tsvwg
Internet-Draft
Intended status: Standards Track
Expires: 17 April 2026

A. Bisht
Cisco Meraki

14 October 2025

Congestion-Aware Multipath Tunnel Selection for Transport Services
draft-altanai-tsv-multipath-nested-tunnels-00

Abstract

This document addresses the transport-layer challenges of path selection in environments with multiple available tunneling options and congestion control mechanisms. It identifies congestion control conflicts that arise from nested tunneling protocols and proposes a congestion-aware multipath tunnel selection algorithm that conforms to the guidelines established in [RFC9599] for adding congestion notification to protocols that encapsulate IP. The proposed approach considers Explicit Congestion Notification (ECN) propagation, transport protocol characteristics, and network conditions to optimize path selection while avoiding multilevel congestion control issues. This work aligns with current Transport and Services Working Group efforts on Non-Queue-Building (NQB) behaviors, careful congestion control resume, and multipath transport protocols.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://altanai.github.io/tunnel-path-selection/draft-altanai-tsv-multipath-nested-tunnels.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-altanai-tsv-multipath-nested-tunnels/.

Discussion of this document takes place on the tsvwg WG mailing list (mailto:tsvwg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tsvwg/. Subscribe at https://www.ietf.org/mailman/listinfo/tsvwg/.

Source for this draft and an issue tracker can be found at https://github.com/altanai/tunnel-path-selection.

Status of This Memo

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

This Internet-Draft will expire on 17 April 2026.

Copyright Notice

Table of Contents

1. Introduction

A path for sending data across a network can consist of a combination of many factors such as uplink, application layer protocol, tunneling protocol among others. For example TLS operates above the network layer, SSH and Secure Real-time Transport Protocol (SRTP) [RFC5764] operates at the application layer to carry data. Stream Control Transmission Protocol (SCTP), intended to tunnel signaling messages over IP networks, can encapsulate data as well. Tunneling can build a secure interconnection called Virtual Private Network(VPN) that provides a private subnet to pass traffic between the tunneled endpoints. This setup caters to enterprises and small private home networks alike. The protocols for such network level tunneling may include GRE, L2P, IPSec, WireGuard, OpenVPN and even proprietary protocols such as AutoVPN or custom implementtion over DTLS. Now multiple proxied stream- and datagram-based flows are possible inside an HTTP connection through the MASQUE which is build on QUIC. However there may be real world use-cases where network tunnels could nest application tunnels, which leads to large overheads in latency and quality of data.

1.1. Fundamental Differences: Tunneled vs Non-Tunneled Traffic Path Selection

Path selection for tunneled traffic operates under fundamentally different constraints and objectives compared to open internet traffic:

Open internet traffic path selection is primarily controlled by either Applications (CDN selection, server choice) or Client-side networking stacks (Happy Eyeballs, etc.). To some extend user preferences and browser settings also play a role. Over this the Internet Service Providers may further controlouting policies and peering agreements and add their propertiary Traffic engineering algorithms.

Tunneled traffic path selection is primarily governed by enterprise or service provider security policies. These policies determine which traffic must be tunneled for compliance or security reasons along with geographic or jurisdictional routing constraints. The data sovereignty and privacy protection requirements apply strongly to secure networks service providers. For this reason Network service providers (NSPs) maintain strict control over tunneled traffic routing through: - Dedicated tunnel infrastructure with specific performance guarantees - Policy-based routing that may override optimal path selection for security - Service Level Agreement (SLA) enforcement mechanisms - Centralized management - predtermined ingress and egress points Unlike open internet traffic where applications can freely choose destinations, tunneled traffic has limited Endpoint Flexibility.

1.2. Current Path Selection Challenges

Today’s heterogeneous networking landscape presents critical path selection challenges that span across multi-cloud environments, SASE architectures, and ISP networks, where vendors employ fundamentally incompatible approaches for routing incoming traffic flows. Multi-cloud deployments (AWS, Azure, Google Cloud so on) suffer from inconsistent path selection decisions as traffic traverses between theirs and other’s cloud providers, each implementing proprietary routing algorithms that optimize for local rather than global performance. SASE architectures compound these challenges by attenmpting to integrate networking and security functions across multiple vendor solutions, yet current path selection mechanisms fail to coordinate effectively between edge security appliances, SD-WAN controllers, and cloud security services. Meanwhile, ISPs and network service providers continue to rely on fragmented approaches—ranging from load balancing across multiple paths to sophisticated ML-based traffic classification and queing—but these vendor-specific implementations lack standardization and interoperability, creating significant operational complexity and suboptimal performance when traffic flows cross vendor boundaries in modern hybrid network deployments.

The key challenge is developing a standardized algorithm that can operate within the constraints of tunneled traffic while providing optimal performance, unlike open internet traffic where applications and ISPs have more flexibility in path selection decisions.

1.2.1. Vendor Interoperability Challenges

When different vendors implement proprietary or incompatible path selection algorithms, several critical disadvantages emerge that significantly impact network operations and performance:

1.2.1.1. 1. Inconsistent Traffic Behavior Across Vendor Boundaries

Different vendor implementations may make contradictory path selection decisions for the same traffic flow, leading to conflicts and worst Suboptimal Routing where traffic may take longer, more expensive, or less reliable paths when crossing vendor boundaries. This leads to performance Oscillation and inconsistencies across vendor implementations.

Example Scenario: A flow from Vendor A’s SD-WAN appliance to Vendor B’s tunnel endpoint may experience optimal path selection within Vendor A’s domain but suboptimal selection when entering Vendor B’s infrastructure due to incompatible metrics and decision criteria.

1.2.1.2. 2. Operational Complexity and Management Overhead

Network operators face significant challenges with multiple proprietary path selection implementation across multi-vendor deployments. The challenges not only include tracking but synchoirnizing any updates. Incompatible algorithms from different vendors lead to redundant Path Probing where Multiple vendors may independently probe the same paths, consuming bandwidth and generating unnecessary traffic, thus duplicate discovery processes. This makes troubleshoting tough ans also deuplicate configurations. This further leads to computational overhead leading to inconsistent resource utilization and cache inefficiency due to no decision sharing among vendors.

Network-Wide Suboptimization is another concern as Vendor algorithms optimizing for local criteria may create network-wide inefficiencies and Load Balancing Imbalances leading to uneven traffic distribution and congestion hotspots. Unpredictable path selection behavior makes accurate capacity planning difficult for everyone.

1.2.1.3. 3. Security and Compliance Risks

Policy Enforcement Gaps based on varied interpretion makes may inadvertently route traffic through non-compliant paths or jurisdictions Multiple proprietary implementations increase the overall attack surface as Vendor-specific path selection patterns may reveal sensitive network topology or traffic information.

Proprietary unstandardised best path selection algorithms create dependencies resulting in increased Total Cost of Ownership (TCO) through vendor lock-in. Service quality degradation occurs through inconsistent user experiences with varying service quality depending on which vendor’s equipment handles their traffic. Unpredictable path selection behavior also makes it difficult to guarantee service

level agreements and createing network continuity risks. Furthermore, this artificial differentiation fragments innovation as vendor-specific implementations prevent collaborative development of industry-wide improvements, create interoperability stagnation, and establish barriers to technology evolution.

1.2.2. The Need for Standardization

These disadvantages collectively demonstrate the critical importance of standardized path selection algorithms that can ensure consistent behavior by providing predictable path selection decisions across vendor boundaries, enable interoperability through seamless integration of multi-vendor network infrastructure, reduce operational complexity and enable global optimization rather than vendor-specific local optimization, and enhance security through consistent policy enforcement and threat response across all network elements.

