Network Working Group
Internet-Draft
Intended status: Informational
Expires: 14 July 2026

E. Fjeldstrom
Independent

10 January 2026

Revisiting End-to-End in a World Without Ambient End-to-End Reachability

draft-fjeldstrom-revisiting-end-to-end-00

Abstract

This document revisits the end-to-end argument by re-deriving its premises under contemporary operational conditions. Rather than treating "end-to-end" as a slogan about host-to-host reachability, it examines how the original argument defines an endpoint in functional and administrative terms, and evaluates whether those assumptions continue to hold in modern networks.

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 14 July 2026.

Table of Contents

1. Re-Deriving the Problem: What the End-to-End Argument Claims

The end-to-end argument is a design heuristic about placement of function, not a mandate for universal connectivity. In its abstract, Saltzer, Reed, and Clark describe it explicitly as "a design principle that helps guide placement of functions among the modules of a distributed computer system" [SRC84]. This framing is intentionally broader than any particular network architecture, client-server model, or Internet-wide behavior, and emphasizes functional placement over topology or connectivity assumptions.

The argument cautions against centrally providing semantics that cannot be made complete for all applications, because doing so increases total system complexity and induces false confidence. The emergence of data communication networks further sharpened these placement questions by making clearer when and why they apply; the argument appeals directly to application requirements and provides a rationale for moving function upward in a layered system, closer to the application that uses the function.

Importantly, the argument is framed around minimal substrates (such as datagrams) because they expose failure honestly and allow applications to add semantics only when required. The argument does not assert that all communication must be host-to-host, nor that networks must be policy-free. Notably, the abstract does not even require that endpoints reside on a network at all: the principle is stated for a distributed computer system, and applies equally to inter-process communication, shared-memory subsystems, point-to-point links, or other compositions of modules.

Given this, the first question is empirical: is there evidence that nominal end-to-end paths are being bypassed in practice?

The widespread reliance on relays, overlays, traversal mechanisms, long-lived tunnels, and application-layer routing suggests that such avoidance is not isolated or accidental. These mechanisms appear independently across applications, protocols, and administrative domains, and persist even where native transport connectivity nominally exists, so it becomes necessary to re-examine what the end-to-end argument meant by an endpoint, and whether that interpretation remains valid under contemporary conditions.

1.1. Scope and Purpose

This document is written for readers evaluating architectural reasoning and historical constraints. It assumes familiarity with common Internet architectural concepts and operational terminology, and relies on background references where concepts are not re-derived in full.

This document does not address traversal, relaying, or gateway control mechanisms (e.g., [TraversalNonGoals]). Such mechanisms are referenced only to clarify scope, not to evaluate their correctness or suitability.

1.2. Historical Context: Endpoint Rationalization and Control Planes

In the original formulation, the definition of an endpoint was inseparable from assumptions about authority, signaling, and responsibility for correctness. The end-to-end argument was developed during a period of active architectural transition in distributed systems. Several contemporaneous transitions illustrate different facets of the same underlying problem: functions moved to new loci of authority without corresponding signaling or control mechanisms, forcing other layers to compensate. Networking served as a particularly clear example, but it was not the only one.

1.2.1. NCP to IP: Connection Semantics and Layering Violations

Early ARPANET communication relied on the Host-Host Protocol (NCP), in which hosts established connection-oriented relationships and the network maintained conversational state. Under NCP, lower-layer components handled reliability, ordering, and session semantics that were tightly coupled to specific applications and usage patterns. While effective in a small, trusted environment, this model did not scale and made it difficult to accommodate diverse application requirements.

The transition to the Internet Protocol (IP) deliberately removed connection semantics from the network layer, making it connectionless and pushing reliability, ordering, and application-specific correctness upward. The end-to-end argument emerged in this context as a general rationale for that shift: functions whose correctness depends on application semantics should not be embedded in lower layers that lack visibility into those semantics.

This example illustrates one direction of the recurring failure mode: when lower layers embed semantics they cannot implement correctly, higher layers must compensate or accept incorrect behavior.

