Service Affinity Solution based on Transport Layer Security (TLS)

[draft-wang-tcpm-tcp-service-affinity-option]

Wei Wang (China Telecom)

Aijun Wang (China Telecom)

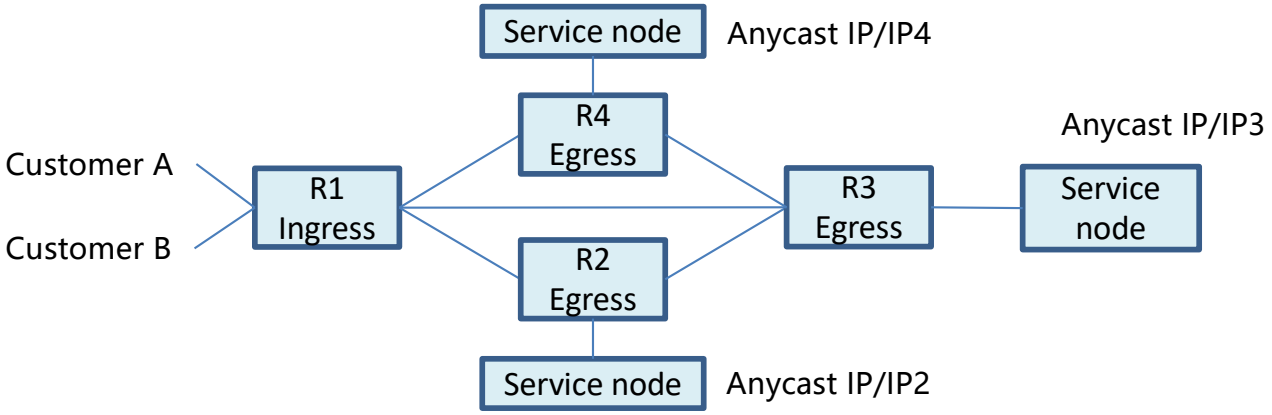
Mohit Sahni (Palo Alto Networks)

Ketul Sheth (Palo Alto Networks)

IETF 124, Nov. 2025

- Background and motivation of this draft
- Considerations on the existing solutions
- Proposed Solutions
- Further Actions

Background and motivation of this draft

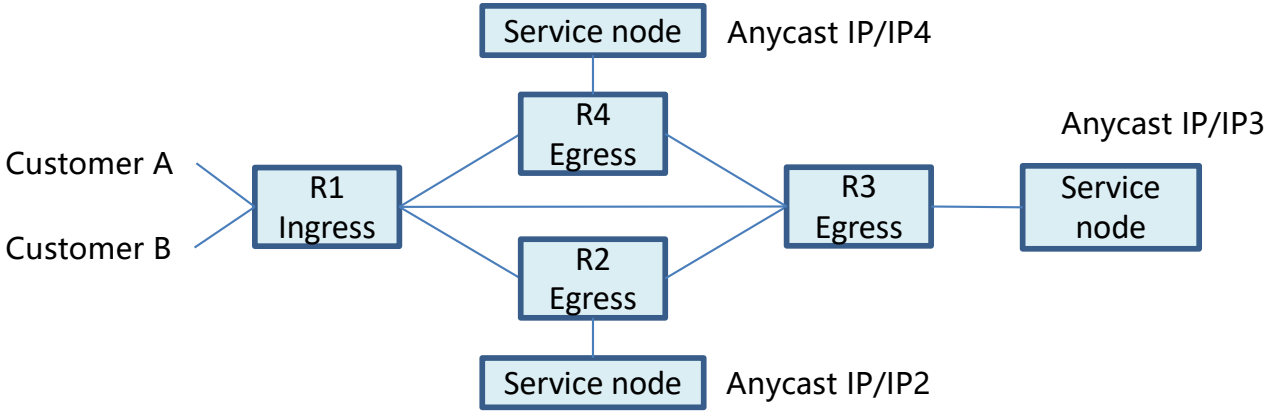


Anycast Services Scheduling scenario

- Service nodes provide the same service function may be deployed different resource pools, and use the same anycast IP address.
- When customer A accesses to the service:
  - Customer A sends packet to R1
  - R1 determine the optimal service node for customer A based on the network status.

