TLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 10 October 2026

W. Wang
A. Wang
China Telecom
M. Sahni
K. Sheth
Palo Alto Networks

8 April 2026

# Service Affinity Solution based on Transport Layer Security (TLS)
draft-wang-tls-service-affinity-01

## Abstract

This draft proposes a service affinity solution between client and server based on Transport Layer Security (TLS). An extension to Transport Layer Security (TLS) 1.3 to enable session migration. This mechanism is designed for network architectures, particularly for multi-homed servers that possess multiple network interfaces and IP addresses.

## 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 10 October 2026.

Table of Contents



## 1. Introduction

The rapid growth of internet services and the increasing complexity of network architectures have created a demand for more flexible and resilient connections between clients and servers. Modern service deployments often utilize multi-homed servers. These are systems that are equipped with multiple network interfaces and IP addresses. This architecture enhances availability, balances loads, and optimizes routing based on dynamic network conditions.

In such environments, clients often need to migrate their connections from one server IP address to another. Service continuity must be maintained during traffic migration. This necessity can arise from changes in network topology, server maintenance requirements, or the need to balance computational resources across different service nodes.

In traditional solutions for maintaining service affinity or facilitating migration, each device needs to maintain a customer-based connection status table. This table will not change dynamically with the change of network status and computing resources. Moreover, the network devices should keep large amounts of status table to keep the service affinity for every customer flow. As the number of sessions increases, this table will grow in size, and an excessive number of sessions will impose pressure on the device.

Besides, in the load balance scenario, a load balancer is usually put in front of all the physical servers so that all the packets sent and received by the physical servers should pass through the load balancer. This deployment may lead to the load balancer become the bottleneck when the traffic increases.

HTTP redirection enables automatic page jumps by having the browser automatically send a new request based on the specific response status code and the value of the Location field returned by the server. 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 can redirect customer requests from one domain name to another by modifying DNS resolution records, or change the resolution result of a domain name to point to a different server IP address. However, due to the caching time of DNS records, it takes some time for the modification to take effect, which may result in customers still accessing servers that have been taken offline, thereby affecting customer experience.

We propose a solution for the service affinity between client and server by extending TLS 1.3. This proposal is designed for environments where operational simplicity and migration speed are paramount. It intentionally omits the path validation steps to minimize the latency of the migration process. Furthermore, it simplifies the trigger mechanism by using a new TLS alert, which is a direct and unambiguous signal.

## 2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] .

## 3. Motivation and design rationale

In distributed cloud and edge computing architectures, traditional session identification based on static IP addresses can no longer meet the demands of dynamic networks. This proposal chooses to implement service affinity at the TLS layer, rather than through redirection at the application layer (e.g., HTTP) or the network layer, based on the following three core dimensions:

First, TLS 1.3 [RFC8446] provides a secure channel for negotiating connection parameters without exposing sensitive data to network intermediaries. By conveying migration instructions and cryptographic material within the handshake, the solution avoids the visibility and interference issues associated with in-band application-layer signaling (e.g., HTTP redirects) or out-of-band network-layer mechanisms.

Second, the TLS session resumption framework offers a natural abstraction for session continuity. The proposed extension leverages the existing NewSessionTicket message to bind migration authorization to the session state. This approach ensures that migration tokens are cryptographically bound to the original session keys, preventing unauthorized redirection or session hijacking.

Third, integrating migration into the handshake enables 0-RTT resumption at the new endpoint. When a client migrates, it presents the ticket containing the migration extension, allowing the new server instance to validate the token and resume the session without performing a full cryptographic handshake. This minimizes the latency impact of migration, which is critical for real-time applications.