1.2.2.1. Path discovery overhead and resulting conflicts

The absence of standardized path selection algorithms creates significant overhead in discovering optimal paths and leads to widespread fragmentation challenges across heterogeneous tunnel deployments. When networking protocols use hole punching to establish paths between endpoints and multiple tunnels are formed for primary-standby or load sharing configurations, each vendor implements proprietary Path MTU Discovery (PMTUD) mechanisms that operate independently and often conflict with each other. This lack of coordination results in redundant path probing, where multiple vendors simultaneously attempt to discover the same path characteristics, consuming valuable bandwidth and generating unnecessary control traffic. The fragmentation problem is particularly acute because nested tunnels greatly impact traffic quality, with each encapsulation layer introducing additional headers that reduce the effective payload size, leading to frequent fragmentation and defragmentation cycles that increase computational overhead especially for time-sensitive applications operating under limited bandwidth constraints.

While [RFC9868] Transport Options for UDP provides mechanisms for enhanced path characteristic discovery through embedded metrics, congestion state signaling, and capability advertisement within regular data packets, UDP options-based signaling falls fundamentally short in addressing the core challenges of optimal path selection in multi-vendor environments. The UDP options approach suffers from limited cross-vendor coordination where different implementations may interpret or prioritize the embedded path quality indicators differently. Furthermore, the DPLPMTUD implementation through UDP

options designed to minimize fragmentation-related overhead becomes ineffective when vendors use incompatible fragmentation strategies or fail to coordinate MTU discovery across nested tunnel layers, resulting in suboptimal path selections that may actually increase rather than decrease fragmentation.

The inadequacy becomes evident in complex multi-vendor scenarios where algorithm negotiation, ECN propagation and conflict detection mechanisms fail to provide the comprehensive coordination needed for truly globally optimal path selection. Even with sophisticated features like authenticated ECN marking propagation through nested tunnel layers and active identification of congestion control interference, the fundamental problem remains that these mechanisms operate in isolation without a standardized framework for making unified path selection decisions. This fragmentation of approaches leads to situations where UDP options may indicate one path as optimal based on embedded metrics, while vendor-specific algorithms select different paths based on proprietary criteria, resulting in inconsistent path selection behavior** that undermines the very benefits that UDP options were designed to provide. The lack of a unified decision-making framework means that even advanced signaling capabilities cannot overcome the coordination challenges inherent in multi-vendor tunnel deployments, necessitating the standardized path selection algorithm proposed in this document.

1.2.3. Multilevel Congestion Control and ECN Propagation

Nested tunneling protocols create significant challenges for congestion control mechanisms, particularly when multiple layers independently attempt to manage network congestion. This section addresses these challenges in accordance with [RFC9599] guidelines that add congestion notification to protocols that encapsulate IP. When transport protocols are tunneled within other transport protocols, multiple congestion control loops can interfere with each other, creating a complex interaction where the inner transport congestion control (e.g., TCP or QUIC) implements its own congestion control based on observed packet loss and RTT measurements, while simultaneously the tunnel transport congestion control (e.g., QUIC for MASQUE, UDP for IPsec) may implement its own independent congestion control mechanisms. These interaction effects result in the tunnel’s congestion control affecting the RTT and loss measurements of the inner transport, leading to suboptimal performance and potential oscillations that degrade overall network efficiency.

During ECN encapsulation, tunnel endpoints must copy the ECN field from the inner header to the outer header as specified in [RFC6040] to negotiate ECN capability during tunnel establishment to enable

proper congestion notification propagation, but this process becomes increasingly complex when multiple ECMP tunnels or aggregated flows are involved, each potentially implementing different ECN handling strategies.

To address these multilevel congestion control challenges, the proposed congestion-aware path selection algorithm incorporates multiple sophisticated mechanisms that work synergistically to provide comprehensive network congestion awareness. The algorithm monitors ECN CE (Congestion Experienced) markings on different paths to assess real-time congestion levels, while it also identifies increased RTT variability that often indicates congestion for path quality assessment that inform path selection decisions, and where available, queue delay measurements offer early congestion detection capabilities that enable proactive path switching before performance degradation becomes severe.

This integrated approach to congestion management represents a significant advancement over current fragmented solutions, providing a standardized framework that can effectively coordinate congestion control across multiple tunnel layers while maintaining compatibility with existing [RFC9599] and [RFC6040] standards. By incorporating these multiple congestion indicators into a unified decision-making process, the algorithm can make intelligent path selection decisions that avoid the oscillations and performance degradation inherent in current multilevel congestion control implementations.

1.2.4. Prioritization

Mainstream techniques such as packet marking(DSCP, ECN so on) and queuing of other non-critical traffic (Fq-CODEL, CAKE AQM) to optimize for realtime streams is essentially prioritization in practice. However, VPN providers, CSPs and/or ISP may employ polar-opposite algorithms to shape traffic based on their interest which could lead to an overall non-synchronized approach, where a stream is prioritized in some networks and deprioritized in other networks.

1.3. Limited scope of past proposals for prioritiztion or path selection

Some prior work presented to IETF with the inevitable need for traffic shaping and prioritiation may include one or more of the following

At present, in the case of multiple active uplinks connecting to various ISPs, there are multiple techniques to steer or prioritize traffic across the network (see [I-D.ietf-intarea-tunnels]), which may include,

1.3.1. 1: Full or Split tunnel based on Diff Serv via Differentiated Services Code Point (DSCP)

[RFC3270] defines how to support the Diffserv architecture in MPLS networks, including how to encode DSCP in an MPLS header. Application priorities even though using the same protocol have also been used to mark the packets differently such as DSCP Packet Markings for WebRTC QoS [RFC8807]. An example of path selection based on prioritiztion is that in case of dual uplinks available hosting an active tunnel tunnel each, use the one more with better performance for RTP since that is more prirotized over FTP data.

The DSCP-based approach offers the advantage of being widely adopted across network infrastructures, making it a familiar and standardized method for traffic differentiation. However, this approach has significant limitations including unreliability in some cases where network operators may choose not to honor markings, and the inherent coarse-grained classification that cannot adequately differentiate between nuanced application requirements.

1.3.2. 2: Multiple Active VPN Uplinks used in weighted round robin order or ECMP

In case of multiple Active VPN Uplinks available, multiple paths are available to a destination which can be split tunneled, tunneled via different application or network layer or both such as nested tunneled. With so many options generic traffic shaping rules are often applied which may be based on QoS such as MOS, loss, latency, jitter, usage history, throughput on all VPN sessions or any other customized score. Attributes such as app type, address or even client identifier such as mac address can also be used to balance load across available options. Simple loop techniques such as Weighted Round Robin do not offer much granularity and can also result in misconfiguration or worse still creating a bottleneck. Other means of balancing the traffic across available paths are Equal-cost multi-path (ECMP) [RFC2992] which is fair by design and routes packets along multiple paths of equal cost but suffers from limited adaptability especially in sudden changes of network conditions and heterogeneous environments of unequal costs. Although it is tough to make a path selection algorithm which is ideal for balance, scalability as well optimized for all kinds of resource utilization, not having a common basis for path selection can not only lead to detrimental user experience but also undue strain on the network.

1.3.3. 3. Policy-Based routing that use flow preferences to pin traffic to a particular path

It is common for device or network policy to manage network flows such as bandwidth allocation or rate limiting, Geo or proximity based rules. At the device level these policies may prioritize some packets over others to avoid queing delay. Modern hybrid deployments employ many uplinks with a varity of traffic shaping policies which can be adjusted dynmaically not only based on Qos but also on hop-by-hop insights from network, tracking uplink’s utilization, uptime, failure or outages.