- The network status is constantly changing, different customers may be scheduled to different service nodes. For customers who have established connections, the service node providing services must remain unchanged.

Background and motivation of this draft



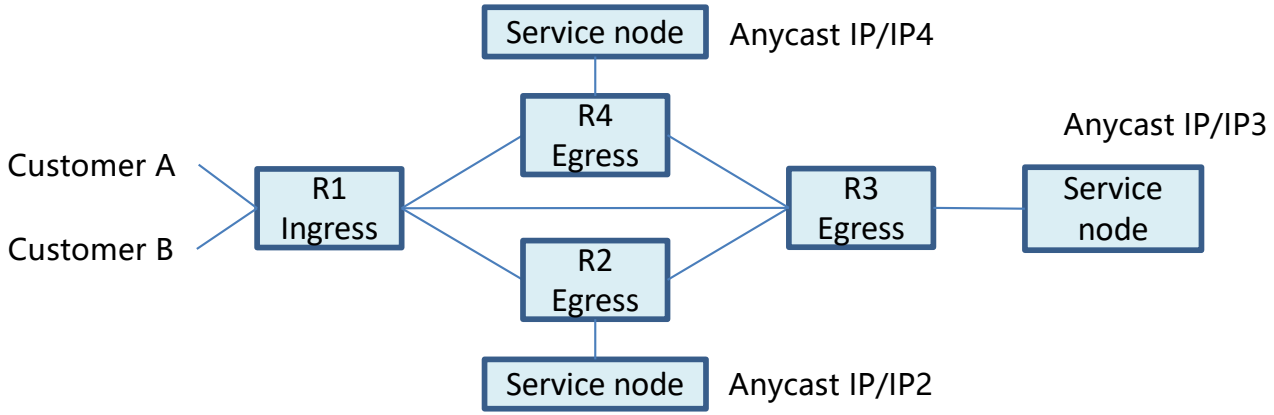
Anycast Services Scheduling scenario

- The current solutions need to maintain the customer-based connection status table in ingress and egress routers.
- We propose a solution for the service affinity between client and server based on Transport Layer Security (TLS), which can realize the comprehensive scheduling based on real-time network status. This solution eliminates the need to maintain customer-based connection status tables for network devices, and improves the flexibility and scalability of large-scale deployment of anycast services scheduling.

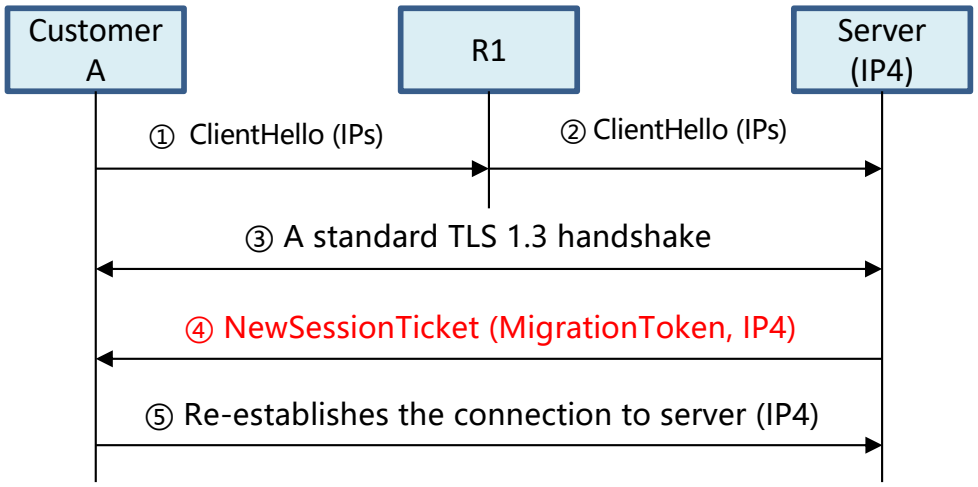
Considerations on the existing solutions

