Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 June 2026

Z. Luo, Ed.
H. Yan
CMCC

25 December 2025

Available Session Recovery Protocol
draft-cmcc-asrp-01

Abstract

This document describes an experimental protocol named the Available Session Recovery Protocol (ASRP). The protocol aims to optimize high-availability network cluster architectures, providing a superior cluster high-availability solution for stateful network services such as load balancing and Network Address Translation (NAT). ASRP defines the procedures for session backup and recovery, along with the message formats used in interactions, enabling efficient and streamlined session state management.

Unlike traditional high-availability technologies that back up session states within the cluster, the core innovation of ASRP lies in distributing state information to clients or servers. This approach offers multiple advantages, significantly enhancing the cluster’s elastic scaling capability; supporting rapid recovery from single-point or even multi-point failures; reducing resource redundancy by eliminating centralized backup nodes; and substantially simplifying cluster implementation complexity.

The ASRP protocol provides network clusters with ultimate elastic scalability, facilitating the implementation and deployment of large-scale elastic network clusters.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 June 2026.

Table of Contents

1. Introduction

Traditional high-availability network cluster architectures, such as active-standby or hierarchical models, rely on session state synchronization and backup within the cluster nodes. While these architectures are functionally complete, they face challenges in the cloud era, including limited flexibility in elastic scaling, resource redundancy, and high implementation complexity.

The Available Session Recovery Protocol (ASRP) proposes an innovative approach aimed at building a simpler, more efficient, and more elastic high-availability solution for stateful services. Its core idea fundamentally shifts the paradigm by distributing the backup of network session state information to the endpoints of the session-clients or servers-rather than within the cluster. This allows network service nodes in the cluster (such as load balancers or NAT devices) to quickly retrieve and reconstruct session states from endpoints during failure recovery or scaling, thereby logically achieving "stateless" nodes.

Based on this concept, ASRP designs corresponding session backup and recovery mechanisms. The backed-up session state is strictly synchronized with the lifecycle of the actual network session-it becomes effective upon session establishment and is cleared upon session termination. This eliminates the need for independent keep-alive or timeout management, ensuring the timeliness and availability of the backup information.

Another key design goal of ASRP is to improve the efficiency of session backup. The protocol ingeniously utilizes in-band original data packets carrying service traffic to transmit session state information, embedding it into the payload of packets such as those in the TCP three-way handshake ([RFC9293] [RFC1122]). This approach avoids the overhead of generating additional control packets for state synchronization in most cases. While its implementation shares similarities with TCP Fast Open (TFO, [RFC7413]), it remains fully transparent to the application layer.

1.1. Elastic Stateful Clusters

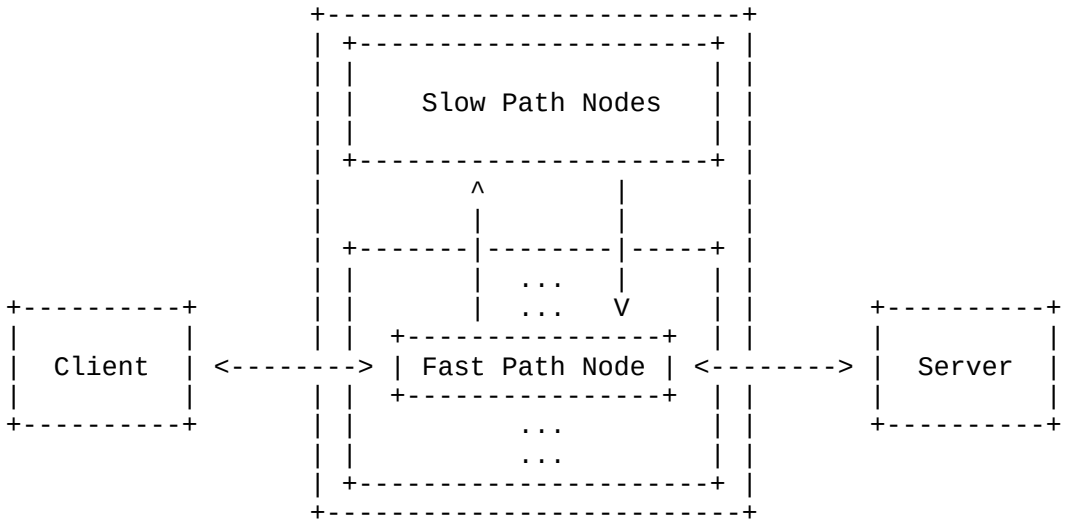

Figure 1: Fast/Slow Path Design for Elastic Stateful Clusters

Elastic Stateful Cluster is a highly available network service cluster composed of multiple coordinated nodes that collectively provide stateful network services such as load balancing (SLB) and network address translation (NAT) to external users. To achieve elastic scaling capabilities, traditional Elastic Stateful Clusters typically adopt a fast-path/slow-path architecture, separating session management from packet forwarding. This allows the fast-path layer to scale elastically with high efficiency. A schematic representation is shown below:

The slow-path nodes are responsible for session creation and session synchronization, while the fast-path nodes handle high-speed packet forwarding. When a fast-path node fails, external traffic is automatically redirected to other healthy nodes, ensuring continuous service availability. The main drawback of traditional Elastic

Stateful Clusters is the limited elastic scalability of the slow-path layer. The slow-path nodes must implement complex session-synchronization mechanisms internally. A typical implementation can be found in the AWS Hyperplane NFV platform.

The ASRP protocol focuses on session-state backup and recovery, ensuring session consistency and continuity for stateful services within cluster nodes. An Elastic Stateful Cluster built on ASRP features atomic internal nodes - nodes do not need to communicate with each other, and no session synchronization is required within the cluster. This design significantly enhances the cluster's elastic scaling capability, supports fast recovery from single-point or even multi-point failures, and reduces resource overhead and implementation complexity by eliminating centralized backup nodes. These characteristics make ASRP particularly suitable for network environments that require frequent elastic scaling, pursue high resource efficiency, and demand high service stability. ASRP thereby provides a powerful solution for the implementation and deployment of large-scale network clusters.

2. 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.

3. Protocol Overview

3.1. Core Concepts

3.1.1. Two Operational Modes

For the ASRP protocol to function properly, all network nodes within the cluster must first deploy service programs that support the ASRP protocol. Additionally, the clients or servers responsible for backing up sessions must deploy EBPF modules or kernel modules that support the ASRP protocol. Whether these modules are deployed on the client side or the server side corresponds to the two operational modes of the ASRP protocol, respectively:

3.1.1.1. Passive (PSV) Mode

In Passive (PSV) mode, the network nodes and the servers are typically located within the same trusted network domain (e.g., inside a data center). The network nodes back up session state information to the servers and recover sessions from the servers when needed. This mode requires both the network nodes and the servers to support the ASRP protocol. Typical application scenarios include traditional load balancers that provide services through a Virtual IP (VIP), or NFV load balancing network elements offering cloud load balancing services.

3.1.1.2. Active (ACT) Mode

In Active (ACT) mode, the network nodes are typically located within the same trusted network domain as the clients (e.g., a corporate intranet). The network nodes back up session state information to the clients and recover sessions from the clients when needed. This mode requires both the network nodes and the clients to support the ASRP protocol. Typical application scenarios include Source Network Address Translation (SNAT) scenarios (such as internal network devices accessing the internet through an SNAT gateway) or NFV SNAT network elements providing cloud SNAT services.

3.1.2. Two Path Models

3.1.2.1. Symmetric Routing

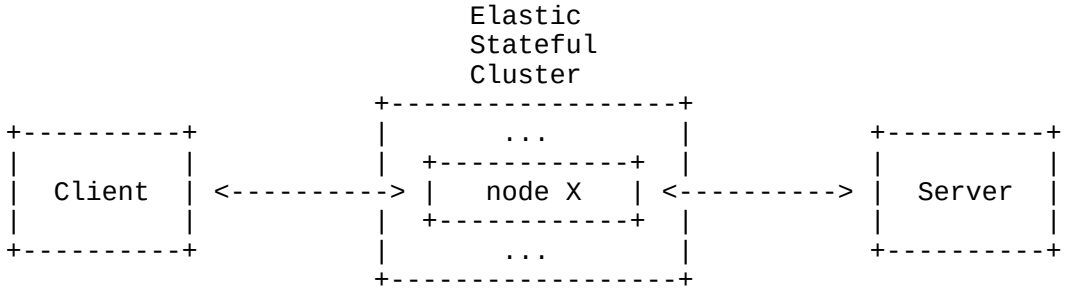

Figure 2: Symmetric Routing

Symmetric routing refers to a path model in which the bidirectional traffic of the same session between a client and a server is always routed to the same node within the cluster. This path consistency is the foundation for the proper functioning of stateful network services (such as NAT and firewalls), ensuring that a single node maintains and processes the complete session state information.