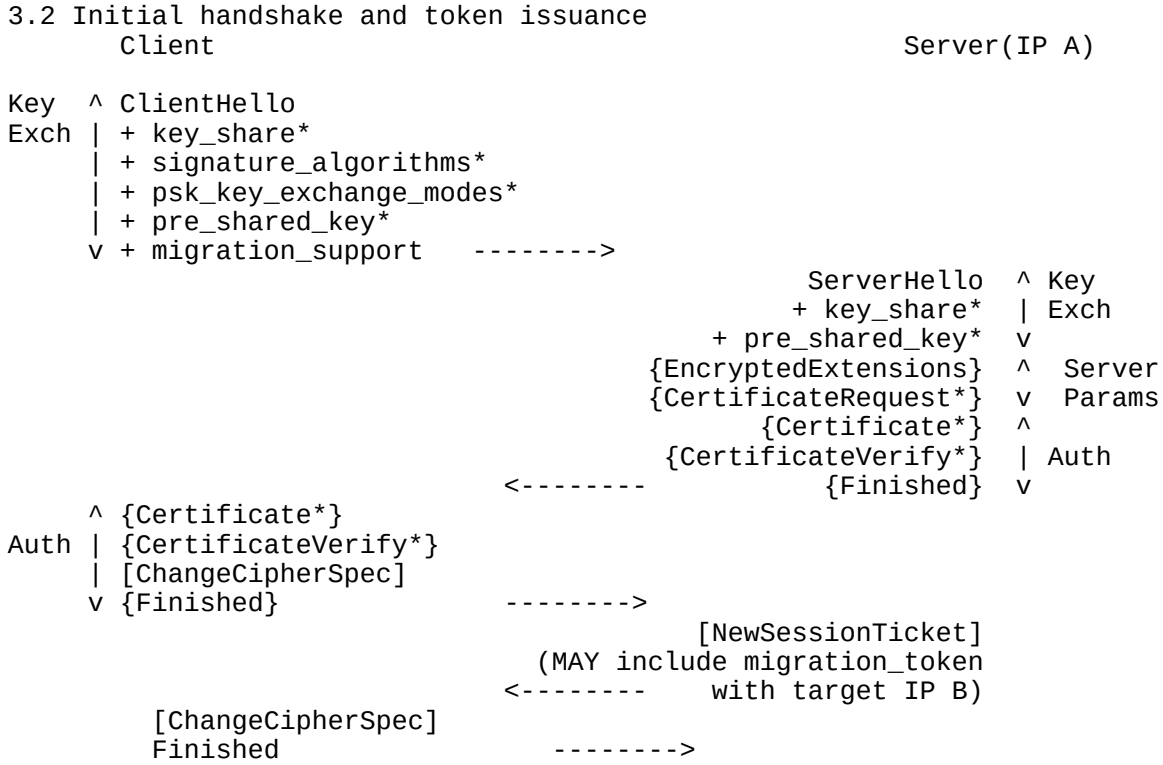
Critically, this design does not require changes to the application data flow. It is transparent to both the application and the network path, making it compatible with any protocol running over TLS.

## 4. Procedures of the proposed solution

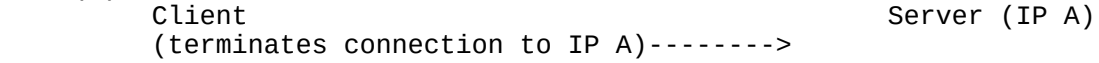
### 4.1. Message flow of the overall procedure

The message flow of the procedures of service affinity mechanism based on TLS are shown in Figure 1.

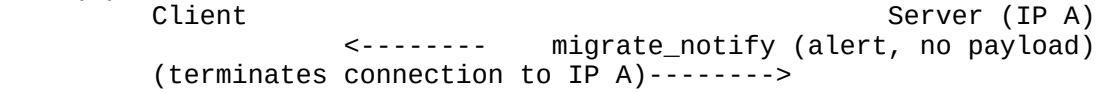
3.2 Initial handshake and token issuance