Policy-based routing provides the benefit of simplicity in implementation and configuration, making it straightforward for network administrators to define and manage traffic flows. However, this approach suffers from poor scalability as the number of policies and network complexity grows, often requiring manual intervention and becoming unwieldy in large-scale deployments.

1.3.4. 4. Dynamic Path Selection with application or domain identification

The aplication knows its type and can directly feed the information to the algorithm. If the sender is not aware of the application it can attempt to obtain this information from intelligent ML models as Network Based Application Recognition (NBAR) from Cisco. Models exist that can suggest bottlenecks for a traffic type on a path by analysising patterns. Dynamic path selection can even rely on explicitly identifying Provisioning Domain Names through a Router Advertisement (RA) option. Discovering Provisioning Domain Names and Data, its architecture involving the authenticatio and trust model has been decribed in prior work [RFC7556] and [RFC8801].

1.3.5. 5. MASQUE (QUIC multiplexing) for all Web trafic

MASQUE provides the advantage of being able to handle both reliable and unreliable data streams efficiently through QUIC multiplexing, offering flexibility in transport protocol selection. However, this approach has the limitation of not being well-suited for non-web-based traffic, potentially requiring additional adaptations or alternative solutions for enterprise applications that do not follow web protocols.

1.3.6. 6. Whitelist for IP address or tuples to prioritize

Using whitelists for IP addresses or tuples offers the advantage of simplicity in implementation and clear, understandable traffic prioritization rules. However, this approach suffers from poor scalability as networks grow and IP address ranges change, requiring constant maintenance and becoming impractical for large-scale dynamic environments.

1.3.7. 7. Entropy headers

Entropy headers are extension to traditional packet header that include information about the randomness of the packet’s payload. These help distributing traffic more evenly in a multipath network, mitigating the risk of hotspots and potential congestion points.

Entropy headers provide the advantage of being protected from in-path modification by making these headers non-updatable, ensuring the integrity of load balancing decisions. However, this approach raises privacy concerns as the entropy information may reveal patterns about the payload content or application behavior that could be exploited by network observers.

1.3.8. 8. Tunnelling of Explicit Congestion Notification(ECN)

Addition of ECN to IP [RFC3168] paved the way for much optimization in managing queues based on these marking. [RFC6040] describes the problems related to obscured original ECN markings in tunneled traffic. It proposes a standard for tunnels to propagate an extra level of congestion severity.

ECN tunneling benefits from having existing standards that provide a foundation for implementation and interoperability across different network equipment vendors. However, the approach becomes significantly more complicated when dealing with nested tunnels, where multiple layers of ECN marking can create conflicts and require complex processing to maintain proper congestion signaling.

1.3.9. 9. Flow labelling or classification for traffic steering

Flow labeling and classification approaches provide the significant advantage of enabling the application of artificially intelligent machine learning models for sophisticated traffic analysis and optimization, allowing for adaptive and predictive traffic management. However, this approach can compromise privacy by potentially exposing sensitive information about user behavior, application usage patterns, and business processes through the classification metadata.

2. Proposal to standardize the selection algorithm

The VPN can be considered a limited premium network that protects confidential information of an organization such as business communication between retail stores. Hybrid work and move towards private access has increased the interest in tunneling traffic between endpoints. However at present, the traffic steering decision is made in a limited scoped or rule based manner which is different for various networks and service providers. Instead an alternative dynamic strategy is proposed which gauges the confidence in the various available options dynamically and may choose to send data directly via edge gateway, use one or more of the available tunnels or create a new on-demand tunnel, leveraging any of the tunneling protocols best suited.

By dynamically deciding the tunnel type for a stream or packet, we could avoid the non-performing or counter-productive use-cases such as * added latency on real time streaming * added encryption for already end-to-end encrypted VoIP calls * NAT traversal nightmare * nested tunneling and double congestion control * exhausting limited bandwidth available from VPN providers

The proposal is to standardize an algorithm that computes multiple available options and decides whether, on-demand tunnels are created (via MASQUE, IPSec, SSH, GRE other proprietary protocols such as AutoVPN), an existing set of tunnels be reused or any other route, based on the current network dynamics and vulnerability of the traffic. Standardized Path selection decision making algorithm would ensure same treatment of the stream across heterogeneous networks.

2.1. Algorithm Input Parameters

The congestion-aware multipath tunnel selection algorithm processes multiple input parameters organized into the following categories. These parameters are selected to ensure vendor-agnostic standardization for inter-cloud and inter-vendor tunneling scenarios:

2.1.1. Transport Layer Metrics

1. *Available Bandwidth*: Measured in bits per second, obtained through bandwidth estimation algorithms or derived from Bandwidth Delay Product (BDP) calculations. The algorithm uses techniques similar to those in QUIC congestion control [RFC9002].

2. *Round-Trip Time (RTT)*: Both baseline RTT and RTT variation measurements, which indicate path latency characteristics and potential congestion.

3. *Packet Loss Rate*: Measured loss percentage that indicates network congestion or path quality issues.

4. *ECN Markings Ratio*: The percentage of packets marked with ECN Congestion Experienced (CE) bits, providing early congestion detection as per [RFC9599].

5. *UDP Options Enhanced Metrics*: Leveraging [RFC9868] Transport Options for UDP to enable real-time path characteristic discovery:

    * Real-time Path Probing: UDP options carrying path quality metrics for immediate assessment without dedicated probe traffic

    * Congestion Control Compatibility Signaling: Transport options indicating supported congestion control algorithms (BBR, CUBIC, etc.) and ECN capabilities

    * Authentication Status: AUTH option validation confirming tunnel endpoint authenticity and preventing path manipulation attacks

2.1.2. Path Characteristics and Historical Performance

1. *Enhanced Tunnel Overhead Calculations*: Leveraging [RFC9868] UDP options for precise overhead assessment:

    * Base Protocol Overhead: Static overhead from tunneling protocols (IPsec, MASQUE, WireGuard, etc.)

    * Dynamic UDP Options Overhead: Additional bytes from [RFC9868] transport options based on active feature set

    * Fragmentation Impact Assessment: Real-time evaluation of fragmentation probability using UDP options for Path MTU Discovery (DPLPMTUD)

    * Processing Overhead Metrics: CPU and memory costs for UDP option parsing and generation at tunnel endpoints

2. *Path MTU Discovery Enhanced*: [RFC9868] DPLPMTUD support enabling:

    * Adaptive MTU Sizing: Dynamic adjustment based on observed fragmentation

* Multi-layer MTU Coordination: Coordinated MTU discovery across nested tunnel layers

* Fragmentation Avoidance: Proactive path selection to minimize fragmentation overhead

3. *Advanced Congestion Control Compatibility Assessment*: [RFC9868] enhanced evaluation including:

* Transport Option Negotiation: Capability discovery for congestion control algorithms through UDP options

* ECN Propagation Verification: Active testing of ECN handling across tunnel layers using authenticated transport options

* Congestion Control Interference Detection: Identification of nested congestion control conflicts through option-based signaling

* Algorithm Compatibility Matrix: Real-time assessment of congestion control algorithm effectiveness on each path

4. *Historical Performance Database*: Time-series data including:

* Average throughput over different time periods (hourly, daily, weekly)

* Failure frequency and duration statistics

* Congestion pattern analysis

* Peak usage periods and capacity utilization trends

* UDP Options Performance History: [RFC9868] specific metrics including option parsing latency, authentication success rates, and DPLPMTUD effectiveness

5. *Predictive Performance Indicators*: - Projected path quality based on historical patterns - Anticipated congestion windows based on time-of-day patterns - Failure probability predictions based on infrastructure health - Expected performance degradation under various load conditions - Real-time Path Quality Prediction: ML models path characteristic forecasting

2.1.3. Network State Information

1. *Provisioning Domains (PvD)*: Network characteristics and policies as defined in [RFC7556] and [RFC8801].