Typical examples that demonstrate symmetric routing include,

1. Active-Standby High Availability Architecture

   In this model, all traffic for a specific session is always handled and maintained by the primary node (e.g., NAT mapping tables, firewall session states). The standby node remains on standby and only takes over when the primary node fails. This architecture inherently ensures symmetric routing of traffic to the primary node.

2. Stateful Load Balancing Cluster with a "Same-Source-Same-Destination" Mechanism

   In this modern extended architecture, network devices (such as OVS or routers) use a "same-source-same-destination" policy to ensure that all packets belonging to the same connection are directed to the same load balancing node, thereby maintaining symmetric routing in a distributed environment.

3.1.2.2. Asymmetric Routing

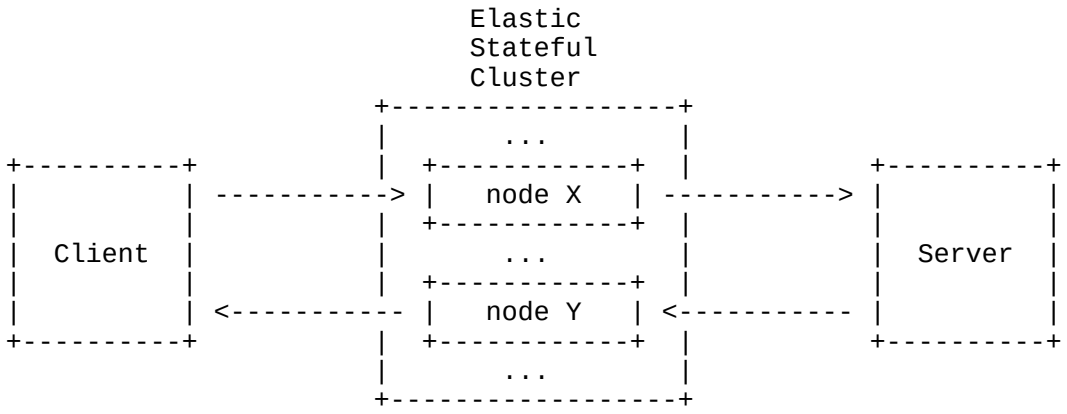

Figure 3: Asymmetric Routing

The bidirectional traffic of the same session may be routed to different nodes within the cluster. Specifically, requests sent from the client to the server may be processed by node X, while responses sent from the server to the client may be processed by node Y. This path inconsistency requires two or more nodes to collaboratively maintain the state information of the same session, posing new challenges to the failure recovery mechanisms of stateful service clusters.

In cloud network environments, asymmetric routing is an extremely common and often default phenomenon. Taking NFV element clusters that require elastic scaling as an example, one of the core goals of their architectural design is to allow nodes to independently and flexibly scale horizontally. In such distributed architectures, without deploying specific traffic steering strategies like "same-source-same-destination," underlying network devices (such as switches or load balancers) typically distribute traffic naturally and evenly across multiple available nodes based on mechanisms like ECMP (Equal-Cost Multi-Path [RFC2991], [RFC2992]), thereby commonly forming asymmetric routing.

3.2. Protocol Message

ASRP achieves distributed backup and on-demand recovery of session state information by exchanging specific protocol messages among clients, servers, and network nodes (such as load balancers and NAT devices). In load balancing scenarios, session states are backed up to individual servers; in Source NAT (SNAT) scenarios, session states are backed up to individual clients.

3.2.1. New session message (NS)

Generated by the network node, it is used to send the initial state information of a session to the designated client (in ACT mode) or server (in PSV mode) for backup when a new session is created. The NS message can be inserted into the original service packet for transmission or independently encapsulated and transmitted.

3.2.2. Hello session message (HS)

Generated by the client, it is used in ACT mode to declare its support for the ASRP protocol to the network node and to trigger the network node to return an NS message to complete session backup. The message also supports in-band or independent transmission.

3.2.3. Query session message (QS)

Generated by the network node, it is used to query the backup party (client or server) for session status when a packet is received from the end where the session is backed up but cannot match a local session. This message is typically transmitted by modifying and echoing the original packet.

3.2.4. Recover session message (RS)

Generated as a response to the QS message by the client or server holding the backup, it contains the state information required to recover the session. The network node parses the RS message and reconstructs the local session, thereby achieving failure recovery.

3.2.5. Encap NS/QS/RS message (ENS/EQS/ERS)

When a packet received by the network node originates from a client or server that does not hold the backup of the session, ASRP messages cannot be transmitted simply by modifying the original packet. Instead, the ASRP message must be encapsulated in a new packet for transmission. In such cases, ASRP defines a format for encapsulating NS, QS, or RS messages within UDP ([RFC0768]) packets for transmission, referred to as ENS, EQS, and ERS messages, respectively.

3.3. Message Flows and Session Recovery Scenarios

This section elaborates on, through a series of typical scenarios, how the ASRP protocol achieves session backup and recovery via message interaction in the event of network node failures under different operational modes. Each scenario details the involved protocol message flows and the processing steps of each entity.

3.3.1. PSV Mode Creation/Recovery Scenarios

3.3.1.1. Scenario 1, direct session creation

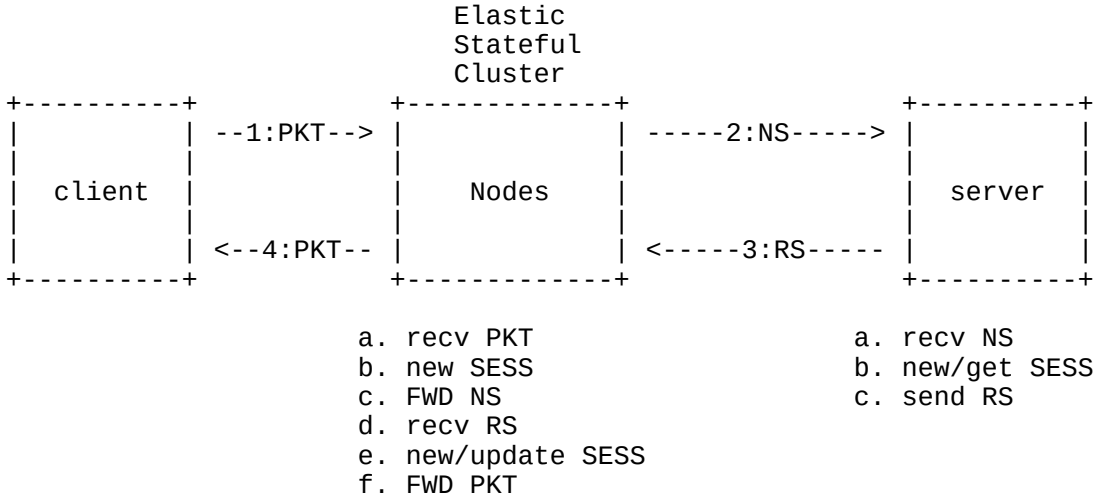

Figure 4: direct session creation

This scenario describes the processing flow when a network node receives a packet that explicitly triggers the establishment of a new session. The most common example is the TCP SYN packet during connection establishment, which indicates that the client is initiating a new connection. Additionally, upon receiving a DNS request packet, the network node may also directly proceed with creating a new session. For convenience, subsequent descriptions will use the TCP SYN packet as the representative example of such packets, without elaborating on other similar packet types.

The processing flow is as follows,

1. *Client sends a packet*: The client sends a packet (e.g., a TCP SYN packet) to the server.

2. *Network node processes and forwards*: Upon receiving this packet, if no matching existing session is found, the network node directly creates a new session. Subsequently, the network node generates an NS (New Session) message, embeds it into the payload of the original packet to form an NS packet, and forwards it to the selected server.

3. *Server responds*: After receiving the NS packet, the server parses the NS message and creates or retrieves the corresponding session state based on its content. The server then generates an RS (Recover Session) message, embeds it into the payload of its response packet (e.g., TCP SYN-ACK) to form an RS packet, and sends it back to the network node.