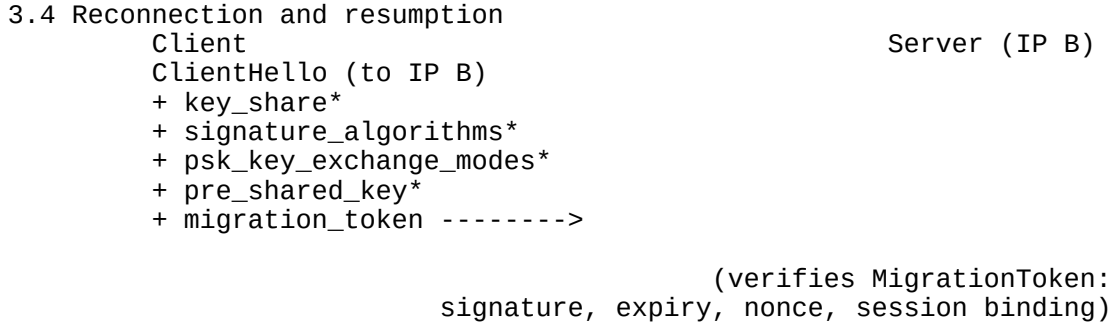

3.3 (a) Client initiated:




3.3 (b) Server initiated:




3.4 Reconnection and resumption



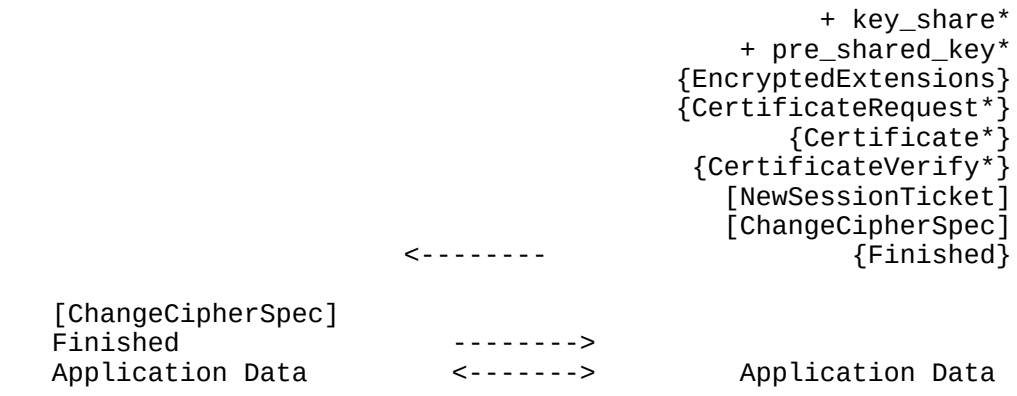



Figure 1: service affinity mechanism based on TLS

## 4.2. Phase 1: initial handshake and token issuance

1. A client supporting this mechanism includes the ‘migration_support‘ extension in its initial ‘ClientHello‘ message to the server at IP A. This extension is empty and serves only to signal capability.

2. The server at IP A completes a standard TLS 1.3 handshake.

3. After the handshake is complete, the server sends a ‘NewSessionTicket‘ message to enable standard Pre-Shared Key-based (PSK-based) session resumption. Within this message, the server MAY include the new ‘migration_token‘ extension. This extension contains the ‘MigrationToken‘, an authorization credential that includes the pre-determined destination (IP B) for a future migration.

## 4.3. Phase 2: migration trigger

a) If the session migration is triggered by the client, the client can directly switch the session to the new server according to business requirements.

b) If the session migration is triggered by the server, it performs as follow:

1. At a later point, the server at IP A initiates the migration.

2. The server sends a new TLS alert, ‘migrate_notify‘, over the encrypted and authenticated connection. This alert has no payload and serves as a simple, direct instruction for the client to initiate the migration process.

### 4.4. Phase 3: reconnection and resumption

1. The client inspect its stored ‘MigrationToken‘. If a valid token exists, it extracts the target IP address and port, terminates its connection to IP A, and initiates a new TLS connection to IP B.

2. The client sends a ‘ClientHello‘ message to IP B. This message MUST include:

* The standard ‘pre_shared_key‘ extension, containing the session ticket identity received from IP A.

* The ‘migration_token‘ extension, containing the ‘MigrationToken‘ it received from IP A.