1.2.2. DHCP Relay: Feasibility Boundaries and Signaling Without Relocating Authority

Early IP networks relied heavily on link-layer broadcast domains for local discovery and configuration (e.g., RARP [RFC903] and BOOTP [RFC951]). As networks grew and were subdivided into multiple IP subnets, certain functions remained logically centralized (because policy, accounting, and coordination needed to be consistent) while becoming physically unreachable using the original discovery mechanism; the systems worked well but could not scale.

Dynamic address configuration is an instructive example. A DHCP server embodies administrative authority: it assigns addresses, enforces lease policy, and reflects local operational constraints. However, DHCP's client discovery mechanism is scoped to the local broadcast domain and does not naturally traverse routing boundaries. As a result, a host may need to obtain configuration from an authoritative service that is intentionally not present on the local link.

DHCP relay [DHCPRelay] repaired this mismatch by introducing an explicit, layer-appropriate signaling path across the feasibility boundary. The relay does not assume the server's semantic role, does not reinterpret policy, and does not become a steady-state intermediary for application traffic. It forwards only configuration signaling to the locus of authority, allowing administrative control to remain centralized while restoring reachability for the control plane.

This illustrates a general pattern of correct layering: when authority cannot be exercised at the original locus of signaling due to scale or topology, the system can remain coherent if signaling is extended to match the new locus of authority, rather than forcing endpoints to compensate by re-implementing or bypassing that authority elsewhere.

1.2.3. DNS: Resolution Authority and Explicit Signaling

A second instance of endpoint rationalization occurred with naming. Early networked applications performed name resolution directly, typically by parsing local host tables (such as /etc/hosts or the centrally distributed HOSTS.TXT file) within application code. In this model, name resolution was tightly coupled to application logic, and applications were directly responsible for interpreting local configuration and resolution policy. As networks grew, name resolution authority moved outward for scalability and coordination reasons [DNSFoundations].

The introduction of the Domain Name System [STD13] provides a clear observational contrast between control-plane and data-plane behavior as scale increases. DNS was not the first distributed naming system; earlier and contemporary networks employed their own naming and directory mechanisms [DNSPriorArt]. DNS was, however, the first such system designed to operate natively over IP and across independently administered networks.

With the introduction of the BSD sockets API in the early 1980s, name resolution was refactored into a common resolver interface (e.g., gethostbyname()), removing direct host-table parsing from

applications and consolidating resolution behavior within the system library. Although resolution remained backed by purely local configuration at the time, this change established a stable application-facing boundary and decoupled application behavior from the mechanics and policy of name resolution.

When DNS was later introduced to address the scaling limits of host-table窶冱asednaming, resolution authority expanded beyond the local system to a distributed service. This shift did not require changes to applications, because applications continued to interact with the same local resolver interface. The resolver absorbed the relocation of authority beneath an existing boundary, allowing naming semantics to evolve without forcing application-level compensation.

As naming authority became distributed in practice (combining local files, DNS, and later additional directory services), systems were eventually refactored to make resolution policy explicit and governable. On Unix systems, this took the form of mechanisms such as nsswitch, which externalized ordering and selection among resolution sources that had previously been implicit. While implementation-specific, this refactoring illustrates a broader architectural pattern: once authority changed at scale, local systems generalized an initially ad hoc resolver into an explicit, policy-directed component rather than forcing applications to compensate.

DNS repaired the resulting end-to-end mismatch by introducing an explicit control plane for resolution. Application semantics remained end-to-end (applications continued to use names), but authority over resolution was relocated to a distributed service with a well-defined signaling interface.

This example illustrates a successful repair: authority moved, and an explicit signaling mechanism followed.

1.2.4. UUCP to SMTP: Rationalizing Delivery Endpoints

Electronic mail delivery provides a third example. Under UUCP-based systems, routing decisions, retry logic, and delivery semantics were embedded in applications and configuration, tightly coupled to topology and link availability. Endpoints were implicit, and responsibility for delivery was diffuse.

The transition to SMTP rationalized mail delivery by introducing explicit roles and handoff points, separating message submission, transfer, and delivery. Responsibility and policy were aligned with operational authority, and applications no longer needed to compensate for missing transport or routing semantics.

This example illustrates endpoint rationalization: by defining explicit endpoints and control-plane semantics, SMTP removed the need for application-level compensation.

