Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 15 June 2026

N. Montero
Independent Researcher

12 December 2025

Unified Transition Overlay (UTO): A Minimal Cross-Version Transport Mechanism for IPv4/IPv6
draft-montero-uto-00

Abstract

This document specifies Unified Transition Overlay (UTO), a minimal encapsulation mechanism designed to enable bidirectional communication between IPv4-only and IPv6-only hosts without requiring NAT64, DNS64, heavy protocol translation, or global changes to Internet routing. UTO introduces a compact 12-byte header that carries a Remote Native Address (RNA) and allows a packet to traverse a network whose forwarding plane uses the opposite IP version. The underlying network remains purely IPv4 or purely IPv6, ensuring compatibility with existing hardware and reducing operational complexity.

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 15 June 2026.

Copyright Notice

Table of Contents

1. Introduction

The transition from IPv4 to IPv6 has progressed far slower than originally expected. Although IPv6 was standardized in 1998 and provides an address space sufficient for global-scale expansion, IPv4 continues to dominate a significant portion of Internet traffic and infrastructure.

Existing coexistence mechanisms such as NAT64, 464XLAT, DS-Lite, MAP-T, MAP-E, and tunneling technologies allow partial interoperability between the two protocol families. However, these solutions introduce operational challenges including stateful translation, DNS rewriting, multi-stage encapsulation, complex customer-premises equipment requirements, and decreased visibility for operators and middleboxes.

Unified Transition Overlay (UTO) defines a minimal and efficient encapsulation mechanism allowing IPv4-only hosts to communicate with IPv6-only hosts, and vice versa, without altering the forwarding behavior of the underlying network. UTO preserves the original IP packet by wrapping it in a compact micro-header that carries the Remote Native Address (RNA) of the intended destination host.

UTO operates exclusively at transition gateways and does not require updates to backbone routers, end-host stacks, or the semantics of the IPv4/IPv6 version field. The mechanism is suitable for incremental deployment and does not disrupt existing routing architectures.

2. Motivation

The continued coexistence of IPv4 and IPv6 has created operational environments in which many hosts are limited to a single IP version. IPv4 depletion has pushed cloud providers and IoT systems toward IPv6-only deployments, while numerous enterprise networks still rely heavily on IPv4 due to legacy applications, security policies, or hardware constraints.

The industry has generally adopted translation-based models for cross-version communication. While functional, these models rely on complex transformation of packet headers, manipulation of DNS records, and maintenance of large state tables. Such approaches can reduce transparency, complicate troubleshooting, and introduce latency.

UTO addresses these issues by avoiding deep translation entirely. Instead, the original IP packet remains intact, and only a concise transition overlay header is added. The underlying network sees only IPv4 or IPv6 traffic, while the UTO-Gateway performs minimal processing at the boundaries between domains.

The goals of UTO include:

* Preserving native IPv4 and IPv6 semantics.

* Enabling communication between IPv4-only and IPv6-only hosts.

* Maintaining compatibility with existing hardware forwarding pipelines.

* Reducing operational complexity compared to NAT64 or dual-stack deployments.

* Supporting incremental deployment without global coordination.

3. Terminology

UTO-Gateway (UGW) A device that applies or removes the UTO header and performs encapsulation or decapsulation based on the direction of cross-version traffic.

Remote Native Address (RNA) The IPv4 or IPv6 address of the final destination host in its native format, carried inside the UTO header.

Underlying Network The forwarding plane over which the encapsulated packet travels. This network may be IPv4-only or IPv6-only, depending on deployment.

Native Packet The sender’s original IPv4 or IPv6 packet before encapsulation.

UTO Header A compact 12-byte transition header inserted before the native packet during encapsulation.

Endpoint A host that supports only one IP version (IPv4-only or IPv6-only).

4. Architecture Overview

UTO enables cross-version communication by inserting a compact overlay header that carries the Remote Native Address (RNA) of the destination host. The underlying network transports the encapsulated packet using its native forwarding plane (IPv4 or IPv6), while the UTO-Gateway (UGW) at the remote boundary restores the original packet and delivers it to the destination.

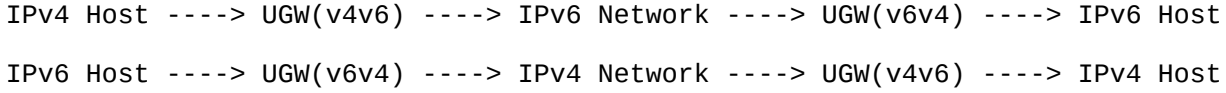

Figure 1: UTO Architecture

In all cases, the underlying forwarding devices (routers, middleboxes, fabric switches, and hardware ASIC pipelines) see only pure IPv4 or pure IPv6 packets. UTO does not modify the meaning of any field in either IP header, does not introduce mixed-version headers, and does not require new behaviors from the routing subsystem.