3. The server at IP B uses the PSK identity to retrieve the session state. It then MUST validate the ‘MigrationToken‘, confirming its signature, expiration, and nonce, and verifying that the token is cryptographically bound to the session.

4. If all checks pass, the server accepts the PSK and completes the abbreviated handshake.

## 5. Detailed formats

This section defines the structure of the new protocol elements, following the presentation language of [RFC8446].

### 5.1. 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;
```

### 5.2. 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.

```
MigrationToken migration_token;
```

The ‘MigrationToken‘ is a credential that authorizes the migration of a specific session to a pre-determined destination.

```
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, derived from the session’s ‘resumption_master_secret‘ using an HKDF-Expand function.

* 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. The signature is computed over a concatenation of the ‘target_address‘, ‘session_id‘, ‘expiry_timestamp‘, and ‘nonce‘ fields. The key for the HMAC MUST be derived from the ‘resumption_master_secret‘.

### 5.3. migrate_notify alert

This proposal introduces a new alert type to trigger the migration.

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

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.

## 6. Use case

Computing-Aware Traffic Steering (CATS) provides a compelling use case for TLS-layer session migration. In CATS architectures, traffic is dynamically steered to optimal endpoints based on real-time network conditions, server load, and computational resource availability.The scenario is shown as Figure 2, and the transmission process of packets is shown in Figure 3.

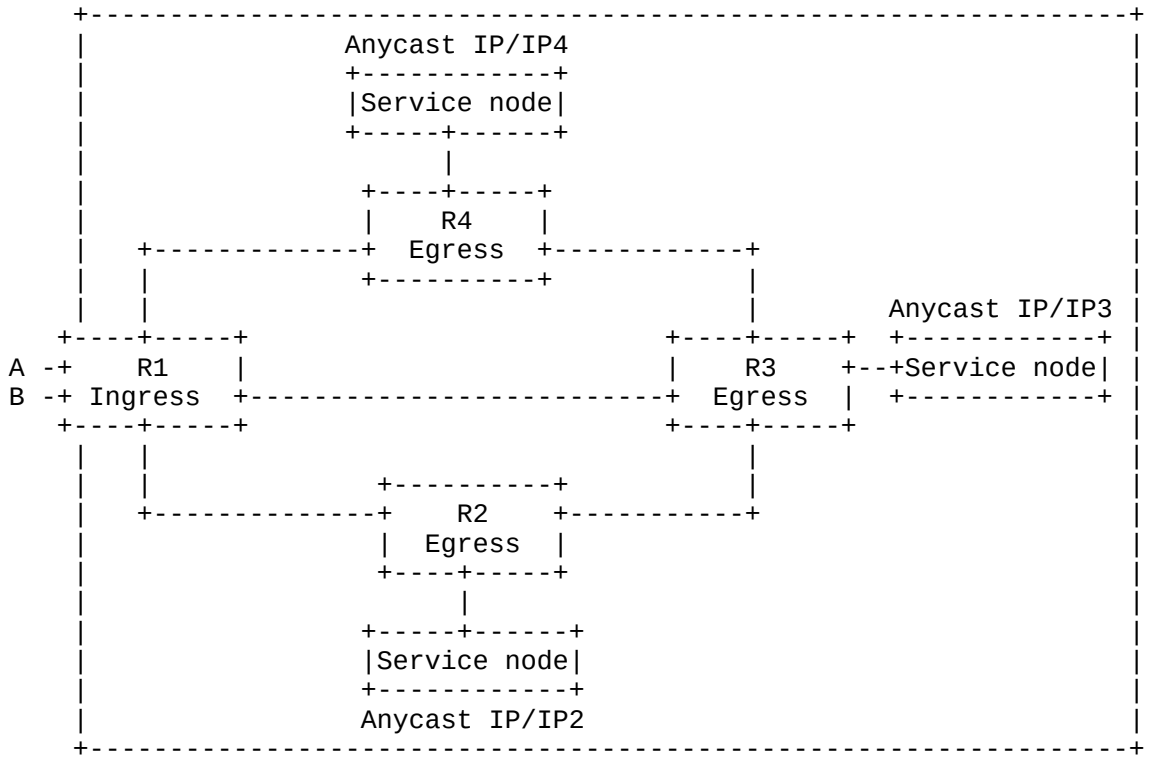



Figure 2: The Computing-Aware Traffic Steering (CATS) scenario

Customer A and customer B want to access the same service. For customer A, the packet will firstly be transmitted to the corresponding anycast IP address. The ingress will determine the optimal service node for customer A based on the access cost, computing resources of each service node, and the scheduled computing resource scheduling algorithm. Similar processing will be performed when customer B accesses the same service.