1.2.5. MBONE: Compensating For Missing Functions

A related and instructive case is provided by the multicast backbone (MBONE). MBONE was explicitly deployed as an experimental system, with the expectation that its limitations would inform future architectural work rather than as an attempt to bypass or subvert existing policy mechanisms. Faced with the absence of native multicast support and control mechanisms in the deployed Internet, MBONE employed tunnels and overlay coordination to provide group communication semantics while preserving application-level abstractions.

Subsequent analyses [MBONE] showed that MBONE’s limitations arose not from negligence or protocol defects, but from a misapplication of the end-to-end principle under new conditions. Multicast group membership, resource management, and abuse prevention depend on shared policy and authority that cannot be correctly implemented purely at the endpoints. By attempting to compensate for a missing lower-layer control plane using higher-layer coordination, MBONE reproduced the same class of layering violations discussed elsewhere in this section. Its documented failure therefore reinforces, rather than contradicts, the central lesson of the end-to-end argument: correct function placement depends on where authority and context actually reside.

1.3. Synthesis: Authority, Signaling, and Layering Discipline

Taken together, these transitions demonstrate a recurring pattern. When authority over a function moves without a corresponding signaling or control mechanism, other layers will compensate by reimplementing missing functionality. When signaling moves with authority, systems regain layering discipline. The architecture described in this document addresses the contemporary instance of this pattern for admission control and reachability by restoring a layer-appropriate control plane rather than embedding lower-layer functions into higher-layer systems.

In earlier cases where authority moved, systems were refactored locally at the semantic boundary to make that movement explicit and governable. In the case of reachability and admission, the comparable refactoring never occurred.

The Internet’s architectural evolution resembles the scaling of a laboratory process to industrial production. Procedures and assumptions that are correct and effective at small scale often fail, distort, or reveal new behaviors when scaled by several orders of magnitude. This does not imply flawed reasoning at the laboratory stage; rather, it reflects the emergence of phenomena that are not observable until scale is reached. As the first network to operate continuously at global scale across thousands of administrative domains, the Internet encountered precisely this form of scale-induced novelty. Many of the resulting changes, particularly the relocation of admission authority and the loss of ambient reachability, were not fully understood at the time they occurred, and became visible only after they had already become systemic.

It is therefore reasonable to argue that these architectural effects were not fully recognized because the Internet grew too large, too quickly. The transition from a small, research-oriented network to a globally deployed infrastructure occurred within a single generation, leaving little opportunity to observe, measure, or reason about second-order effects before they became widespread. In such conditions, local solutions and operational practices necessarily preceded architectural understanding. By the time the consequences of relocated admission authority and lost ambient reachability became clear, they were already entrenched as global norms.

2. Semantic and Proximate Endpoints

It is essential to distinguish between the "semantic endpoint" and the "proximate endpoint".

The "semantic endpoint" is the locus where application-level meaning, intent, and correctness reside. This remains the host or application component that interprets data, applies application logic, and determines semantic success or failure. Nothing in this document suggests that application semantics have moved into the network.

The "proximate endpoint" is the component that first admits, rejects, or conditions communication attempts under contemporary operational policy conditions. This is the policy-enforcing boundary (firewall, gateway, or NAT) that controls reachability, authentication, authorization, accounting, and exposure before any application on the host can participate.

The core failure addressed here arises when the proximate endpoint is treated as transparent while the semantic endpoint is treated as authoritative. Applications are expected to make end-to-end decisions without being reachable, while the network boundary enforces policy without an explicit mechanism for application-level intent or consent.

The devices acting as proximate endpoints currently correspond to the broad class of middleboxes and policy-enforcing intermediaries discussed in [RFC3234].

An intuitive analogy is useful: modern networks resemble houses with locked doors. The locks are necessary and beneficial, but many systems were built as if the doors were still open. What is missing is not the lock, but the doorbell: an explicit, policy-respecting way for an outside party to announce intent and request entry without bypassing the lock.

Abstracting from historical context, an endpoint is the component that:

1. possesses application semantics relevant to correctness;
2. is trusted to make decisions;
3. can observe success and failure truthfully;
4. enforces policy and security constraints;
5. bears responsibility for failures; and
6. operates under a single administrative authority.

Nothing in this definition requires the endpoint to be a physical host; it requires alignment between semantic authority and policy enforcement.