4. *Network node completes session establishment*: Upon receiving the RS packet, the network node parses the RS message and uses the information within it to finalize the establishment or update of the local session. Next, the network node removes the RS message from the packet, restores the original service response packet, and forwards it to the client.

At this point, the session is successfully established, and subsequent packets can be forwarded normally based on this session.

3.3.1.2. Scenario 2, session recovery triggered by server

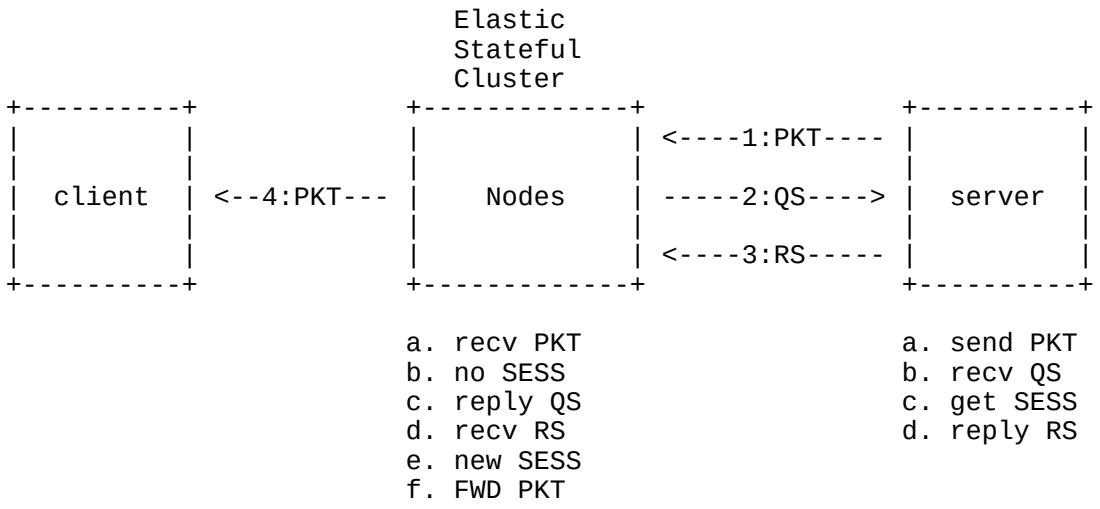

Figure 5: session recovery triggered by server

The server first sends the original packet (PKT) to the client. Upon receiving this packet, the network node checks its local session table. If no corresponding session is found, the node does not forward the packet directly to the client. Instead, it attempts to insert a QS message into the original packet, swaps the source and destination addresses and ports to form a QS packet, and sends it back to the server. After receiving the QS packet, the server looks up the corresponding session state, generates an RS message to replace the QS message in the packet, swaps the source and destination addresses and ports again to form an RS packet, and returns it to the network node. Upon receiving the RS packet, the network node parses the RS message, creates a new session state based on the message content, removes the RS message from the packet, and then forwards the original packet to the client.

This scenario describes the recovery process when a server actively sends a packet to the client, but the intermediate network node cannot find the corresponding session.

The processing flow is as follows,

1. *Server sends a packet*: The server first sends an original packet (PKT) to the client.

2. *Network node queries the session*: Upon receiving this packet, the network node checks its local session table. If no corresponding session is found, it does not forward the packet directly but initiates the recovery process. The network node

generates a QS (Query Session) message, inserts it into the received packet, swaps the source and destination addresses and ports of the packet to form a QS packet, and sends it back to the server.

3. *Server responds to the query*: After receiving the QS packet, the server looks up the locally stored corresponding session state based on the message content, generates an RS (Recover Session) message, and uses it to replace the QS message in the packet. The server then swaps the source and destination addresses and ports of the packet again to form an RS packet and sends it back to the network node.

4. *Network node recovers and forwards*: Upon receiving the RS packet, the network node parses the RS message and creates a new local session based on this information. Subsequently, the network node removes the RS message from the packet, restores the original packet (PKT) initially sent by the server, and forwards it to the client.

At this point, the session is successfully recovered, and the communication link is reestablished.

3.3.1.3. Scenario 3: probe-based session creation/recovery

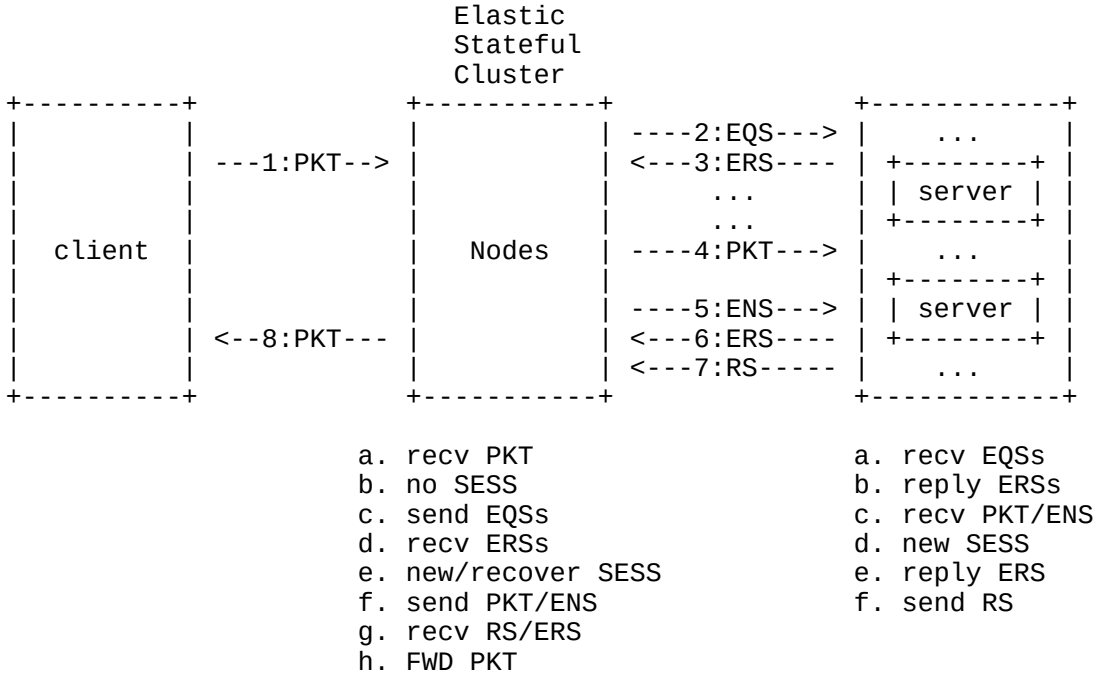

Figure 6: probe-based session creation/recovery

This scenario describes that, in PSV mode, when a network node receives a client packet that neither can trigger the creation of a new session (e.g., a non-TCP SYN packet) nor matches any existing session entry, ASRP relies on the cluster to employ a deterministic backend server selection algorithm (such as Deterministic Bucket-Mapped Consistent Hashing, DBMCH).

By leveraging the deterministic mapping property of the DBMCH algorithm, all packets belonging to the same session will always be scheduled to the same node (or the same group of nodes). ASRP utilizes this property to determine, on the target node and in conjunction with synchronized session state replicas, whether the packet belongs to an existing session.

The processing flow is as follows,

1. *Network Node Receives Packet and Performs Lookup*: The network node receives an original packet (PKT) from the client and checks its local session table. If no matching session is found, the network node uses the DBMCH algorithm to compute a set of potential servers (typically small in number) corresponding to the packet’s 5-tuple.

2. *Parallel Probing Queries*: The network node generates EQS (Encapsulated Query Session) messages and sends them to each candidate server in the set in parallel to probe for the existence of a corresponding session.

3. **Server Responses to Probes

   If a server has the session: That server will reply with an ERS (Encapsulated Recover Session) message containing the state information needed to recover the session.

   If none of the servers has a corresponding session: The process proceeds to the new session creation flow.

4. *Network Node Processes Probe Results*

   Case A: Successful Session Recovery: If the network node receives a valid ERS message from a server, it uses the information therein to recover the session locally. Subsequently, the original packet (PKT) is forwarded to that server based on the recovered session (corresponding to step 4 in the diagram).

Case B: New Session Required: If all EQS probes return responses indicating no session, the network node creates a new session and selects a server for it via the DBMCH algorithm. The network node then generates an ENS (Encapsulated New Session) message and sends it to the selected server (step 5).