2. *Network Telemetry*: Real-time network health data including queue depths, interface utilization, and error rates.

3. *NQB Traffic Classification*: Identification of Non-Queue-Building traffic that requires special handling as per TSVWG NQB PHB specifications.

4. *UDP Options Network Discovery*: [RFC9868] enhanced network state assessment:

   * Real-time Path Characteristic Discovery: Active measurement of path properties using UDP options without additional probe traffic

   * Transport Capability Negotiation: Dynamic discovery of network and endpoint transport feature support through option exchange

   * Security Policy Enforcement: Authenticated path validation using AUTH options to prevent path spoofing and ensure policy compliance

2.1.4. Operational Requirements and Constraints

1. *Failure Tolerance Specification*: Configurable parameters defining acceptable service degradation:

   * Maximum acceptable RTT increase which is a percentage or absolute value

   * Minimum acceptable bandwidth threshold

   * Maximum tolerable packet loss rate

   * Service availability requirements e.g., 99.9% uptime

   * Graceful degradation preferences vs hard failover

2. *Cost Considerations*: Economic factors affecting path selection:

   * Per-byte or per-minute tunnel service charges

   * Bandwidth cost differentials between providers

   * Infrastructure maintenance and operational costs

   * Service Level Agreement penalty costs

   * Peak usage surcharge implications

3. *Green Networking Parameters*: Environmental impact considerations:

   * Carbon intensity of different network paths (grams CO2/kWh)

   * Energy efficiency ratings of tunnel endpoints

   * Renewable energy percentage of infrastructure providers

   * Geographic routing preferences for low-carbon paths

   * Time-of-day energy grid composition data

4. *Prioritization Matrix*: Traffic classification and handling preferences:

   * Real-time traffic prioritization (VoIP, video conferencing)

   * Business-critical application identification

   * Time-sensitive transaction requirements

   * Bulk transfer tolerance levels

   * Interactive vs batch workload classification

2.1.5. Security and Policy Metrics

1. *Traffic Vulnerability Assessment*: A normalized value (0.0-1.0) indicating the security exposure of traffic, where higher values indicate greater need for tunneling protection.

2. *Policy Constraints*: Enterprise or regulatory policies that may mandate or prohibit certain tunneling approaches for specific traffic types including:

   * Data sovereignty requirements (geographic routing restrictions)

   * Compliance framework mandates (GDPR, HIPAA, SOX)

   * Cryptographic algorithm requirements

   * Audit trail and logging requirements

2.1.6. Path Evaluation Input Structure

1. *Available Path Inventory*: Comprehensive catalog of available paths including:

For example : yaml path_inventory: - path_id: "path_001" tunnel_type: "ipsec" endpoints: ["gateway1.provider.com", "gateway2.provider.com"] characteristics: bandwidth_capacity: "1000 Mbps" baseline_rtt: "25 ms" provider: "Provider A" geographic_route: ["US-West", "US-East"] sla_guarantees: uptime: "99.95%" max_latency: "50 ms" min_bandwidth: "800 Mbps" historical_performance: avg_throughput_30d: "850 Mbps" failure_incidents_30d: 2 avg_failure_duration: "4.2 minutes" peak_usage_hours: ["09:00-12:00", "14:00-17:00"] cost_structure: base_cost_per_gb: "$0.12" peak_surcharge: "25%" setup_cost: "$500" environmental_impact: carbon_intensity: "0.45 kg CO2/kWh" renewable_percentage: "75%"

1. *Incoming Traffic Profile*: Characteristics of traffic requiring path assignment:

For example : yaml traffic_profile: - flow_id: "flow_001" application_type: "video_conference" requirements: min_bandwidth: "5 Mbps" max_latency: "100 ms" max_jitter: "20 ms" priority: "high" failure_tolerance: "low" security_requirements: encryption_required: true compliance_frameworks: ["SOC2", "GDPR"] geographic_constraints: ["no_transit_through_china"] duration_estimate: "60 minutes" data_volume_estimate: "2.25 GB"

2.2. Technical Algorithm Details

The core algorithm performs a multi-dimensional optimization to match incoming traffic profiles with available paths while respecting policy constraints and performance objectives.

2.2.1. Algorithm Overview for a sample implementation

‘‘‘ Input: - Traffic_Profile (T) - Available_Paths (P = {p1, p2, ..., pn}) - Policy_Constraints (C) - Optimization_Weights (W)

Output: - Selected_Path (p*) - Confidence_Score (0.0-1.0) - Fallback_Paths (list of p_fallback1, p_fallback2, etc.)

Function: SelectOptimalTunnelPath(T, P, C, W) ‘‘‘

2.2.1.1. Step 1: Policy Filtering

Apply hard security and compliance constraints to eliminate paths that violate geographic, regulatory, or encryption requirements. Remove paths that cannot satisfy data sovereignty, compliance framework, or mandatory security policy requirements.

‘‘‘python def policy_filter(traffic_profile, available_paths, constraints): """ Filter paths based on hard security and compliance constraints """ eligible_paths = ..

```
for path in available_paths:
    # Check geographic constraints
    if not satisfies_geographic_constraints(path, constraints):
        continue

    # Check compliance requirements
    if not meets_compliance_requirements(path, traffic_profile.compliance):
        continue

    # Check encryption requirements
    if traffic_profile.encryption_required and not path.supports_encryption:
        continue

    # Check data sovereignty requirements
    if violates_data_sovereignty(path, constraints):
        continue

    eligible_paths.append(path)

return eligible_paths ‘‘‘
```

2.2.1.2. Step 2: Performance Scoring

Calculate performance compatibility scores for bandwidth adequacy, latency suitability, reliability, and congestion likelihood. Generate normalized scores (0.0-1.0) based on current network metrics and historical performance data.

‘‘‘python def calculate_performance_score(traffic_profile, path): """ Calculate performance compatibility score (0.0-1.0) """ scores = {}

```
# Bandwidth adequacy
bandwidth_ratio = path.available_bandwidth / traffic_profile.min_bandwidth
scores['bandwidth'] = min(1.0, bandwidth_ratio)

# Latency suitability
if path.current_rtt <= traffic_profile.max_latency:
    scores['latency'] = 1.0 - (path.current_rtt / traffic_profile.max_latency)
else:
    scores['latency'] = 0.0

# Reliability based on historical data
uptime_score = path.historical_uptime / 100.0
mtbf_score = calculate_mtbf_score(path.failure_history)
scores['reliability'] = (uptime_score + mtbf_score) / 2

# Congestion likelihood
current_utilization = path.current_load / path.capacity
congestion_risk = predict_congestion_risk(path, traffic_profile.duration)
scores['congestion'] = 1.0 - (current_utilization * 0.6 + congestion_risk * 0.4)

return scores '''
```

2.2.1.3. Step 3: Cost-Benefit Analysis

Evaluate economic factors including data transfer costs, duration charges, and setup fees against traffic profile budget constraints. Normalize cost scores where lower total costs yield higher scores, with zero score for budget-exceeding paths.

```
'''python def calculate_cost_score(traffic_profile, path): """
Calculate normalized cost score (lower cost = higher score) """ #
Calculate total cost for traffic profile data_cost =
traffic_profile.data_volume * path.cost_per_gb duration_cost =
traffic_profile.duration * path.cost_per_minute setup_cost =
path.setup_cost if path.requires_setup else 0

total_cost = data_cost + duration_cost + setup_cost

# Normalize against maximum acceptable cost
if total_cost <= traffic_profile.max_cost:
    return 1.0 - (total_cost / traffic_profile.max_cost)
else:
    return 0.0  # Exceeds budget '''
```

2.2.1.4. Step 4: Environmental Impact Scoring

Assess carbon intensity, renewable energy percentage, energy efficiency ratings, and geographic routing efficiency. Combine environmental factors into composite green scores to support sustainable networking decisions.