2.1. What "Host" Meant Then, and What It Means Now

In the context of the original end-to-end argument, the term "host" referred to a computing system that was both a semantic endpoint and an administrative endpoint. A host typically had a stable network identity, direct attachment to the network, and local authority over admission and exposure for the services it ran. In that environment, placing functions "at the host" generally meant placing them at the same point where application semantics, policy decisions, and responsibility converged.

It is also important to understand what those hosts were in practice at the time the original paper was written (early 1980s). Internet-connected hosts were almost exclusively mainframes and minicomputers operated by universities, research laboratories, and government contractors. These systems were scarce, centrally administered, and

professionally managed. They typically operated under explicit Acceptable Use Policies (AUPs) and formal service-level or operational agreements, often tied to government funding or research programs. Admission control, exposure, and accountability were therefore already embedded in the administrative and contractual context surrounding each host.

In such an environment, assuming that a host could safely manage its own exposure and admission policy was reasonable. Hosts were few, well-known, and operated by organizations with both the authority and the incentive to enforce policy locally. The network itself was not expected to perform coarse-grained admission control because that function was already satisfied by institutional governance and operator oversight.

Modern usage of the term host is broader and less precise. Today, a host may be a device, a virtual machine, a container, or an application instance operating behind multiple layers of policy enforcement. Such a host often lacks stable network identity and does not control admission or exposure directly. Decisions about reachability, authentication, and authorization are frequently made upstream by components outside the host’s administrative control.

This shift does not invalidate the end-to-end argument, but it changes the meaning of applying it "at the host." In contemporary networks, the functions that SRC84 assumed to be co-located at the host are now split across semantic endpoints (applications) and proximate endpoints (policy-enforcing boundaries). Correct application of the end-to-end argument therefore requires recognizing this split and ensuring that functions are placed according to where authority and context actually reside, rather than according to legacy notions of host locality.

2.2. Boundary Security and Application Security

It is also important to distinguish between different notions of "security" that were implicitly aligned in early networked systems but are now separated.

Early multi-user operating systems already distinguished between "boundary security" and "application (or business) security". Boundary security concerned admission to the system as a whole: which users or remote systems were permitted to connect, and under what conditions. Application or business security concerned what authenticated principals were permitted to do after admission: access control within applications, enforcement of business rules, and protection of data and functionality.

In early Internet deployments, these concerns were often co-located at the host. The operating system enforced coarse-grained admission and identity, while applications enforced finer-grained semantics and policy. Because hosts were stable, centrally administered, and institutionally governed, this layering was sufficient.

As networks evolved, boundary security moved outward to dedicated policy-enforcing devices. This relocation did not eliminate application or business security, which remains the responsibility of semantic endpoints. However, the movement of boundary security was not accompanied by an explicit control plane that allowed applications and external parties to interact with boundary admission decisions.

The architecture described in this document does not seek to collapse these distinct security roles. Instead, it explicitly recognizes their separation and supplies the missing signaling mechanism required to coordinate them: boundary security continues to govern whether communication may occur, while application security continues to govern what may occur once communication is established.

A physical analogy is helpful. The lock on a building’s exterior door exists for a different purpose than the lock on a safe inside the building: one controls admission to the space, the other controls access to specific assets. Similarly, possession of a keycard or badge granting access to a particular room does not imply permission to enter the entire building. These mechanisms address different scopes of authority and different threat models.

Boundary security and application or business security function in the same way. Admission to the network boundary does not imply authorization within applications, and application credentials do not imply permission to traverse network boundaries. Correct system design preserves this separation while providing an explicit control plane that allows these layers to coordinate without conflation.

2.3. Non-Examples: Misplaced Authorization Authority

This section enumerates historical classes of mechanisms in which authorization or routing authority was placed at a location that could not reliably enforce it under realistic, multi-domain conditions. These examples are not cited to assign fault or criticize individual designs, but to delimit design space by identifying recurring structural failure modes.

Although these mechanisms often functioned initially and, in some cases, provided short-term utility, each failed to scale or remain robust once assumptions about trust, topology, or administrative unity ceased to hold.

2.3.1. Source-Address-Based Authorization

Mechanism:
: Some early systems granted access based on the apparent network origin of a request, treating source address or subnet membership as evidence of trust.