5. *Server Acknowledgment and Response*: Upon receiving the ENS message, the server creates a locally associated session and replies with an ERS message carrying no service data as an acknowledgment (step 6). In cases of asymmetric routing, the first service packet (PKT) sent from the server to the client will also carry an RS (Recover Session) message (step 7) to allow other nodes along the path to recover the session.

6. *Final Packet Forwarding*: Upon receiving the service packet carrying the RS message, the network node parses and extracts the session state, removes the RS message from the packet, and forwards the restored original packet to the client (step 8).

Technical Notes:

* *Reason for using ENS/EQS/ERS*: In this scenario, the IP addresses used for communication between the network node and the servers may be completely different from those in the original packet. Therefore, NS/QS/RS messages need to be encapsulated within UDP packets (i.e., as ENS/EQS/ERS) to ensure routing reachability.

* *Algorithm Choice*: The DBMCH algorithm maintains the mapping of all existing sessions unchanged when the number of servers changes, thereby minimizing session disruption caused by scaling. Its principles are detailed in Appendix A.

3.3.2. ACT Mode Creation/Recovery Scenarios

3.3.2.1. Scenario 1: session creation, session recovry triggered by server

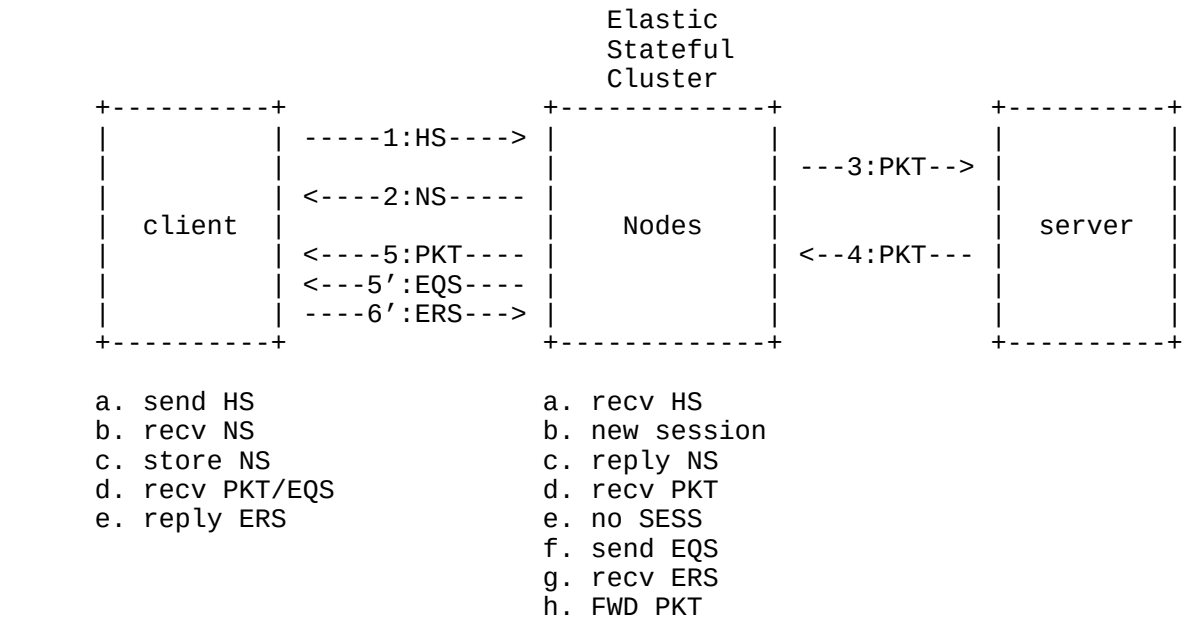

Figure 7: session creation, session recovry triggered by server

In ACT mode, the client first sends a packet containing an HS message to the network node, declaring its support for the ASRP protocol. Upon receiving the HS message, the network node follows the normal procedure to create a new session and should immediately return an NS message to the client to back up the newly created session to the client.

In ACT mode, when a response packet from the server does not match any session, the session needs to be queried from the client. The network node faces the challenge of determining which client to query. To simplify implementation, ASRP recommends using a fixed mapping to locate the client.

This scenario describes the complete process in ACT mode, from session creation to the handling triggered by server response packets. Depending on whether the session exists on the network node, the processing path diverges into two branches.

Processing Flow:

1. *Capability Declaration and Session Creation*: When initiating a new session, the client first sends a packet containing an HS (Hello Session) message to the network node, declaring its support for ASRP. Upon receiving the HS message, the network

node creates a new session following the normal procedure and immediately generates an NS (New Session) message to return to the client, thereby completing the backup of the session state on the client side.

2. *Service Request Forwarding*: The network node forwards the original request packet (PKT) used to establish the session to the server.

3. *Server Response*: After processing the request, the server sends a response packet (PKT) back to the network node.

4. *Network Node Processes the Response (Branching Occurs)*

   Path A (Session Exists, Step 5): If the network node successfully finds a matching session locally, it directly forwards the server's response packet (PKT) to the client according to the session's forwarding rules. This is the normal forwarding path.

   Path B (Session Lost, Steps 5' and 6'): If the network node loses the session due to reasons such as a reboot or failure and cannot find a match, the recovery process is triggered. At this point, the network node generates an EQS (Encapsulated Query Session) message (which may encapsulate the original response packet or be sent separately) and sends it to the corresponding client through a fixed mapping relationship (e.g., port mapping in SNAT scenarios).

5. *Client Assists in Recovery*: Upon receiving the EQS message, the client queries its locally stored session backup and replies with an ERS (Encapsulated Recover Session) message containing the session state to the network node. If the EQS message encapsulates the original packet, the network node's ASRP module, after processing the EQS message, processes the original packet according to the session and then submits it to the normal packet processing module.

6. *Recovery and Completion of Forwarding*: After receiving the ERS message, the network node restores the session state locally.

Technical Notes:

* *Processing Path Branching*: This scenario clearly illustrates the two key branches when the network node processes server response packets in ACT mode: Fast Forwarding (session exists) and Query Recovery (session lost).

* *Flexibility of EQS Messages*: EQS messages can be sent independently or encapsulate the original packet, providing flexibility in balancing protocol overhead and processing efficiency.

* *Core Mapping Mechanism*: Unlike during session creation where the client is known, in the recovery phase, the network node must be able to deterministically locate the client that backed up the session. ASRP recommends using a static, configurable mapping strategy as the foundation for achieving efficient and reliable recovery. If such a mapping cannot be established, it is not recommended to use ASRP in this scenario. For SNAT services, ports can typically be used to map clients. Different clients use configurable, distinct port ranges. When a server packet arrives at the network node, the network node can locate the client through the destination port.

* *Design Choice*: Unlike some mechanisms that rely on keep-alive timers to trigger recovery, ASRP adopts an on-demand query approach. This avoids the additional latency or packet overhead introduced by timer interval settings, enabling fast and precise recovery.

3.3.2.2. Scenario 2: recover session triggered by client

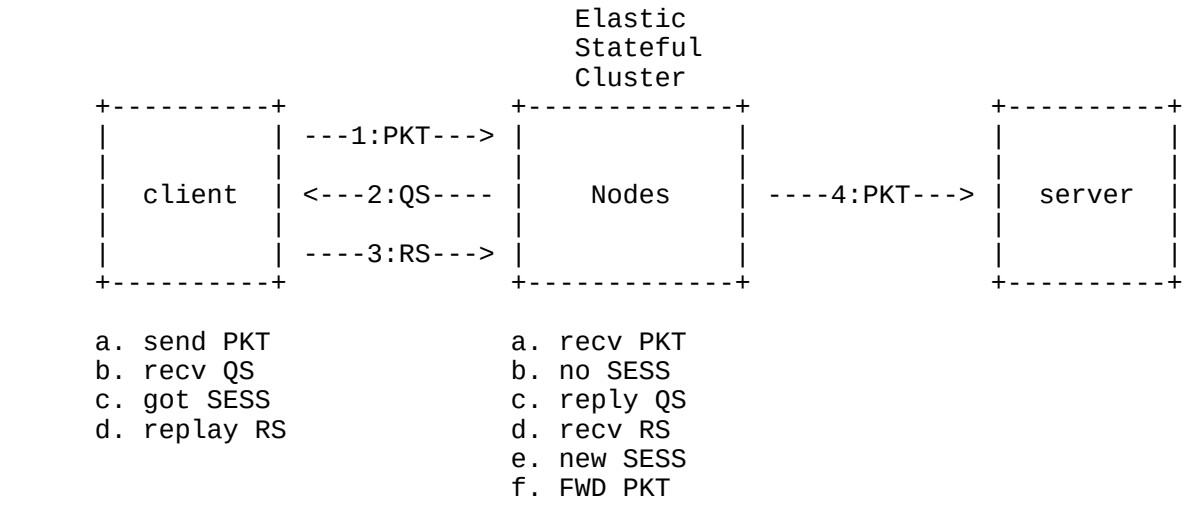

Figure 8: recover session triggered by client

This scenario describes the recovery process in ACT mode when a client actively sends a data packet to the server, but the network node has lost the corresponding session.