Encapsulation occurs only at the edges of transition domains, where UGWs have explicit configuration defining which prefixes or hosts reside in opposite-version networks. Traffic between same-version hosts (IPv4-IPv4 or IPv6-IPv6) remains native and is unaffected by UTO.

5. UTO Header Format

The UTO header precedes the native packet and has a fixed size of 12 bytes. It contains the Remote Native Address (RNA) of the destination host, encoded according to the address family and compressed when necessary. The general layout is:

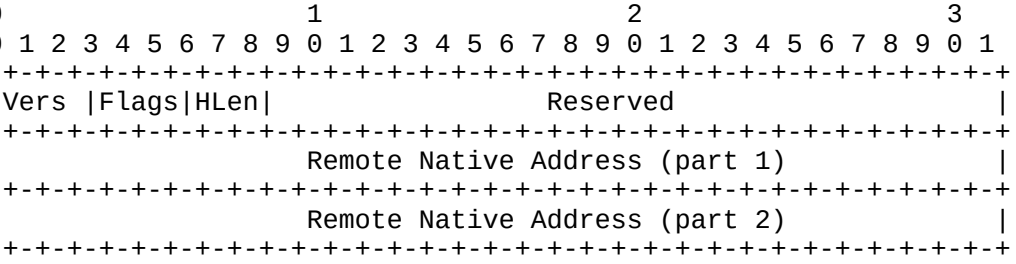

Figure 2: UTO Header Layout

The UTO header is inserted immediately after the outer IP header (IPv4 or IPv6), depending on the direction of encapsulation. When decapsulating, the receiving UGW removes the UTO header, restores the native packet, and forwards it according to its normal routing table.

5.1. Field Definitions

Vers (4 bits): Indicates the version of the UTO protocol. This specification defines Vers=1 (0001).

Flags (4 bits): Identify the address family encoded in RNA. 0001 - RNA contains an IPv4 address. 0010 - RNA contains an IPv6 address (compressed).

HLen (8 bits): Total length of the UTO header in bytes. For this version, HLen MUST be set to 12.

Reserved (16 bits): Reserved for future use. MUST be set to zero on transmission and ignored on receipt.

Remote Native Address (RNA): Carries the destination host's native IPv4 or IPv6 address. When the address is IPv6, the RNA block contains a compressed representation sufficient to reconstruct the full 128-bit address at the receiving UGW.

5.2. Encoding of IPv4 Remote Native Address

When the Flags field indicates an IPv4 address (0001), the RNA field carries the 32-bit IPv4 address left-aligned within the 64-bit RNA space. The remaining bits MUST be set to zero.

Example:

IPv4 address: 192.0.2.44
RNA encoding: 0xC000022C 00000000

Figure 3: IPv4 RNA Example

5.3. Encoding of IPv6 Remote Native Address

When the Flags field indicates IPv6 (0010), the RNA carries a compressed representation of the full 128-bit IPv6 address. Compression is deterministic and reversible.

UTO uses a two-component encoding scheme:

* A local 16-bit Label that identifies a known IPv6 prefix.

* A 64-bit Interface Identifier (IID) copied directly from the native IPv6 address.

Each UGW maintains an internal Prefix Table:

Label (16 bits) IPv6 Prefix (48 bits)

Figure 4: Prefix Table

During encapsulation, the UGW:

1. Splits the native IPv6 address into: Prefix (48 bits) + Subnet-ID (16 bits) + IID (64 bits).
2. Replaces the Prefix+Subnet-ID (64 bits) with a single 16-bit Label.
3. Keeps the original IID intact.
4. Inserts: [Label | IID] into the RNA field of the UTO header.

During decapsulation, the receiving UGW:

1. Reads the Label.
2. Recovers the 48-bit Prefix from its Prefix Table.
3. Reconstructs the IPv6 address as: Prefix (48 bits) + Subnet-ID (16 bits from Label context) + IID (64 bits).

Compression reduces RNA size while preserving uniqueness and allowing fast O(1) reconstruction.

6. Protocol Operation

This section describes the behavior of UTO-Gateways (UGWs) during encapsulation and decapsulation, and defines the forwarding logic for IPv4-to-IPv6 and IPv6-to-IPv4 transitions. UTO operates strictly at administrative boundaries and does not modify packets traveling within a single address family.

Cross-version traffic is identified by UGW policy, typically based on destination prefixes, routing tables, interface roles, or explicitly configured host mappings.