Appeal:
: In small or administratively unified networks, network location correlated strongly with organizational boundary, making this approach simple and operationally efficient.

Structural failure:
: As the Internet grew, addresses became shared, mobile, and spoofable, and intermediaries obscured true origin. Authorization inferred from network location could no longer be reliably enforced by the receiving system, leading to widespread abuse and eventual abandonment of this model.

2.3.2. Sender-Asserted Path Authority (IPv4 Source Routing)

Mechanism:
: Mechanisms that allowed a sender to specify all or part of a packet’s forwarding path effectively placed routing authority with the sender rather than with the routing domains responsible for enforcement.

Appeal:
: Source routing appeared to offer flexibility, diagnostics, and fine-grained path control in a network that was initially small and cooperative.

Structural failure:
: In a multi-domain environment, sender-asserted path authority bypassed operator policy, enabled traffic steering around controls, and undermined inter-domain trust. Networks responded by filtering or disabling such mechanisms, as path selection authority could not safely reside with remote senders.

2.3.3. Unbound Credential Assertion

Mechanism:
: Early protocols transmitted credentials or identifiers without binding them to a secure channel, session context, or time window.

Appeal:
: These mechanisms were easy to implement and interoperable in environments where interception and replay were not primary concerns.

Structural failure:
: Credentials that were not contextually bound could be replayed, forwarded, or intercepted, allowing authorization to be exercised out of context. Authority asserted without confinement proved unenforceable once adversarial conditions became common.

2.3.4. Application-Layer Authentication Above an Unprotected Substrate

Mechanism:
: Some systems placed authentication logic entirely at the application layer while assuming lower layers provided confidentiality and integrity guarantees that were not, in fact, present.

Appeal:
: This approach allowed rapid deployment without requiring changes to transport or network layers.

Structural failure:
: Authentication mechanisms could not compensate for the absence of protection beneath them. Passive interception, credential harvesting, and session manipulation were possible, demonstrating that authority cannot be enforced above a substrate that does not support it.

2.3.5. Unconfined Bearer Authority

Mechanism:
: Systems in which possession of a token, reference, or secret was sufficient for authorization, without binding that authority to a specific client, channel, scope, or lifetime.

Appeal:
: Bearer mechanisms simplified delegation and decoupled authorization from identity management.

Structural failure:
Tokens could be leaked, replayed, or misused once obtained. Authority expressed solely through possession became ambient power unless narrowly scoped and contextually constrained.

2.4. Synthesis: Authority Must Reside at the Enforcing Boundary

The non-examples described in Section 2.3 illustrate violations of the requirements derived in Section 2.2. They span multiple layers and historical periods, but share a common structural failure. In each case, authorization or routing authority was placed at a location that could not reliably observe, constrain, or account for its effects once the Internet ceased to be small, static, or trusted.

Whether inferred from network origin, asserted unilaterally by the sender, embedded in unbound credentials, or conveyed through unconstrained bearer artifacts, these mechanisms relied on assumptions that did not survive scale, mobility, or adversarial conditions. In each case, the enforcing entity lacked the ability to reliably observe context, apply policy, or account for consequences, as required by Section 2.2.

The consistent lesson is not that authentication or authorization mechanisms were poorly designed, but that authority was mislocated with respect to enforcement. Enforcement must occur at the boundary that can apply policy, bear consequences, and account for outcomes. This synthesis narrows the viable design space for addressing the endpoint shift described earlier. Any approach that attempts to restore end-to-end semantics by bypassing or ignoring the proximate enforcement boundary reproduces the same class of failure illustrated above.

2.5. Why Layering Matters: OSI as "Why," Not "What"

This document uses the OSI reference model not as a taxonomy of protocols, nor as a claim about where particular implementations belong, but as a way of reasoning about why certain questions can only be answered at certain points in a system. The OSI model was originally articulated to separate concerns of authority, feasibility, and correctness, not merely to label protocol stacks. Each layer exists because certain decisions cannot be made earlier, and must not be deferred later. The model therefore encodes impossibility results as much as it encodes structure.