The processing flow is as follows:

1. *Client Sends a Packet*: The client sends a data packet (PKT) to the server.

2. *Network Node Triggers a Query*: Upon receiving this packet, the network node checks its local session table. If no matching session is found (and the packet does not contain an HS message), it generates a QS (Query Session) message, inserts it into the received packet, and swaps the source and destination addresses and ports of the packet to form a QS packet. It then sends this QS packet back to the client (i.e., the packet's originator).

3. *Client Replies with State*: After receiving the QS packet, the client queries its locally stored backup of the corresponding session, generates an RS (Recover Session) message, and uses it to replace the QS message in the packet. The client then swaps the source and destination addresses and ports of the packet again to form an RS packet and sends it back to the network node.

4. *Network Node Recovers and Forwards*: Upon receiving the RS packet, the network node parses the RS message and uses the information to create a new session locally. Subsequently, the network node removes the RS message from the packet, restores the original data packet (PKT) initially sent by the client, and forwards it normally to the server.

At this point, the session is successfully recovered, and the request path from the client to the server is reestablished.

4. Protocol Details

4.1. Message Formats

The ASRP protocol defines protocol signature, multiple message types, and their associated packet formats. All messages are encoded using a TLV (Type-Length-Value) structure. All numeric fields use network byte order (big-endian).

All ASRP messages consist of the following five parts:

1. *Type*: 1 byte, used to uniquely identify different ASRP messages.

2. *Flags*: 1 byte, indicating message attributes. Two flag bits are currently defined:

ASRP_F_ACT (0x1): If set, indicates the message belongs to ACT mode; otherwise, it belongs to PSV mode.

ASRP_F_MSG (0x2): If set, indicates this message is transmitted independently (not together with the original service packet); otherwise, it indicates the message is embedded within the original service packet for transmission.

3. *Length*: 2 bytes, representing the total length of the entire ASRP message (including the header).

4. *Session-Tuple*: Carries the address and port information of the session. Its length depends on the address type ([RFC0791] [RFC8200]):

   IPv4: Contains source IPv4 address, destination IPv4 address, source port, and destination port, totaling 12 bytes.

   IPv6: Contains source IPv6 address, destination IPv6 address, source port, and destination port, totaling 36 bytes.

5. Session-Data: A variable-length field that carries the private state information of the network node. Its specific content is implementation-defined.

4.1.1. ASRP Signature

ASRP messages are placed before the packet data. In actual transmission, not all packets carry ASRP messages, so a ASRP Signature or a new TCP option needs to be added to the packet to quickly determine whether it contains an ASRP message.

For the TCP protocol, a new TCP option can be defined in the TCP header to indicate that an ASRP message is located at the beginning of the TCP data. The newly added TCP option type is TCPOPT_ASRP(60), with a length of 2. Its format is as follows:

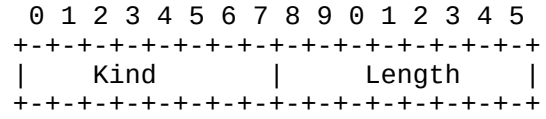

Figure 9: ASRP TCP Option

For other transport-layer protocols, similar to the Proxy Protocol, a 12-byte ASRP Signature is used at the beginning of the data: 0x0D 0x0A 0x0D 0x0A 0x00 0x0D 0x0A 0x41 0x53 0x52 0x50 0x0A

This Signature contains a CRLF pattern, a null byte, and the specific ASCII sequence ASRP. The probability of this sequence occurring in normal data streams is less than 2^{-96}, making it easy to debug and identify: during packet capture analysis, the clear ASRP identifier is visible.

After an ASRP message is inserted into a packet, ASRP Signature must be applied to the packet. This is the default operation and will not be reiterated in subsequent discussions.

4.1.2. NS message format

The NS message is used by the network node to back up session state information to the client or server. There are two types of NS messages, corresponding to IPv4 and IPv6 respectively.

Type Assignments:

ASRP_NS4 = 1 (IPv4)

ASRP_NS6 = 2 (IPv6)

NS (IPv4) Message Format:

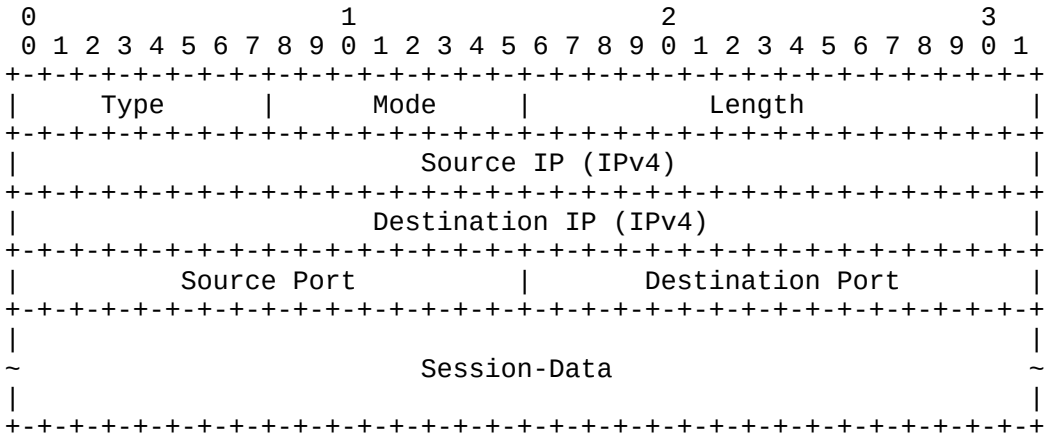

Figure 10: ASRP NS(IPv4) Message format

NS (IPv6) Message Format:

The structure of IPv6 NS messages is the same as IPv4, with the main difference being the IP address length in the Session-Tuple field. IPv6 uses 128-bit addresses, so both Source IP and Destination IP occupy 16 octets each, expanding the entire Session-Tuple field to 36 octets.

The NS message is inserted before the original packet data. The packet carrying the NS message is referred to as an NS packet, and its format is as follows:

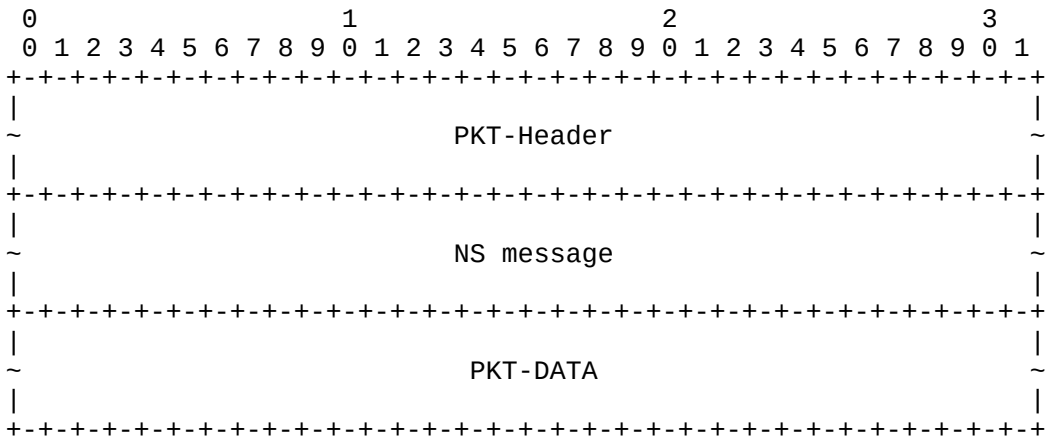

Figure 11: ASRP NS Packet format

In PSV mode, the NS message is inserted into the original packet, and the source and destination addresses remain unchanged. The NS packet is then forwarded to the server.

In ACT mode, the NS message is inserted into a newly allocated packet. The source and destination addresses and ports of the original packet are copied and swapped, and the NS packet is returned to the client.