6.1. IPv4-to-IPv6 Flow

When an IPv4-only host sends a packet toward an IPv6-only host, the following steps occur:

1. The IPv4 host transmits a normal IPv4 packet.
2. The outbound UGW checks its policy and identifies the destination prefix as belonging to an IPv6 domain.
3. The UGW constructs a UTO header with Flags=0010 (IPv6 RNA) and encodes the destination IPv6 address.
4. The UGW encapsulates the native IPv4 packet inside an IPv6 outer header. The RNA in the UTO header allows the remote UGW to restore the intended destination address.
5. The underlying IPv6 network forwards the packet normally, with no knowledge of UTO.
6. At the receiving boundary, the remote UGW removes the UTO header, reconstructs the full IPv6 address, and delivers a native IPv6 packet to the IPv6-only host.

6.2. IPv6-to-IPv4 Flow

When an IPv6-only host sends a packet toward an IPv4-only host:

1. The IPv6 host generates a normal IPv6 packet.
2. The outbound UGW identifies that the destination belongs to an IPv4 domain.

3. The UGW constructs a UTO header with Flags=0001 (IPv4 RNA) and inserts the 32-bit IPv4 address.
4. The UGW encapsulates the native IPv6 packet inside an IPv4 outer header.
5. The underlying IPv4 network forwards the encapsulated packet without inspecting the UTO header.
6. The remote UGW decapsulates the packet, extracts the IPv4 RNA, and forwards a native IPv4 packet to the destination.

6.3. Behavior of UTO-Gateways (UGW)

UGWs MUST implement the following logic:

* Maintain policy tables for identifying opposite-version prefixes or host mappings.

* Apply encapsulation only when necessary; native same-version traffic MUST NOT be altered.

* Validate UTO header integrity and RNA encoding upon receipt.

* Reconstruct full IPv6 destinations using prefix tables associated with their domain.

* Enforce security policies that restrict which prefixes may originate cross-version traffic.

* Drop malformed UTO headers or reconstruction failures.

UGWs MAY implement rate limiting or filtering on cross-version flows, depending on administrative policy.

UGWs MUST NOT modify the inner native IP header except when decrementing the Hop Limit or TTL, as required by normal forwarding semantics.

Performance Considerations: UTO is designed for efficient forwarding in both software and hardware implementations. Encapsulation and decapsulation are stateless operations requiring only header insertion/removal and optional prefix-table lookup. This avoids the per-flow state, checksum rewrites, and multi-field translations required by stateful transition mechanisms such as NAT64. Implementations MAY leverage hardware offload where available.

7. Deployment Scenarios

UTO is designed for deployment in environments where one IP version dominates the core, while hosts or services using the opposite version exist at the edges. This section outlines representative deployment patterns.

7.1. Enterprise with Legacy IPv4 Core

Many organizations operate IPv4-centric networks due to legacy equipment or applications. Cloud services, IoT systems, and external APIs may be IPv6-only. UTO allows such enterprises to:

* Maintain an IPv4-only core.
* Enable IPv4 hosts to reach IPv6-only resources.
* Avoid NAT64 or DNS64 deployment.
* Preserve end-to-end semantics for IPv6 destinations.

UGWs are deployed at the enterprise perimeter, typically at WAN routers or firewalls.

7.2. ISP with IPv6-Only Access Network

ISPs deploying IPv6-only access (e.g., mobile or FTTH networks) may still need to support IPv4-only services hosted externally. UTO enables:

* Native IPv6 access network operation.
* Cross-version communication without per-flow state.
* Reduction of CGNAT pressure.
* Clean separation between access IPv6 and external IPv4 routes.

UGWs are deployed at the ISP’s edge to minimize translation complexity.

7.3. IoT IPv6 Devices Reaching IPv4 Infrastructure

IoT deployments commonly use IPv6 due to address abundance and neighbor-discovery efficiency. However, control servers or legacy monitoring systems may operate only in IPv4. UTO allows IPv6-only IoT devices to:

* Reach IPv4 management systems without NAT-based hacks.
* Maintain native IPv6 addressing internally.
* Operate over an IPv4-only backhaul if required.

UGWs in this scenario are placed at aggregation points or border concentrators.

8. Advantages and Limitations

8.1. Advantages

UTO offers the following benefits:

* Minimal encapsulation overhead (12-byte header).
* No modification to IPv4 or IPv6 protocol semantics.
* No need for NAT64, DNS64, or stateful per-flow translation.
* Compatible with existing IPv4-only and IPv6-only hosts.
* Works with hardware forwarding pipelines already deployed.
* No impact on middleboxes that operate on the outer IP header.
* Easily deployable at administrative boundaries.
* Reduces operational and troubleshooting complexity.
* Supports incremental and partial deployments.

8.2. Limitations

UTO introduces certain limitations:

* Requires UTO-Gateways at transition points.
* Does not eliminate the need for proper routing policies.
* Requires prefix mapping tables for IPv6 RNA reconstruction.
* Encapsulated traffic may have a slightly larger MTU footprint.
* Cannot function as a global replacement for dual-stack architectures.