The OSI reference model (as shown in Annex A of [OSI]) was not constructed as a catalogue of protocols, but as an application of architectural principles governing feasibility, scope of responsibility, and correctness. Each layer was introduced only

where a distinct class of function could not be correctly or completely performed elsewhere. In particular, the model explicitly distinguishes systems that terminate communication from those that merely forward it, and defines transport control as an end-system responsibility that is not exercised by intermediate nodes. Layer boundaries therefore encode limits on where authority and control can be placed, rather than prescribing specific mechanisms.

The OSI reference model explicitly allows for systems that do not operate across all seven layers [TR37]. An open system may be complete, with all layers functioning, or intentionally limited to a subset of layers appropriate to its role. Common examples include systems that operate only at the network layer (such as routers) or only at the physical layer (such as repeaters). These are not violations of the model, but degenerate cases of it, in which higher-layer functions are neither present nor required.

The model also accommodates failure and degradation. A system that becomes inoperative or unreliable above a given layer may be viewed as having collapsed into a less functional form (referred to as "faulty" in the report), while still potentially operating correctly below that layer. In such cases, functionality does not "shift upward" to repair the failure; rather, the effective system boundary moves downward to the highest layer that can still operate correctly.

Importantly, the presence or absence of higher-layer functionality does not eliminate responsibility for management. Each participating system remains responsible both for managing its own resources and for contributing to the management of the overall network, regardless of whether it is operating as a complete system, a limited system, or a degraded one.

Readers are not expected to be fluent in OSI terminology. The analysis that follows does not depend on memorizing layers or mapping specific protocols. It depends only on taking seriously the idea that no layer can truthfully answer questions that belong to another. Once that constraint is internalized, many behaviors that appear complex or controversial become straightforward.

This is consistent with long-standing Internet architectural guidance that emphasizes simplicity, role separation, and avoiding hidden function placement (see [RFC3439]).

2.5.1. A Pedantic but Necessary Reading of the Stack

To avoid ambiguity, the following sequence states, deliberately and pedantically, the conditions that must be satisfied for communication to be meaningful. These conditions are ordered bottom-up in terms of feasibility, but success is judged top-down by the application.

1. Can we get to the link? (Physical feasibility)
    * Is there signal, carrier, light, or RF energy?
    * Failure here renders all higher layers irrelevant.
2. Can we participate in the local network? (Link participation)
    * Are frames accepted on the medium?
    * Is there association, VLAN membership, or port enablement?
    * Failure here prevents packets from existing at all.
3. Does the destination exist at the network layer? (Network existence)
    * Is there a routable address?
    * Does the path exist to a host that can receive packets?
    * This layer answers a binary question: the service either exists here or it does not.
    * Network address translation removes ambient existence by interposing stateful mapping; whether that removal is repairable depends on administrative control of the boundary.
4. Is there an admissible transport interaction? (Transport admission)
    * Is this interaction permitted to proceed as a flow?
    * Are policy, firewalls, or admission controls satisfied?
    * This is the layer of entrance control: the building exists, but not every door is open.
5. Can a secure session be established? (Session-presentation semantics)
    * Can identity be negotiated and verified?
    * Can confidentiality and integrity be established?
    * Until this succeeds, no application-level meaning exists.
6. Is access to the correct function authorized? (Application semantics)

  * Are the requester’s credentials valid for the requested operation?
  * This is access to the correct room within the building.

No higher layer can repair a failure below it. Attempts to do so merely obscure where authority actually lies.

2.5.2. Multiple Proximate Endpoints per Layer

At any given layer, there may be one or more proximate endpoints, each independently capable of denying progress. These correspond to distinct loci of authority. For example:

* At the network layer, a host may be behind multiple translation or routing boundaries, each answering independently whether the service exists beyond that point.

* At the transport layer, admission may be enforced by host firewalls, middlebox firewalls, load balancers, or proxies.

* At the application layer, authorization may be enforced by reverse proxies, gateways, or internal policy engines.

Progress requires consent from all such endpoints. Introducing admission or authorization at one boundary does not eliminate the need for consent at others; it merely allows communication to proceed further inside. When layers are merged, such as radio systems that merge physical and link admission, or transports that merge session and presentation semantics, the number of proximate endpoints may be reduced, but their importance increases rather than diminishes.