It is important to note that the NS message internally contains only one Session-Tuple. However, a session on a network node requires two Session-Tuples, representing the connections between the network node and the client, and between the network node and the server, respectively. To improve transmission efficiency, ASRP extracts the other Session-Tuple directly from the packet header of the NS message packet. Similarly, QS and RS messages also require extracting the other Session-Tuple from the message packet header. This will not be elaborated on further in subsequent sections.

4.1.3. HS message format

The HS message is sent by the client during the initial connection establishment phase to declare its support for the ASRP protocol capability to the network node.

Type Assignment:

ASRP_HS = 3

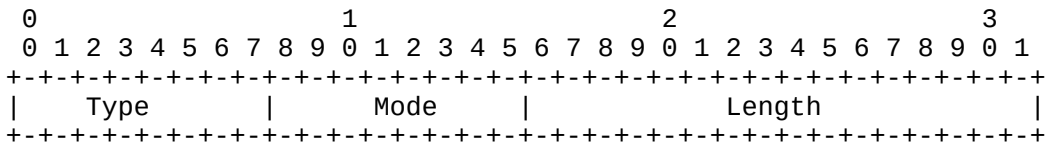

Figure 12: ASRP HS Message format

The HS message is inserted at the beginning of the original packet data. The packet carrying the HS message is referred to as an HS packet, and its format is similar to that of the NS packet, with the only difference being the type of message it carries.

4.1.4. Query Session (QS) message format

The QS message is used by the network node to query session information.

In PSV mode, if the network node receives a packet from the server and cannot find a matching session, it uses the QS message to query the server for the session.

In ACT mode, if the network node receives a packet from the client, and the packet does not contain an HS message while no matching session is found, it uses the QS message to query the client for the session.

Type Assignment:

ASRP_QS = 4

The QS message format is identical to HS message, differing only in the type identifier.

The QS message is inserted before the original packet data, and the source and destination addresses and ports are swapped (in order to return the QS message to the client or server that backs up the session). The packet carrying the QS message is referred to as a QS packet, and its format is similar to that of the NS packet, with the only difference being the type of message it carries.

4.1.5. Recover Session (RS) message format

The RS message is used to recover the network node session. There are three types of RS messages, corresponding respectively to IPv4 sessions, IPv6 sessions, and null sessions.

Type Assignments:

ASRP_RS4 = 5 (IPv4)

ASRP_RS6 = 6 (IPv6)

ASRP_RSN = 7 (null)

When the RS message type is ASRP_RS4 or ASRP_RS6, its message format is identical to the NS message of the corresponding IP version, differing only in the type identifier.

When the RS message type is ASRP_RSN, the message length is 4 bytes, indicating that the message does not contain information for session recovery (i.e., a null session). Its message format is similar to that of the HS message.

The RS message is the response to the QS message. The packet carrying the QS message is referred to as a QS packet, and its format is similar to that of the NS/HS packet, with the only difference being the type of message it carries.

4.1.6. ENS/EQS/ERS message format

When the packet received by the network node originates from a client or server that does not back up the session, the original destination address of the original packet may differ entirely from that of the ASRP message packet. Therefore, encapsulation is required before transmission. ASRP employs a UDP encapsulation format to encapsulate NS/QS/RS message packets. The encapsulated NS/QS/RS messages are referred to as ENS/EQS/ERS messages.

The format of the ENS/EQS/ERS message packet is as follows:

IP + UDP + NS/QS/RS Packet

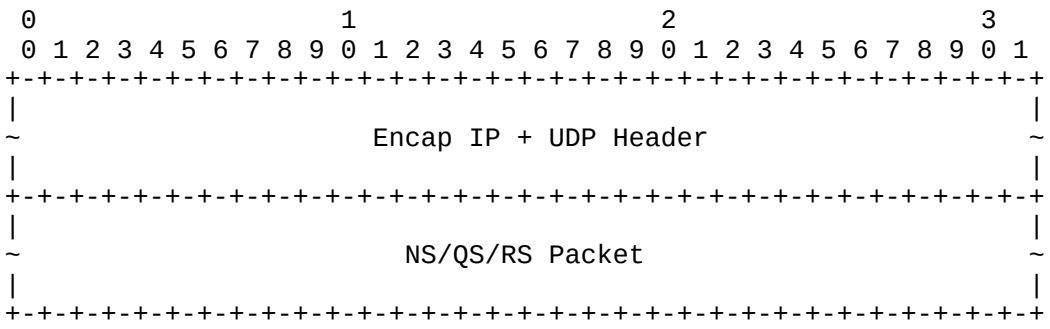

Figure 13: ASRP ENS/EQS/ERS Packet format

The source and destination addresses in the Encap IP header ensure the normal delivery of messages between the network node and the client or server. In the Encap UDP header, the source port is adjustable, while the destination port defaults to 55555 (configurable). The receiving end uses the destination port to distinguish between ENS, EQS, and ERS packets.

4.2. Message Processing

4.2.1. ASRP Signature Detection

For the TCP protocol, the TCP option TCPOPT_ASRP(60) is checked. If this option is present, it indicates that the beginning of the data contains an ASRP message.

For other protocols, the Signature at the beginning of the data is matched first. A successful match indicates that an ASRP message follows the Signature.

In subsequent discussions regarding the processing of ASRP messages in packets, it is assumed that ASRP Signature has already been handled and will not be reiterated.

4.2.2. NS Message Processing

In PSV mode, when the network node receives an original packet such as a TCP SYN, it proceeds directly to the session creation flow. After creating the session, it generates an NS message and sends it to the server.

In ACT mode, after creating a session, the network node needs to allocate a new packet, insert the NS message into this new packet, and return the NS packet to the client. Upon receiving the NS message, the client parses its content and stores the session state information locally.

Avoiding IP Fragmentation:

In PSV mode, the NS message is typically transmitted together with the TCP SYN packet. In ACT mode, the NS message is transmitted independently (not together with the original packet), so the NS packet generally does not exceed the MTU. If inserting the NS message into the original packet would cause it to exceed the MTU, to avoid IP fragmentation, the NS message should be transmitted separately and prioritized, with the Flags field set to ASRP_F_MSG, followed by the transmission of the original packet. This approach of separate transmission (not combined with the original packet) will be referenced subsequently.

Solutions for NS Message Loss:

In PSV mode, the NS message is typically bound to the TCP SYN packet. If the packet is lost, reliance is placed on the TCP protocol’s own retransmission mechanism or application-layer retries to resend the packet carrying the NS message.

In ACT mode, if the client does not receive the NS message, its subsequent data packets will continue to carry HS messages. When the network node receives these subsequent HS messages, it must also return an NS message to the client.

4.2.3. HS Message Processing

HS messages are generated only in ACT mode. When a client initiates a new session, it creates an HS message, inserts it into a packet, and sends it to the network node. After sending the HS message, the client waits for an NS message. If the NS message is not received, the client continues to insert the HS message into subsequent packets it sends, thereby prompting the network node to return the NS message.

Upon receiving an HS message, if the network node does not find a matching local session, it creates a session, generates an NS message, and sends it to the client. If the network node receives subsequent HS messages and finds a matching local session, it should also immediately send an NS message to the client.

Avoiding IP Fragmentation:

Although the HS message is only 4 bytes long, inserting it into the original packet may cause the total size to exceed the MTU. In such cases, the HS message must be sent separately and prioritized, followed by the transmission of the original packet. For the standalone HS message, the Flags field should be set to ASRP_F_MSG to indicate that the message is not transmitted together with the original packet.

Solution for HS Message Loss:

All packets sent by the client must carry the HS message until an NS message is received.

4.2.4. QS Message Processing

The QS message is generated by the network node to query sessions.

In PSV mode, when the network node receives a packet from the server and cannot find a matching session, it returns a QS message to the server to query the session.

In ACT mode, when the network node receives a packet from the client and cannot find a matching session (and the packet does not contain an HS message), it returns a QS message to the client to query the session.

The QS message attempts to be inserted into the original packet, after which the source and destination addresses and ports of the original packet are swapped. This allows the QS message and the original packet to be returned together to the client or server. The benefit of this approach is that there is no need to discard or buffer the original packet.

Avoiding IP Fragmentation:

When inserting the QS message into the original packet, it is necessary not only to check whether the size of the resulting QS packet exceeds the MTU but also to calculate whether the size of the responding RS message packet would exceed the MTU. Since RS messages are generally larger than QS messages, if the calculation indicates that the original packet length of the RS message packet would exceed the MTU, the original packet should be discarded. In this case, the Flags field of the QS message should be set to ASRP_F_MSG to indicate that the QS message is not transmitted together with the original packet.