When customer A accesses to the service, it presents the following steps:

* 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.

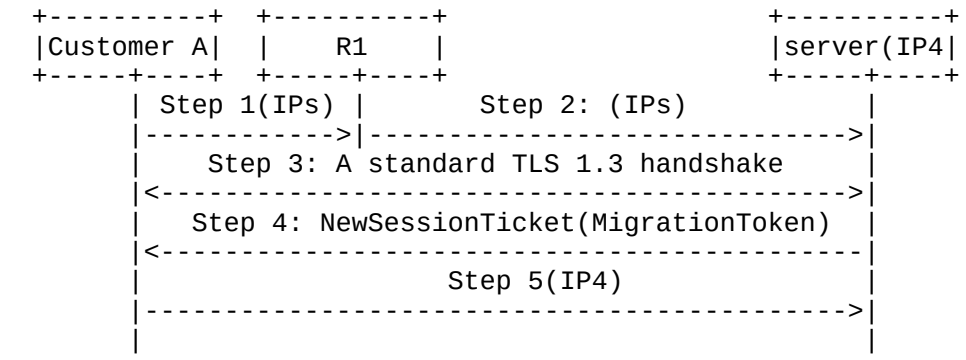



Figure 3: Procedures for the service affinity solution

In the whole process, devices in the network only need to broadcast the information of the computing network <Anycast IP Address, Service node Status>, and perform optimized scheduling of computing network resources according to this information.

Comparing to the existing solutions such as maintaining the customer-based connection status table in network devices, HTTP redirection and DNS redirection, this solution can avoid the waste of resources caused by saving a large amount of customer status data in the network devices, and realize the optimized scheduling of resources based on network conditions and computing resources in the computing-aware traffic steering scenario, so as to realize the reasonable operation of network resources, cloud resources and computing resources.

## 7. 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.

## 8. IANA Considerations

This document requires IANA to allocate new codepoints from the following TLS registries, as defined in [RFC8446]:

1. From the "TLS ExtensionType Values" registry for ‘migration_support‘ and ‘migration_token‘. This document suggests the values TBD1 and TBD2.

2. From the "TLS Alert Registry" for the ‘migrate_notify‘ alert. This document suggests the value TBD3.

## 9. Normative References

[I-D.ietf-cats-usecases-requirements]
Yao, K., Contreras, L. M., Shi, H., Zhang, S., and Q. An, "Computing-Aware Traffic Steering (CATS) Problem Statement, Use Cases, and Requirements", Work in Progress, Internet-Draft, draft-ietf-cats-usecases-requirements-14, 2 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cats-usecases-requirements-14>.

[I-D.li-cats-attack-detection]
Zhou, H., Wang, W., and S. Deng, "Computing-aware Traffic Steering for attack detection", Work in Progress, Internet-Draft, draft-li-cats-attack-detection-01, 8 April 2024, <https://datatracker.ietf.org/doc/html/draft-li-cats-attack-detection-01>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[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>.

## Authors' Addresses

Wei Wang
China Telecom
Beiqijia Town, Changping District
Beijing
Beijing, 102209
China
Email: weiwang94@foxmail.com

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing
Beijing, 102209
China
Email: wangaj3@chinatelecom.cn

Mohit Sahni
Palo Alto Networks
San Francisco
Email: msahni@paloaltonetworks.com

Ketul Sheth
Palo Alto Networks
San Francisco
Email: ksheth@paloaltonetworks.com