2.5.3. "Wrong Time, Wrong Place" as a General Failure Mode

Several historical and contemporary systems exhibit the same structural failure mode: placing strong guarantees at a layer whose authority no longer aligns with the environment into which the mechanism is deployed. This is not a matter of poor design. It is an environmental mismatch.

Mechanisms that embed guarantees into a layer assume that:

* authority is routable at that layer;
* identity is stable there; and
* intermediaries are sufficiently transparent or cooperative.

Where those assumptions hold (typically within locally administered or contractually governed environments) such mechanisms function well. Where they do not, the mechanisms become brittle, niche, or require compensatory overlays that relocate the effective endpoint upward.

This pattern is not confined to any single layer. Transport-visible withdrawal of ambient reachability is merely where the effects became most visible.

2.5.4. Why This Is Not "Just a Transport-Layer Problem"

The issues described here do not originate at the transport layer. They arise whenever a layer is asked to enforce guarantees that depend on authority or identity no longer resident there.

The correct conclusion is therefore not that layer 4 is broken, but that mechanisms fail to generalize when deployed into environments whose administrative and trust structure no longer matches the layer at which their guarantees are enforced. This distinction matters because it implies that future designs which attempt to address reachability or security by modifying a single layer in isolation are likely to fail along the same lines.

3. Why Remediation Focuses Near the Transport Layer

The analysis above shows that proximate endpoints now exist from the link layer through the application layer, and failures may arise at any of these points. No single layer can be corrected in isolation. Nevertheless, remediation efforts focus on mechanisms adjacent to the transport layer. This choice reflects practical leverage, not attribution of blame.

Below the transport layer, authority is either too local, is fragmented across administrative domains, or is constrained by physical realities. Mechanisms introduced there cannot be assumed to generalize across the open Internet. Above the transport layer, applications already compensate for missing reachability through relays, overlays, and rendezvous services. While effective, these approaches harden fallback paths into steady-state infrastructure.

The transport layer therefore represents the lowest layer at which:

* the service already exists (network-layer existence has been satisfied);
* admission decisions are expected;
* intent can be expressed before forwarding; and
* success or failure can be reported honestly to applications.

Improvements at this layer propagate upward naturally, allowing higher layers to simplify rather than compensate, while avoiding the need to reassert authority at layers where it no longer resides. Accordingly, the focus on the transport layer should be understood not as privileging it, but as selecting the only layer where coordinated improvement can propagate without fighting reality elsewhere in the stack.

4. Contemporary Operational Reality

Modern networks differ fundamentally from the environment assumed by early host-centric interpretations of end-to-end communication. These differences are the result of deliberate and widely accepted operational practices.

Default-deny firewalls are ubiquitous. Network address translation (NAT) is also common, but it is largely orthogonal to the architectural problem discussed here. Hosts no longer control admission or exposure; they are shielded by policy-enforcing boundaries. Authentication, authorization, accounting, and logging are performed at the network edge, not by individual applications. Applications infer failure indirectly through timeouts and retries rather than receiving authoritative admission decisions.

As a result, a proximate endpoint now exists between communicating hosts. This endpoint is defined primarily by admission policy at the transport layer (L4), not by address translation at the network layer (L3). This boundary determines whether communication may proceed at all, independently of application intent. Treating this boundary as transparent while continuing to place endpoint assumptions at the host introduces structural ambiguity.

4.1. Why Firewalls Were Introduced

Firewalls (and related policy-enforcing gateways) were introduced to satisfy requirements that the original host-to-host communication model could not. Early Internet designs assumed mutual reachability, largely cooperative behavior, and application-managed admission. As networks grew to include multiple administrative domains, untrusted users, mobile endpoints, and economically sensitive services, these assumptions became untenable.

In its original usage in the mid-1980s, a firewall was not an inline filter but a dedicated gateway host that explicitly terminated outside connections and exercised admission authority on behalf of internal systems. Contemporary practitioner jargon defined a "firewall machine" as a Unix gateway with public-facing interfaces on one side and "one carefully watched connection back to the rest of

the cluster" on the other, explicitly tasked with servicing and controlling external access [JF]. Later commentary noted that this originally precise usage had already been semantically diluted by the late 1990s.