‘‘‘python def calculate_green_score(traffic_profile, path): """ Calculate environmental impact score (lower impact = higher score) """ # Carbon intensity scoring carbon_score = 1.0 - (path.carbon_intensity / MAX_CARBON_INTENSITY)

```
# Renewable energy percentage
renewable_score = path.renewable_percentage / 100.0

# Energy efficiency of path
efficiency_score = path.energy_efficiency_rating / 5.0  # Assume 5-star rating

# Geographic routing efficiency (shorter paths generally more efficient)
routing_efficiency = calculate_routing_efficiency(path.geographic_route)

return (carbon_score * 0.4 + renewable_score * 0.3 +
        efficiency_score * 0.2 + routing_efficiency * 0.1) ‘‘‘
```

2.2.1.5. Step 5: Failure Tolerance Evaluation

Verify each path’s ability to meet RTT degradation tolerance, bandwidth guarantees, and availability requirements. Generate binary tolerance scores based on SLA guarantees and traffic profile failure tolerance specifications.

‘‘‘python def evaluate_failure_tolerance(traffic_profile, path): """ Assess path’s ability to meet failure tolerance requirements """ tolerance_scores = {}

```
   # RTT degradation tolerance
   predicted_rtt_increase = predict_rtt_degradation(path)
   if predicted_rtt_increase <= traffic_profile.max_rtt_increase:
       tolerance_scores['rtt_tolerance'] = 1.0
   else:
       tolerance_scores['rtt_tolerance'] = 0.0

   # Bandwidth degradation tolerance
   min_guaranteed_bw = path.sla_guarantees.min_bandwidth
   if min_guaranteed_bw >= traffic_profile.min_bandwidth:
       tolerance_scores['bandwidth_tolerance'] = 1.0
   else:
       tolerance_scores['bandwidth_tolerance'] = 0.0

   # Availability requirements
   if path.sla_guarantees.uptime >= traffic_profile.min_availability:
       tolerance_scores['availability'] = 1.0
   else:
       tolerance_scores['availability'] = 0.0

   return tolerance_scores '''
```

2.2.1.6. Step 6: Composite Scoring and Selection

Calculate weighted composite scores using optimization weights and apply priority multipliers. Sort paths by final scores and select optimal path with confidence calculation and fallback path identification.

```
   '''python def select_optimal_path(traffic_profile, eligible_paths,
   weights): """ Calculate composite scores and select optimal path """
   scored_paths = ...

for path in eligible_paths:
    perf_scores = calculate_performance_score(traffic_profile, path)
    cost_score = calculate_cost_score(traffic_profile, path)
    green_score = calculate_green_score(traffic_profile, path)
    tolerance_scores = evaluate_failure_tolerance(traffic_profile, path)

    # Calculate weighted composite score
    composite_score = (
        perf_scores['bandwidth'] * weights['bandwidth'] +
        perf_scores['latency'] * weights['latency'] +
        perf_scores['reliability'] * weights['reliability'] +
        perf_scores['congestion'] * weights['congestion'] +
        cost_score * weights['cost'] +
        green_score * weights['environmental'] +
        tolerance_scores['rtt_tolerance'] * weights['rtt_tolerance'] +
```

```
        tolerance_scores['bandwidth_tolerance'] * weights['bw_tolerance'] +
        tolerance_scores['availability'] * weights['availability']
    )

    # Apply priority multiplier
    priority_multiplier = get_priority_multiplier(traffic_profile.priority)
    final_score = composite_score * priority_multiplier

    scored_paths.append({
        'path': path,
        'score': final_score,
        'component_scores': {
            'performance': perf_scores,
            'cost': cost_score,
            'environmental': green_score,
            'tolerance': tolerance_scores
        }
    })

# Sort by score (descending)
scored_paths.sort(key=lambda x: x['score'], reverse=True)

if not scored_paths:
    return None, 0.0, ..

# Select best path and prepare fallbacks
best_path = scored_paths[0]
fallback_paths = [sp['path'] for sp in scored_paths[1:4]]  # Top 3 alternatives

# Calculate confidence based on score separation
confidence = calculate_confidence_score(scored_paths)

return best_path['path'], confidence, fallback_paths '''
```