- Load-balancer
  - This deployment may lead to the load balancer become the bottleneck when the traffic increases.
  - Direct traffic redirection and traffic scheduling between the client and server can avoid the bottleneck of load balancer.
- HTTP redirection
  - It mainly involve the communication between client and server.
  - Both client and server do not perceive changes in network status and cannot achieve comprehensive optimization based on network status and computing resource status.
- DNS redirection
  - The caching time of DNS records will take some time for the modification to take effect
  - customer experience will be affect.

Proposed Solution



Anycast Services Scheduling scenario



Procedures for the service affinity solution

Step 1: Customer A access to the service. It sends a initial `ClientHello` message which includes the `migration_support` extension to R1. The destination address of this packet is set to the anycast IP address of this service (IPs).

Step 2: R1 schedules the customer A's service connection request according to the real-time status of the network and computing resources, and determine that the server (IP address = IP4) will provide services to customer A.

Step 3: the server completes a standard TLS 1.3 handshake.

Step 4: the server sends a `NewSessionTicket` message to enable standard PSK-based session resumption. It carry the `MigrationToken`, an authorization credential that refers to IP4.

Step 5: customer A re-establishes the connection to server through IP4.

Proposed Solution

This draft defines 3 extensions to TLS:

1. **migration_support** extension

This extension is sent in the `ClientHello` to indicate support for this protocol. The `extension_data` field of this extension is zero- length.

```
struct { } MigrationSupport;
```

2. **migration_token** extension

This extension is sent in the `NewSessionTicket` message and contains the `MigrationToken` structure. It is also sent by the client in the `ClientHello` during a migration attempt.

```
enum { ipv4(0), ipv6(1) } IPAddressType;

struct {
    IPAddressType type;
    select (IPAddress.type) {
        case ipv4: uint8 ipv4_address[4];
        case ipv6: uint8 ipv6_address[16];
    };
    uint16 port;
} IPAddress;

struct {
    IPAddress target_address;
    opaque session_id<32..255>;
    uint64 expiry_timestamp;
    opaque nonce<16..255>;
    opaque signature<32..255>;
} MigrationToken;
```

Where:

- **target_address:** An `IPAddress` structure specifying the destination IP address (v4 or v6) and port for the client to reconnect to.
- **session_id:** A unique identifier for the TLS session.
- **expiry_timestamp:** A 64-bit unsigned integer representing the Unix timestamp after which this token becomes invalid.
- **nonce:** A unique, single-use value generated by the server to prevent replay attacks.
- **signature:** An HMAC tag providing integrity and authenticity.

Proposed Solution

This draft defines 3 extensions to TLS:

3. **migrate_notify** alert

The `migrate_notify` alert is a notification-level alert. Upon receiving this alert, the client SHOULD initiate the migration process as described in Section 3.3. It does not indicate a protocol error.

```
enum {
    ...,
    migrate_notify(TBD3),
    ...
} AlertDescription;
```

Proposed Solution

Security considerations

- **Token Integrity and Authenticity:** The `MigrationToken` is protected by an HMAC signature keyed with a secret derived from the session's master secret. This prevents forgery and ensures the token was generated by a server with access to the original session's cryptographic state.
- **Session Binding:** The inclusion of the session-derived `session_id` in the signature calculation ensures that a token issued for one session cannot be used to authorize the migration of a different session.
- **Replay Attacks:** The `nonce` field in the `MigrationToken` prevents an attacker from capturing and replaying a token. The server infrastructure is responsible for tracking and invalidating used nonces.
- **Operational Inflexibility:** Including the `target_address` in the initial token makes the migration path static. The server cannot dynamically choose a new destination at the time of migration, which reduces operational flexibility.

Further Action

- Comments?

wangw36@chinatelecom.cn
wangaj3@chinatelecom.cn
msahni@paloaltonetworks.com
ksheth@paloaltonetworks.com
IETF124