Solution for QS Message Loss:

Subsequent packets continue to generate QS messages, and attempts to recover the session persist.

4.2.5. RS Message Processing

The RS message is generated by either the client or the server and processed by the network node to recover a session.

When the client or server receives a QS message, it looks up the corresponding session state information locally. If found, it returns this information to the network node by converting the QS message. If the corresponding session state information is not found, it returns an RS message indicating a null session.

The method to convert a QS message packet into an RS message is as follows: replace the QS message in the packet with the RS message, and swap the source and destination addresses and ports of the packet.

Upon receiving a non-null RS message, the network node uses the session information within the RS message to recover the session locally.

4.2.6. ENS/EQS/ERS Message Processing

4.2.6.1. Passive (PSV) Mode

When a network node receives a packet from a client, cannot match an existing session, and cannot directly create a new session, it will communicate with the server using ENS/EQS/ERS messages.

PSV EQS/ERS/ENS Flow:

1. The network node uses the DBMCH algorithm to determine the candidate server set.

2. It sequentially sends EQS messages (encapsulating session queries) to the candidate servers.

3. The network node checks the returned ERS messages from the servers. If a session is found, it recovers the session.

4. If the network node does not find the session after querying all candidates, it uses an ENS message to create a new session.

5. Upon receiving the ENS message, the server returns an ERS message for acknowledgment.

ENS/EQS/ERS messages can be transmitted along with the original packet. To avoid IP fragmentation: If the ENS message packet size exceeds the MTU, send the ENS message separately first; If the EQS/ERS message packet size exceeds the MTU, buffer the original packet and send the EQS/ERS message separately.

Message Loss Handling:

1. EQS message loss: A query timeout occurs; the node deletes the associated data and buffered packet (if any).

2. ENS message loss: The network node continues to send ENS messages periodically until it receives an ERS response.

4.2.6.2. Active (ACT) Mode

When a network node receives a packet from a server and cannot match an existing session, it will communicate with the client using EQS/ERS messages.

ACT EQS/ERS Flow:

1. The network node locates the client using a fixed mapping (e.g., SNAT port mapping).

2. It sends an EQS message to query that client, and the client returns an ERS message.

3. The network node processes the ERS message and recovers the session if found.

EQS/ERS messages can be transmitted along with the original packet. To avoid IP fragmentation: If the EQS/ERS message packet size exceeds the MTU, discard the original packet and send the EQS message separately.

Message Loss Handling: Lost EQS/ERS messages are handled by triggering a new EQS query upon the arrival of subsequent packets from the server.

5. Security Considerations

5.1. General Defenses Against Message Forgery Attacks

The security design of the ASRP protocol is based on its typical deployment model.

Deployment Boundaries and Access Control: ASRP recommends deploying network nodes and the clients or servers that back up sessions within the same trusted internal network domain. Under this model, all ASRP protocol packets communicate within the internal address space. By implementing appropriate network segmentation (e.g., using firewall policies or security groups) and rigorously checking the source addresses of packets, forged ASRP packets originating from untrusted external networks can be effectively prevented from reaching the target nodes.

Session Legitimacy Validation: When processing any ASRP packet that may establish a new session (e.g., packets carrying NS or RS messages), network nodes must perform basic validation according to the specific policies of the upper-layer application or service. For example, in a load balancing scenario, a node should verify whether the session points to a known and healthy server; in a NAT scenario, it should validate whether the address translation complies with predefined rules. This prevents the establishment of illegal sessions at the application level.

Internal Threat Assessment: Even if an attacker is located within the trusted network and can forge ASRP packets, the scope of impact is inherently limited. The attacker can only forge sessions where they themselves are the endpoint (e.g., impersonating a client to request recovery of a non-existent connection). Such forged sessions are indistinguishable in form from those established through normal access, do not directly endanger the security of other users or nodes, and cannot elevate the attacker’s privileges or grant access to unauthorized resources.

Trade-offs Regarding Enhanced Authentication: Theoretically, stronger authentication mechanisms could be introduced for ASRP, such as having the network node generate a random number (Cookie) during session backup and include it in the transmission, then requiring the other party to echo this Cookie during the recovery phase. However, considering that ASRP primarily operates within trusted domains and faces a limited external attack surface, introducing such mechanisms would increase protocol complexity and processing overhead. Therefore, this protocol does not recommend mandating such inter-node Cookie validation in its base design.

5.2. Mitigation Against EQS/ERS Flood Attacks

In the PSV mode's probe-based session creation/recovery scenario (Section 2.4.1, Scenario 3) and the ACT mode scenario where recovery is triggered by server traffic (Section 2.4.2, Scenario 1), a network node may send a large number of EQS query messages to multiple backup parties upon session loss. This behavior, if maliciously exploited or resulting from a fault, could lead to flood attacks.

To mitigate such risks, implementers should consider the following protective measures:

Message Aggregation: When a network node needs to query multiple sessions from the same backup party (e.g., a specific server) within a short timeframe, the implementation SHOULD support aggregating multiple QS messages into a single EQS packet for transmission. This can significantly reduce the number of control packets, lowering network and processing overhead.

Rate Limiting and Traffic Shaping: Each network node MUST implement monitoring and limiting of the rate at which EQS packets are sent. A reasonable threshold (e.g., the maximum number of EQS packets allowed per second) should be established. When the rate exceeds this threshold, the node should adopt a packet discarding policy, such as:

Discarding newly arrived service packets that would trigger queries, or, discarding lower-priority pending EQS query requests.

This measure aims to prevent EQS packet floods caused by node failures, configuration errors, or malicious traffic, thereby protecting the backup parties (clients or servers) from resource exhaustion attacks. The parameters for rate limiting should be configurable to adapt to deployment environments of different scales.

State Monitoring and Alerting: Implementations are advised to log the frequency and patterns of EQS/ERS messages. Abnormally high query rates should trigger logging and system alerts, enabling administrators to promptly detect potential network issues or security attacks.

6. IANA Considerations

This document defines the Application-layer Session Recovery Protocol (ASRP), an application-layer protocol whose message types and internal identifiers are scoped to its own specification. Therefore, no registration in the "Assigned Internet Protocol Numbers" registry is required.

However, to support in-band signaling over TCP and encapsulated control messaging over UDP, the following IANA allocations are requested.

6.1. TCP Option for ASRP

The ASRP protocol uses a TCP option to signal that the beginning of the TCP payload contains an ASRP message (e.g., HS, NS, or RS). This option is defined as follows:

Option Name: TCPOPT_ASRP

Option Kind: To be assigned by IANA

Length: 2 bytes

The format and semantics of this option are specified in Section ASRP Signature.

IANA is requested to assign a value from the "TCP Option Kind Numbers" registry (within the "TCP Parameters" registry) for TCPOPT_ASRP and to reference this document.

Note: Early implementations used Kind 60 for experimentation. However, this document requests a permanent, unassigned value suitable for standards-track use.

6.2. UDP Port for Encapsulated ASRP Messages

Encapsulated ASRP control messages (ENS, EQS, ERS) are transmitted over UDP between network nodes and servers or clients. To support interoperable testing and early deployments, this document requests a standardized UDP port assignment.

Service Name: asrp-encap

Transport Protocol: udp

Port Number: To be assigned by IANA (from the User Ports range, 1024-49151)

Description: UDP port for receiving encapsulated ASRP protocol messages (ENS/EQS/ERS).

For experimental implementations and interoperability testing prior to IANA assignment, *UDP port 55555* MAY be used as a temporary default. This port falls within the dynamic/private port range (49152-65535) reserved for local or temporary use and documentation examples [RFC6335].

IANA is requested to assign a permanent port number in the "User Ports" range (1024-49151) for the "asrp-encap" service in the "Service Name and Transport Protocol Port Number Registry", with a reference to this document.

7. References

7.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

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

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[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>.

[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>.

[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>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

7.2. Informative References

[RFC2991] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection", RFC 2991, DOI 10.17487/RFC2991, November 2000, <https://www.rfc-editor.org/info/rfc2991>.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, DOI 10.17487/RFC2992, November 2000, <https://www.rfc-editor.org/info/rfc2992>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <https://www.rfc-editor.org/info/rfc7413>.