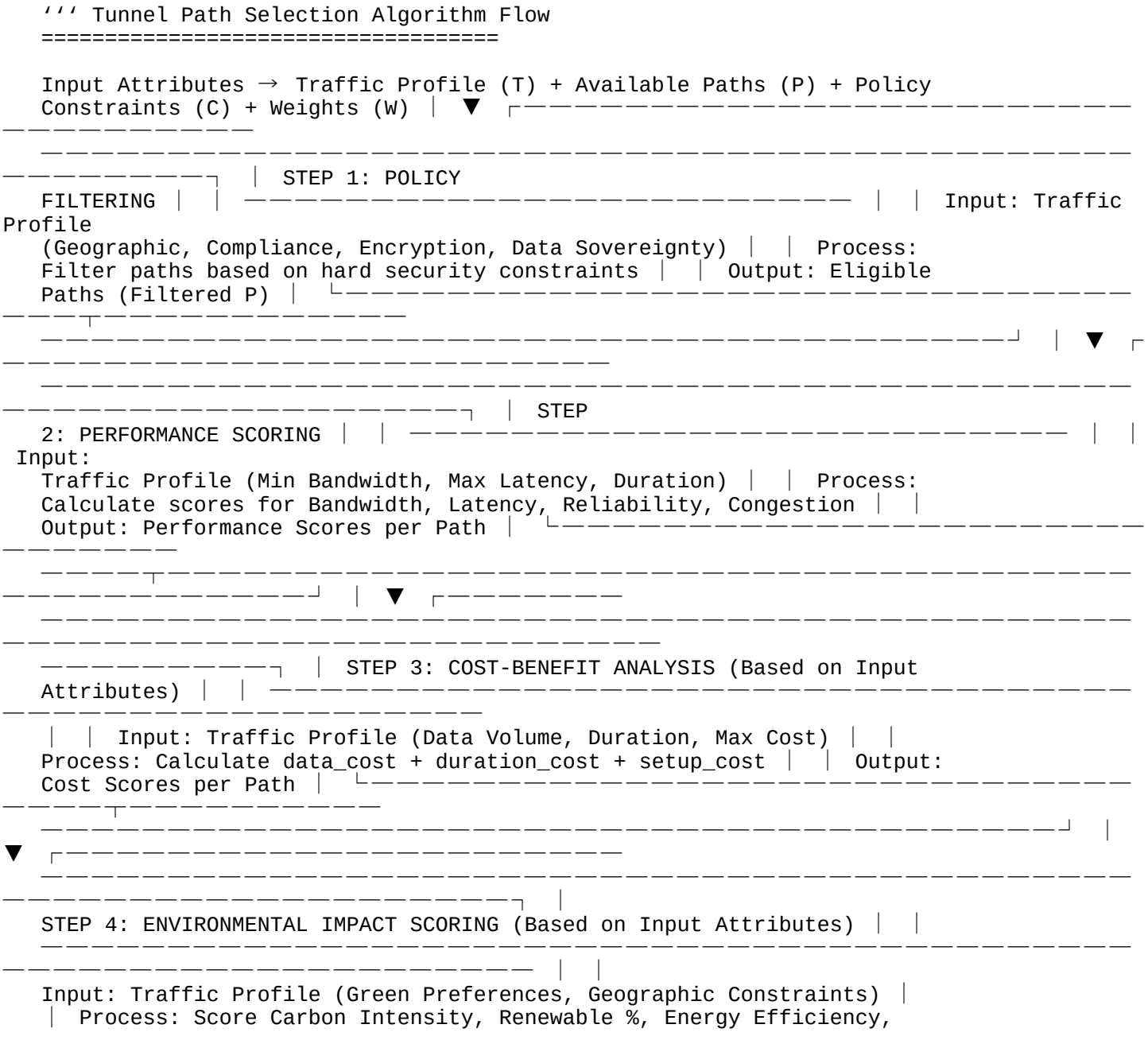
2.2.1.7. Step 7: Dynamic Re-evaluation

Monitor selected path performance continuously and trigger re-evaluation when degradation exceeds thresholds. Implement seamless path migration with minimal service disruption when better alternatives become available. The algorithm includes continuous monitoring and re-evaluation capabilities:

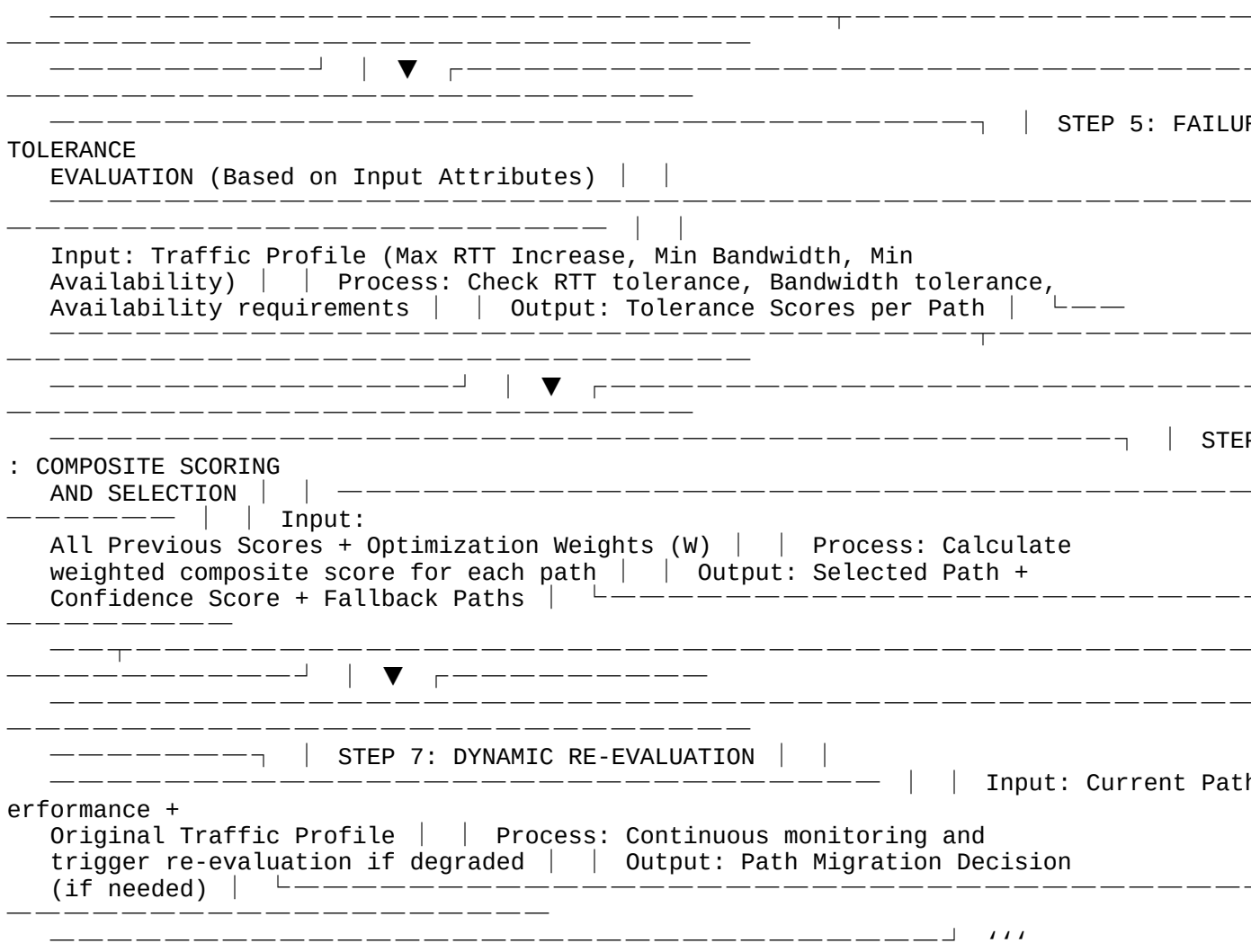
'''python def continuous_path_monitoring(selected_path, traffic_flow): """ Monitor path performance and trigger re-evaluation if needed """ while traffic_flow.active: current_metrics = measure_path_performance(selected_path)

```
    # Check if path performance has degraded
    if performance_degraded(current_metrics, expected_performance):
        # Trigger re-evaluation
        new_path, confidence, fallbacks = SelectOptimalTunnelPath(
            traffic_flow.profile,
            get_current_available_paths(),
            get_current_constraints(),
            get_current_weights()
        )

        if new_path != selected_path and confidence > SWITCH_THRESHOLD:
            initiate_path_migration(traffic_flow, selected_path, new_path)
            selected_path = new_path

    time.sleep(MONITORING_INTERVAL) '''
```

2.2.2. Algorithm Flow Diagrams



Route Efficiency
Output: Environmental Scores per Path



2.2.3. Algorithm Configuration and Weights

The algorithm uses configurable weights to adapt to different deployment scenarios:

```yaml
algorithm_weights:
  # Performance factors (sum to 1.0)
  bandwidth: 0.25
  latency: 0.20
  reliability: 0.20
  congestion: 0.15

  # Operational factors
  cost: 0.10
  environmental: 0.05

  # Tolerance factors
  rtt_tolerance: 0.02
  bw_tolerance: 0.02
  availability: 0.01

  # Priority multipliers for different traffic classes
  priority_multipliers:
    critical: 1.5
    high: 1.2
    normal: 1.0
    low: 0.8

  # Thresholds for decision making
  thresholds:
    minimum_acceptable_score: 0.6
    path_switch_threshold: 0.15 # Switch if new path scores 15% higher
    monitoring_interval: "30s"
    re_evaluation_triggers:
      - "rtt_increase > 50%"
      - "bandwidth_drop > 20%"
      - "packet_loss > 1%"
      - "availability < sla_requirement"
```

Prior work that standardized algorithms for networking include:

* Happy Eyeballs [RFC6555] and Happy Eyeballs Version 2 [RFC8305] algorithms for dual-stack hosts

2.3. Design goals

The goal of standardizing such a path selection algorithm is to enable the network devices including endpoints to make decisions independently when choosing path characteristics over others. An endpoint, for example, can achieve different prioritization based on the application contained inside flows. At the network devices the decision can be propagated or the device can re-use the same decision making algorithms at its end with richer data points to make a more optimized decision. The goals of this design are as follows : * Applications do not need to understand Failover Groups with multiple uplinks. * Avoid strict priority ordering of multiple paths. * Avoid static scheduling algorithms such as weighted round robin which do not benefit the majority of use cases such as low latency path for time-sensitive data. * Other indirect impacts of the algorithm may also be to overcome strategies which unfairly maximize bandwidth usage in the public internet.