UTO increases packet size by 12 bytes in addition to the outer IP header. UGWs MUST implement Path MTU Discovery (PMTUD) on the encapsulation path, or alternatively support IP-layer fragmentation when required. Operators SHOULD provision appropriate MTU values on boundary interfaces to avoid encapsulation-induced loss.

9. Security Considerations

UTO does not alter IPv4 or IPv6 security properties. However, several operational considerations apply:

* UGWs MUST validate UTO header integrity before decapsulation.
* RNA values MUST be verified against authorized prefixes.
* Administrators SHOULD restrict which hosts can initiate cross-version traffic.
* Administrators SHOULD apply ingress filtering on UGW interfaces to prevent spoofing.
* UTO traffic MAY be protected using IPsec, MACsec, or tunnel-mode encryption if confidentiality is required.
* Misconfigured prefix tables could result in incorrect RNA reconstruction; implementations SHOULD detect inconsistencies.

Since UTO encapsulates entire packets, its security posture is similar to other tunneling protocols. Operators SHOULD ensure that monitoring systems can observe both outer headers and encapsulated flows when necessary.

10. IANA Considerations

This document requests that IANA allocate:

* A new IP Protocol Number for use as the UTO protocol header.
* A new IPv6 Next Header value identifying UTO encapsulation.

These values are required for correct identification of UTO packets when conveyed over native IPv4 or IPv6 networks.

11. Backward Compatibility

UTO is compatible with existing IPv4 and IPv6 networks because:

* UGWs handle all encapsulation and decapsulation.

* End hosts remain unchanged.

* Routers forward only native IPv4 or IPv6 outer headers.

* Middleboxes operate normally, inspecting only outer layers.

No modifications to TCP, UDP, ICMP, IPv4, or IPv6 behavior are required. UTO can coexist with other transition technologies, including NAT64, DS-Lite, MAP-E, MAP-T, and traditional tunnels.

12. Examples

12.1. Complete IPv4-to-IPv6 Packet Flow

The following example illustrates a full cross-version transition:

Source IPv4 host: 10.1.1.10

Destination IPv6: 2001:db8:20::44

Steps:

1. The IPv4 host sends: IPv4 Header, TCP/UDP/ICMP Payload.
2. The outbound UGW constructs a UTO header with Flags=0010 and encodes the IPv6 address using the prefix and IID reconstruction method.
3. The UGW encapsulates the native IPv4 packet inside an IPv6 outer header: IPv6 Outer Header, UTO Header, IPv4 Native Packet.
4. The IPv6 backbone forwards the packet.
5. The remote UGW decapsulates, reconstructs the IPv6 address, and forwards a native IPv6 packet to the destination.

12.2. Complete IPv6-to-IPv4 Packet Flow

Example:

Source IPv6 host: 2001:db8:10::50

Destination IPv4: 192.0.2.99

Steps:

1. The IPv6 host sends a native IPv6 packet.
2. The outbound UGW inserts a UTO header with Flags=0001 and encodes the 32-bit IPv4 address (RNA field).
3. Encapsulation produces: IPv4 Outer Header, UTO Header, IPv6 Native Packet.
4. The IPv4 network forwards the packet normally.

5. The receiving UGW extracts the IPv4 RNA and delivers a native IPv4 packet to the destination.

13. Normative References

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

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

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6", RFC 2473, December 1998, <https://www.rfc-editor.org/rfc/rfc2473>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000, <https://www.rfc-editor.org/rfc/rfc2784>.

14. Informative References

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64", RFC 6146, April 2011, <https://www.rfc-editor.org/rfc/rfc6146>.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, April 2013, <https://www.rfc-editor.org/rfc/rfc6877>.

[RFC7599] Li, X., Bao, C., and F. Baker, "Mapping of Address and Port using Translation (MAP-T)", RFC 7599, July 2015, <https://www.rfc-editor.org/rfc/rfc7599>.

[RFC8925] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "LISP Architecture", RFC 8925, October 2020, <https://www.rfc-editor.org/rfc/rfc8925>.

[RFC8986] Filsfils, C., Leddy, J., and D. Voyer, "Segment Routing over IPv6 (SRv6)", RFC 8986, February 2021, <https://www.rfc-editor.org/rfc/rfc8986>.

Appendix A. Acknowledgments

The author would like to thank the members of the operational and research communities whose discussions on transition technologies motivated the development of this document. Their practical feedback on coexistence mechanisms, routing architectures, and deployment challenges helped shape the UTO design goals.

Special appreciation is extended to engineers who contributed insights on encapsulation performance, IPv6 prefix management, and hardware forwarding behavior. Their operational experience provided valuable context for identifying the limitations of translation-based models.

Author’s Address

Nicolas Montero Torrealba
Independent Researcher
Santiago
Chile
Email: nicolas.montero@usach.cl