Appendix A. Deterministic Bucket Mapping Hash, DBMH

A.1. Principles of the DBMH Algorithm

When the passive (PSV) mode of ASRP is applied to load balancing scenarios, the core requirement is to stably map client connections (identified by a 5-tuple) to a backend server. Even when the server cluster scales in or out, the mapping of existing connections should remain unchanged to ensure session recoverability. The DBMCH (Deterministic Bucket Mapping Consistent Hash) algorithm achieves this goal by introducing the concepts of virtual buckets, server weight, and bucket weight, combined with ASRP’s session probing capability.

A.1.1. Core Concepts and Design Goals

1. *Virtual Buckets*: Set a fixed number (N, default 65536) of virtual buckets. N should be far greater than the maximum expected number of servers to ensure balance. Once set, the value of N does not change.

2. *Fixed Mapping from Sessions to Buckets*: Through a stable hash function, deterministically map the 5-tuple of each connection to a virtual bucket in the range 0, N-1. This ensures that all packets of the same connection always hit the same bucket.

3. *Server List in a Bucket*: Each virtual bucket maintains an ordered list of servers. The first server in this list is called the preferred server of the bucket, specifically used to handle new connections.

4. *Server Weight*: The weight of a server is defined as the total number of times the server appears in the server lists of all N virtual buckets (regardless of whether it is the preferred server). It reflects the "distribution breadth" of the server in the global mapping.

5. *Bucket Weight*: The weight of a virtual bucket is defined as the sum of the weights of all servers in its server list. It reflects the "global influence" of the bucket.

6. *Core Design Goals*

   Session Stability: When the cluster scales in or out, the preferred server of the virtual bucket where an existing connection resides never changes. This ensures uninterrupted recovery of these connections through the ASRP mechanism.

   Load Balancing: After scaling, by adjustment, ensure that each server serves as the preferred server an equal number of times, so that new connections are evenly distributed.

A.1.2. Dynamic Scaling Steps of the Algorithm

The ingenuity of the algorithm lies in achieving the above goals by adjusting weights and preferred servers when the server set changes.

1. *Scaling Out* (Add m servers, currently n servers, target total n+m)

   Goal: The number of times each server is preferred should be adjusted to N/(n+m).

   Step 1, Selection and Retention: Sort the existing n servers by weight from lightest to heaviest. For each existing server, from all virtual buckets that currently include this server, retain the heaviest N/(n+m) buckets, and ensure that the preferred server of these buckets is set to this server.

Step 2, Allocation of New Buckets: After step 1, there will be a batch of remaining virtual buckets. Sort these buckets by weight from lightest to heaviest. Then, allocate buckets to each new server in turn and cyclically until each new server obtains N/(n+m) buckets. Set the preferred server of these allocated buckets to the corresponding new server.

2. *Scaling In* (Remove some servers, leaving k servers)

   Goal: The number of times each remaining server is preferred should be adjusted to N/k.

   Step 1, Cleanup and Calculation: Remove the deleted servers from the lists of all virtual buckets. Calculate the weight N/k that each of the remaining k servers should obtain.

   Step 2, Priority Satisfaction: Sort the remaining k servers by weight from lightest to heaviest. For each server, from all available virtual buckets, allocate the heaviest N/k buckets and make this server the preferred server of these buckets. Record which servers have obtained enough buckets.

   Step 3, Make Up the Shortfall: After step 2, there may still be some servers that have not obtained enough buckets (less than N/k). Match the servers that are still short with the remaining virtual buckets (sorted by weight from lightest to heaviest) and allocate them in turn until all remaining servers have reached N/k preferred buckets.

Summary: The DBMCH algorithm dynamically manages "weight" and "preferred server" to achieve uniform load distribution of new connections after cluster scaling while ensuring absolute stability of all historical connection mappings. Combined with ASRP's session probing capability, it constitutes a robust and efficient scaling solution for stateful services.

A.2. Estimation of Server Count per Virtual Bucket in DBMCH

The core conclusion is: The length of the server list within a virtual bucket grows slowly and has a definite upper bound. Its growth is primarily related to two factors: (1) the number of scaling operations, and (2) the ratio of the number of newly added servers to the current scale during each scaling operation.

Quantitative analysis of the following two typical scaling strategies clearly demonstrates this characteristic:

1. *Incremental Scaling Strategy*

Assume an initial server count of K, and each scaling operation adds K new servers (i.e., the scale doubles each time). After 8 consecutive scaling operations, the total number of servers reaches 9K. During this process, the mathematical expectation (which can approximate the average number) of servers in any virtual bucket is roughly 1 + 1/2 + 1/3 + ... + 1/8 approximately equal to 2.7. Therefore, the maximum number of servers per bucket is 3, and the minimum is 2.

Scale Example: This means that scaling from a small cluster of 4 servers to 36 servers, or from 32 servers to 288 servers, the maximum number of servers per bucket remains merely 3.

2. *Aggressive Scaling Strategy*

Assume an initial server count of K, and each scaling operation adds 8K new servers (i.e., significant expansion). After 4 scaling operations, the total number of servers reaches 33K. The mathematical expectation of servers per bucket is roughly 1 + 8/9 + 8/17 + 8/25 + 8/33 approximately equal to 2.92. Similarly, the maximum number of servers per bucket is 3, and the minimum is 2.

Scale Example: This means scaling from 4 servers to 132, or from 32 servers to 1056, the maximum number of servers per bucket still stabilizes at 3.

Implications and Guidance

The above analysis demonstrates that the DBMCH algorithm possesses excellent scalability. Even when the cluster scale grows by tens or even hundreds of times, the number of server probes required to recover any connection remains extremely limited (typically no more than 3). This theoretically ensures the efficiency of the ASRP session recovery mechanism, making it suitable for large-scale, elastically scaling cloud-native environments.

In practical operations, it is recommended to adopt relatively centralized scaling batches (such as the aggressive strategy mentioned above) to better control the average probing cost.

A.3. Advantages and Disadvantages of the DBMCH Algorithm

A.3.1. Advantages

1. *Complete Session Stability*

During dynamic scaling (expansion or contraction) or node failures, the algorithm guarantees that the mapping of all existing connections remains absolutely unchanged. This provides a solid foundation for ASRP-based session recovery and is a key factor in achieving high availability.

2. *Excellent Load Balancing for New Connections*

   After each scaling operation, the algorithm reallocates virtual buckets so that each server serves as the preferred server for an equal number of buckets. This ensures that newly established connections are evenly distributed across all available servers, effectively preventing load imbalance.

3. *Enhanced Robustness through Synergy with ASRP*

   The algorithm maintains a server list for each virtual bucket, providing a well-defined target set for ASRP’s session probing mechanism (EQS/ERS). Working in synergy, they enable precise location and recovery of any connection after a node failure, achieving connection recoverability at the cluster level.

A.3.2. Limitations and Considerations

1. *Probe Overhead and Server Count per Bucket*

   For connectionless protocols (e.g., UDP) or TCP connections that need recovery, ASRP must sequentially probe the server list within a bucket via EQS messages. In the worst-case scenario of frequent scaling operations and continuous cluster growth, the number of servers in a bucket may gradually accumulate, leading to decreased probing efficiency.

2. *Operational Recommendations*

   To control the average length of bucket lists, it is recommended to adopt a batch-operation strategy when planning scaling events: "reduce the frequency of operations, but adjust more servers each time."

3. *Optimization Directions*

   Since new connections are handled only by the preferred server, it can be envisioned that once all old connections previously mapped to non-preferred servers are closed, those servers can be safely removed from the corresponding virtual bucket lists. Ideally, after some time, most virtual buckets will retain only the preferred server, bringing the average list length close to

1, which would significantly reduce probing costs. For the few remaining long-lived connections, special handling mechanisms (such as maintaining a separate small mapping table) could be considered to further reduce the number of servers in virtual bucket lists.

In summary, the DBMCH algorithm provides an innovative solution for seamless scaling and high availability of stateful services. Its value is particularly pronounced when combined with distributed session state backup (ASRP). By understanding its characteristics, designers and operators can maximize its benefits and control its potential overhead through reasonable cluster sizing planning and operational strategies.

Appendix B. Acknowledgments

The authors would like to thank all individuals who have provided valuable feedback and contributions during the development of this document.

Authors' Addresses

Zhaoyu Luo (editor)
CMCC
No. 58 Kunlunshan Road
Suzhou
215000
China
Email: luozhaoyu@cmss.chinamobile.com

Haishuang Yan
CMCC
No. 58 Kunlunshan Road
Suzhou
215000
China
Email: yanhaishuang@cmss.chinamobile.com