2.4. Benefits for SD-WAN and SASE Architectures

The standardized congestion-aware multipath tunnel selection algorithm provides significant advantages for Software-Defined Wide Area Network (SD-WAN) and Secure Access Service Edge (SASE) deployments:

Unified Path Selection Framework: The standardized algorithm enables consistent path selection decisions across heterogeneous SD-WAN deployments, regardless of vendor-specific implementations. This addresses a key challenge in multi-vendor SD-WAN environments where different vendors may use incompatible path selection algorithms.

Dynamic Service Chaining: SD-WAN architectures benefit from the algorithm’s ability to dynamically route traffic through appropriate service chains based on real-time network conditions and security requirements. This enables optimal integration with Network Function Virtualization (NFV) and service function chaining deployments.

Branch Office Optimization: The algorithm’s consideration of cost factors and environmental impact aligns with SD-WAN’s focus on optimizing branch office connectivity, enabling automatic selection between MPLS, broadband, and cellular connections based on application requirements and cost constraints.

Zero-Touch Provisioning: The standardized approach supports ero-touch provisioning capabilities by providing automated path selection without requiring manual configuration of complex routing policies at each branch location.

Integrated Security and Networking: SASE architectures benefit from the algorithm’s ability to make security-aware path selection decisions that integrate network optimization with security policy enforcement, supporting the SASE principle of converged networking and security.

Edge-to-Cloud Optimization: The algorithm’s multi-dimensional scoring approach optimizes paths between edge locations and cloud services, considering latency, security, and compliance requirements that are critical for SASE deployments.

Dynamic Service Selection: SASE environments require dynamic selection between different security service instances (firewall, secure web gateway, CASB) based on traffic characteristics and performance requirements, which the algorithm supports through its service-aware path selection capabilities.

2.4.1. Zero Trust Security Considerations

In zero trust network architectures, traditional network-based trust assumptions are eliminated, requiring more sophisticated approaches to traffic classification and path selection. DSCP marking limitations present significant security and privacy risks in zero trust environments, as these markings can be easily spoofed or manipulated by malicious actors, making them unreliable indicators of actual traffic priority or security requirements. Furthermore, consistent DSCP marking patterns may reveal sensitive information about traffic nature, application types, and business processes to unauthorized observers, while violating zero trust principles that mandate "never trust, always verify" by inherently trusting network-provided markings without independent verification of their authenticity or accuracy.

The proposed algorithm addresses these zero trust challenges through self verifying traffic characteristics rather than relying on easily spoofed network markings.

2.5. Algorithm’s Requirements

The algorithm has primary goal of optimization the network path for the traffic stream for achieving the best result in terms of fairness and criticality. This algorithm must be implemented on a stateful system where the sender can make decisions on the path to be traversed.

The algorithm requires prior categorization pf paths such as uplinks based on their characteristics as type and bandwidth for example ethernet/10 megabit per second or cellular/5 megabit per second. The algorithm also requires available tunneling protocols for data transfer such as GRE, L2P, IPSec, OpenVPN and even propertiary protocols such as AutoVPN as avaiable.

The algorithm doesnot involve the approach to break out critical traffic from non-critical traffic. The algorithm should fairly suggest what traffic is to be passed through the available best path or failover to best path when experincing issues such as loss, jitter on current path.

It should - encourage load sharing between available paths - collect all data points in realtime - weights for various data points must be adjustable - have the ability to input feedback from observed performance which may be due to nested congestion control or multi-layer redudnant security etc

It should not - cause a surge of unnecessary traffic - be impacted by NAT setups - impact the outbound firewall policies

2.6. Implementation Strategies

The simplest venue for the implementation of the Path selection algorithm is within the application itself.

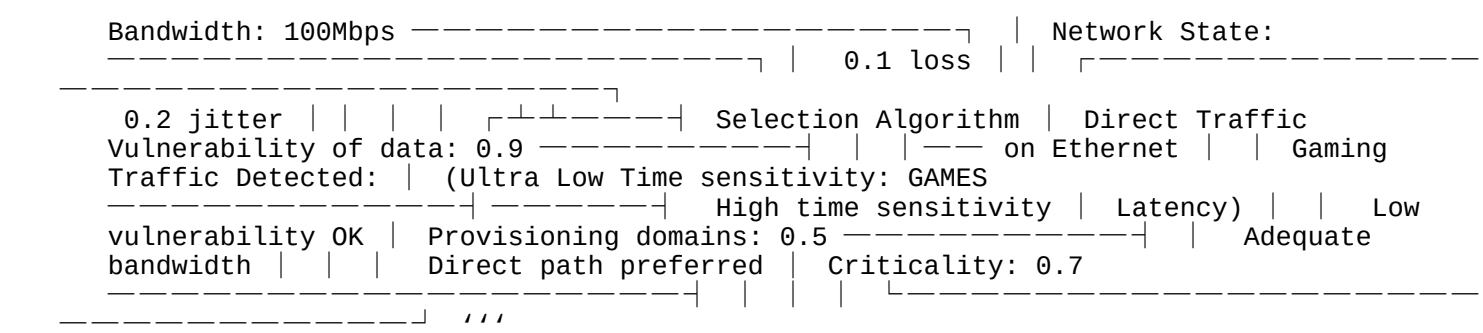
* Minimal OS support : This algorithm require no specific support from the operating system beyond the commonly available APIs that provide transport service.

* Feedback loop : The algorithm has feedback on the path consumed by all applications for this sender and tries to balance the utilization by load balacing between them. The proposed path selection algorithm is only tasked with suggesting the protocol and path and can be overridden by the application.

* Course correction : While the algorithm relies on the data points to suggest a transport protocol on a link, it can also be misguided by ambiguous or untrust worthy input. The algorithm should be self correcting with the help of feedback and any course correction should minimally impact cross traffic.

Examples of the decision that may be taken by the standardized algorithm could include: - Example 1 : Resource intensive ultra low latency application benefit from direct internet connection such as multiplayer games and if the algorithm’s path suggestion doesn’t meet the latency target the application can select its own path. ‘’’
Example 1: Multiplayer Games - Ultra Low Latency Path Selection
==========

Input Parameters (Gaming Scenario) Selection Algorithm Output Decision ========== ==========



’’’

3. Conventions and Definitions

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.

4. Security Considerations

The congestion-aware multipath tunnel selection algorithm introduces several security considerations that implementers and operators MUST address to ensure secure operation in production environments.

4.1. Algorithm Input Integrity

The path selection algorithm relies on multiple input parameters including network metrics, historical performance data, and policy constraints. Attackers who can manipulate these inputs may influence path selection decisions to their advantage:

Metric Manipulation: An attacker with access to network telemetry systems could inject false latency, bandwidth, or loss measurements to force traffic onto paths under their control or to degrade service quality. Implementations SHOULD authenticate and integrity-protect all metric collection channels. Where possible, metrics SHOULD be validated through independent measurement sources.

Historical Data Poisoning: Long-term manipulation of historical performance databases could gradually shift path selection preferences. Implementations SHOULD implement anomaly detection for historical data and SHOULD maintain audit logs of all data modifications.

Policy Injection: Unauthorized modification of policy constraints could bypass geographic routing restrictions or compliance requirements. Policy databases MUST implement strong access controls and SHOULD require multi-party authorization for policy changes affecting security-sensitive traffic classifications.

