDoS and Unexpected Traffic on Waypoint Addresses

Any host can send packets to a Waypoint address. SVR routers MUST treat such packets as either bearing SVR Metadata or belonging to an established session/protocol; everything else is dropped.

Source-address ACLs SHOULD restrict accepted SVR traffic to provisioned peers. The 8-octet SVR cookie is checked first; on mismatch the packet is dropped. The HMAC is checked next; on failure the packet is dropped. Together these checks deflect DDoS attempts targeting Waypoint addresses.

13. IANA Considerations

This document is published as an Independent Submission and does not require IANA to create new top-level registries. The SVR Metadata Type code points used in this document are administered by the Authority that operates an SVR deployment (see Section 1.4) and are summarized below for implementer convenience. Type values are scoped separately for Header and Payload TLVs.

Header TLV Types currently defined: 1 (Fragment), 16 (Security ID), 18 (Disable Forward SVR Metadata), 20 (IPv4 ICMP Error Location), 21 (IPv6 ICMP Error Location), 24 (SVR Control Message), 26 (Path Metrics), 46 (Session Health Check).

Payload TLV Types currently defined: 2 (Forward Context IPv4), 3 (Forward Context IPv6), 4 (Reverse Context IPv4), 5 (Reverse Context IPv6), 6 (Session UUID), 7 (Tenant Name), 10 (Service Name), 11 (Session Encrypted), 12 (TCP SYN Packet), 14 (IPv4 Source NAT Address), 15 (Source Router Name), 19 (Peer Pathway ID), 25 (Remaining Session Time), 42 (Security Policy), 46 (Security Key), 50 (Multicast Group Context), 51 (Multicast Egress List).

Unassigned values in either space are available for future use. Implementations MUST ignore unknown TLVs as required by Section 5.6.2.1.

14. Acknowledgements

The authors would like to thank Anya Yungelson, Scott McCulley, and Chao Zhao 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 Hewlett Packard Enterprise (formerly Juniper Networks) for their support and guidance.

15. Normative References

[ECDH_Key_Exchange]
Nakov, S., "Practical Cryptography for Developers", ISBN 978-619-00-0870-5, Publisher Sofia, November 2018, <https://cryptobook.nakov.com/asymmetric-key-ciphers/ecdh-key-exchange>.

[NIST_SP_800-56A]
Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography", ISBN NIST Special Publication 800-56A Rev3, Publisher National Security Agency, April 2018, <https://csrc.nist.gov/pubs/sp/800/56/a/r3/final>.

[NIST_SP_800-90B]
Turan, M., Barker, E., Kelsey, J., McKay, K., Baish, M., and M. Boyle, "Recommendation for the Entropy Sources Used for Random Bit Generation", ISBN NIST Special Publication 800-90B, Publisher National Institute of Standards and Technology, January 2018, <https://csrc.nist.gov/pubs/sp/800/90/b/final>.

[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 Control Message 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>.

[RFC9562] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 9562, DOI 10.17487/RFC9562, July 2005, <https://www.rfc-editor.org/info/rfc9562>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC5758] Dang, Q., Santesson, S., Moriarty, K., Brown, D., and T. Polk, "Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA", RFC 5758, DOI 10.17487/RFC5758, January 2010, <https://www.rfc-editor.org/info/rfc5758>.

[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>.

[RFC9300] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 9300, DOI 10.17487/RFC9300, January 2013, <https://www.rfc-editor.org/info/rfc9300>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/info/rfc7468>.

[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>.

[RFC9341] 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 9341, DOI 10.17487/RFC9341, January 2018, <https://www.rfc-editor.org/info/rfc9341>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

16. Informative References

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997, <https://www.rfc-editor.org/info/rfc2236>.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002, <https://www.rfc-editor.org/info/rfc3376>.

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005, <https://www.rfc-editor.org/info/rfc4193>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using IGMPv3 and MLDv2 for Source-Specific Multicast", RFC 4604, August 2006, <https://www.rfc-editor.org/info/rfc4604>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, June 2010, <https://www.rfc-editor.org/info/rfc5881>.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, June 2010, <https://www.rfc-editor.org/info/rfc5883>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

Appendix A. Changes from -07 to -08

* Added Section 8 describing SVR support for IPv4 and IPv6 multicast, including a hybrid native-multicast / ingress-replication distribution model, edge-only IGMP/MLD interaction, and per-(S,G) session lifecycle.

* Added Section 9 consolidating IPv6-specific behavior: addressing and Waypoints, IPv4/IPv6 interworking, IPv6 extension header handling, Hop Limit / Traffic Class / Flow Label rules, ICMPv6 interaction, and BFD over IPv6.

* Added two new payload TLVs: Multicast Group Context (Type 50, Section 10.4.16) and Multicast Egress List (Type 51, Section 10.4.17).

* Updated Section 13 to enumerate the currently-defined Header and Payload TLV Type code points, including the new multicast TLVs.

* Tightened the Abstract.

Authors' Addresses

Abilash Menon
Maia Edge
77 S Bedford St Suite 150
Burlington, MA 01803
United States of America
Email: abilashmenon@maiaedge.io

Patrick MeLampy
Retired
1024 Main St.
Dustable, MA 01827
United States of America
Email: pmelampy@gmail.com

Michael Baj
Hewlett Packard Enterprise
10 Technology Park Dr.
Westford, MA 01886
United States of America
Email: michael.baj@hpe.com

Patrick Timmons
Maia Edge
77 S Bedford St Suite 150
Burlington, MA 01803
United States of America
Email: ptimmons@maiaedge.io

Hadriel Kaplan
Hewlett Packard Enterprise
10 Technology Park Dr.
Westford, MA 01886
United States of America
Email: hadriel.kaplan@hpe.com