4.2. Congestion Signal Security

The algorithm’s reliance on ECN markings and congestion signals creates potential attack vectors:

ECN Spoofing: Malicious intermediate nodes could inject false ECN Congestion Experienced (CE) markings to influence path selection away from legitimate paths. While [RFC6040] provides guidance on ECN propagation in tunnels, implementations SHOULD implement mechanisms to detect anomalous ECN marking patterns that may indicate spoofing attempts.

Congestion Amplification: An attacker could artificially induce congestion on specific paths to force traffic redistribution, potentially overwhelming alternative paths. The algorithm SHOULD implement rate limiting on path switching decisions and SHOULD detect patterns consistent with deliberate congestion induction.

4.3. Traffic Analysis and Privacy

Path selection decisions may inadvertently reveal sensitive information:

Selection Pattern Analysis: Consistent path selection patterns may reveal information about traffic types, application usage, or organizational priorities to network observers. Implementations SHOULD consider adding controlled randomization to path selection decisions for non-critical traffic to reduce fingerprinting opportunities.

Timing Correlation: The timing of path switches may correlate with specific application behaviors or user activities. Implementations SHOULD avoid immediate path switching in response to transient conditions and SHOULD implement hysteresis mechanisms that obscure the relationship between traffic characteristics and path changes.

Metadata Exposure: The algorithm's input parameters, if transmitted across network boundaries, could expose sensitive operational information. All algorithm signaling between distributed components MUST be encrypted and authenticated.

4.4. Multi-Vendor Trust Boundaries

In heterogeneous deployments spanning multiple vendors, trust relationships require careful consideration:

Cross-Vendor Metric Sharing: When path selection decisions depend on metrics from different vendors' equipment, implementations MUST NOT blindly trust metrics from external sources. Cross-vendor metric exchanges SHOULD be authenticated and SHOULD be validated against locally observable network behavior.

Algorithm Coordination Attacks: In federated deployments where multiple instances of the algorithm coordinate, a compromised or malicious instance could provide false information to influence global path selection. Implementations SHOULD implement reputation systems and anomaly detection for federated algorithm participants.

Vendor-Specific Vulnerabilities: Different vendor implementations may have varying security postures. The algorithm SHOULD support configurable trust levels for different vendor domains and SHOULD allow operators to constrain path selection based on security assessments of traversed infrastructure.

4.5. Policy Enforcement and Compliance

The algorithm must ensure that security and compliance policies are consistently enforced:

Policy Bypass Prevention: The algorithm MUST ensure that performance optimization cannot override mandatory security policies. Geographic routing restrictions, encryption requirements, and compliance constraints MUST be treated as hard constraints that cannot be relaxed by the optimization process.

Audit and Accountability: All path selection decisions affecting security-sensitive traffic MUST be logged with sufficient detail to support forensic analysis. Logs SHOULD include the input parameters, evaluated alternatives, and rationale for the selected path.

Regulatory Compliance: Operators deploying this algorithm in regulated environments MUST ensure that path selection decisions comply with applicable data protection regulations. The algorithm SHOULD support configurable compliance profiles for different regulatory frameworks (e.g., GDPR, HIPAA, SOX).

4.6. Denial of Service Considerations

The algorithm may be targeted by denial of service attacks:

Path Exhaustion: An attacker could attempt to make all available paths appear unsuitable, forcing traffic to fail or use suboptimal routing. Implementations MUST maintain fallback paths and SHOULD implement graceful degradation rather than complete service denial when optimal paths are unavailable.

Algorithmic Complexity Attacks: Carefully crafted inputs could potentially cause excessive computation in the path selection algorithm. Implementations SHOULD bound computational complexity and SHOULD implement timeouts for path selection decisions.

Oscillation Induction: An attacker could manipulate network conditions to induce rapid path switching, potentially destabilizing network operations. The algorithm MUST implement dampening mechanisms to prevent rapid oscillation between paths.

4.7. Authentication and Authorization

Access to algorithm configuration and control interfaces requires protection:

Configuration Access Control: Modification of algorithm weights, thresholds, and policies MUST require authentication and authorization. Role-based access control SHOULD be implemented to limit configuration capabilities based on operator responsibilities.

Runtime Control Security: Interfaces that allow runtime modification of path selection behavior MUST be protected against unauthorized access. All control plane communications SHOULD use mutual TLS authentication.

4.8. Zero Trust Alignment

As discussed in Section 2.4.1, the algorithm SHOULD NOT rely on network-layer trust indicators that can be easily spoofed. Instead, implementations SHOULD:

* Verify traffic characteristics through behavioral analysis rather than declared markings
* Implement continuous validation of path security properties
* Assume that any network segment may be compromised and select paths accordingly
* Support integration with zero trust network access (ZTNA) frameworks for identity-aware path selection

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

[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/rfc/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", September 2001.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", November 2010.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", June 2017.

[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/rfc/rfc8174>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", December 2017.

[RFC9000] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", May 2021.

[RFC9002] Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", May 2021.

[RFC9599] Briscoe, B. and J. Kaippallimalil, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", August 2024.

6.2. Informative References

[CAREFUL-RESUME]
Seemann, M. and I. Swett, "Convergence of Congestion Control from Retained State", Work in Progress, Internet-Draft, draft-ietf-tsvwg-careful-resume-24, October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-careful-resume-24>.

[I-D.ietf-intarea-tunnels]
Touch, J. D. and M. Townsley, "IP Tunnels in the Internet Architecture", Work in Progress, Internet-Draft, draft-ietf-intarea-tunnels-15, 9 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-intarea-tunnels-15>.

[L4S-OPS] Briscoe, B., "Operational Guidance on Coexistence with Classic ECN during L4S Deployment", Work in Progress, Internet-Draft, draft-ietf-tsvwg-l4sops-08, July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-l4sops-08>.

[MULTIPATH-DCCP]
Dreibholz, A., "DCCP Extensions for Multipath Operation with Multiple Addresses", Work in Progress, Internet-Draft, draft-ietf-tsvwg-multipath-dccp-24, April 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-multipath-dccp-24>.

[NQB-PHB] White, G., Fossati, T., and R. Geib, "A Non-Queue-Building Per-Hop Behavior (NQB PHB) for Differentiated Services", Work in Progress, Internet-Draft, draft-ietf-tsvwg-nqb-33, September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-nqb-33>.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", November 2000.

[RFC3270] Le Faucheur, F., Ed., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <https://www.rfc-editor.org/rfc/rfc3270>.

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", April 2006.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/rfc/rfc5764>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", April 2012.

[RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <https://www.rfc-editor.org/rfc/rfc7556>.

[RFC8801] Pfister, P., Vyncke, ., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", RFC 8801, DOI 10.17487/RFC8801, July 2020, <https://www.rfc-editor.org/rfc/rfc8801>.

[RFC8807] Gould, J. and M. Pozun, "Login Security Extension for the Extensible Provisioning Protocol (EPP)", RFC 8807, DOI 10.17487/RFC8807, August 2020, <https://www.rfc-editor.org/rfc/rfc8807>.

[RFC9868] Herbert, T. and C. Huitema, "Transport Options for UDP", RFC 9868, December 2024, <https://www.rfc-editor.org/rfc/rfc9868>.

[UDP-ECN] Fairhurst, G., "Configuring UDP Sockets for ECN for Common Platforms", Work in Progress, Internet-Draft, draft-ietf-tsvwg-udp-ecn-03, August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-udp-ecn-03>.

Acknowledgments

TODO acknowledge.

Author’s Address

Altanai Bisht
Cisco Meraki
Email: albisht@cisco.com