As devices with direct network access ("hosts") proliferated, operators required a place to enforce admission policy before exposure: to authenticate external parties, authorize communication attempts, account for usage, and deny unsolicited traffic by default. Individual hosts and applications had differing requirements and could not perform or coordinate these functions consistently or safely across many services, particularly when policies differed or needed to change rapidly. Firewalls therefore emerged as centralized admission points. They did not assume application semantics or end-to-end correctness; instead, they controlled whether communication could occur at all. This function corresponds directly to the "proximate endpoint" defined earlier in this document.

The architectural failure addressed here is not the existence of firewalls, but the absence of a corresponding control plane to bridge the resulting split in endpoints. While NAT may alter addressing, it does not itself decide whether communication is permitted; that decision is made by admission policy at the boundary. Admission authority moved to the boundary without an explicit, authenticated mechanism by which applications and external parties can express intent and obtain consent through them. Treating the firewall as transparent re-creates the original role mismatch and forces applications to compensate for missing system primitives through heuristic traversal, relaying, and tunnel-centric designs.

4.2. Resolution vs. Reachability: Control- and Data-Plane Observations

The introduction of the Domain Name System provides a clear observational contrast between control-plane and data-plane behavior under scale. Name resolution authority moved outward from individual hosts for scalability and coordination reasons, while application semantics remained end-to-end. DNS introduced an explicit control plane for resolution, allowing applications to continue using names without embedding resolution logic or inferring policy from data-plane behavior.

DNS operates as a higher-layer service that returns metadata rather than forwarding application traffic. As such, it necessarily uses a relay model. This is architecturally correct for name resolution, because relaying is intrinsic to the function being performed.

These examples differ not in intent but in kind: DNS addresses a control-plane reachability problem, restoring access to naming authority through explicit signaling, while reachability and admission govern data-plane forwarding itself.

Admission control and reachability differ in an important respect: they govern whether transport-visible connectivity may occur at all. At this layer, introducing relays or proxies transforms permission into mediation, alters transport semantics, and violates the layering discipline the end-to-end argument was intended to preserve. The architecture described in this document therefore supplies explicit authorization for native forwarding rather than proxying or relaying traffic.

In some environments, direct topology between endpoints is not available, and higher-layer relays may be unavoidable. Such relays remain appropriate as fallback mechanisms when native forwarding cannot be established. However, they are not substitutes for an explicit lower-layer control plane if one is possible. Treating relay-based connectivity as the primary model merely embeds routing and policy logic at the wrong layer and obscures the underlying architectural gap.

5. Evidence the Endpoint Has Moved

The functional shift described above is not theoretical; it is reflected in long-standing operational practice. The persistence of a proximate endpoint is demonstrated by these patterns:

* VPNs terminate connections at the policy edge to restore communication across administrative boundaries. They do not assume host-level inbound reachability; instead, they extend a trusted context inward from the boundary.
* DMZs scope exposure and centralize admission, authentication, authorization, and logging. Their ubiquity demonstrates that communication begins at the boundary, not at the host.
* Protocol misuse and layering violations, such as tunneling arbitrary traffic over HTTP or forcing real-time media onto TCP, is a symptom of missing system primitives. These designs compensate for the absence of explicit admission mechanisms and reproduce the very complexity and fragility warned against by the end-to-end argument.

6. Control-Plane Loss, Forwarding After Authority, and an STP Analogy

The preceding sections establish that nominal end-to-end paths are being systematically bypassed, and that this behavior is rational rather than accidental. The next question is therefore architectural: how is the path being bypassed, and what are the conditions requiring its bypass?

This section addresses the common objection that application-layer relays (e.g., TURN) "work well enough" in practice. The intent is not to criticize the correctness of these mechanisms, but to explain why their steady-state use produces a predictable, system-level failure mode.

6.1. TURN and Relay Mechanisms as Too-Successful Bottom-Up Compensation

Application-layer relay mechanisms such as TURN are best understood as compensatory responses to the loss of ambient transport-visible reachability. They were designed to preserve application correctness in the presence of restrictive boundary policies by re-introducing connectivity at a higher layer when direct forwarding is unavailable. This approach was both rational and necessary under the conditions in which it emerged. The relocation of admission authority to policy-enforcing boundaries occurred incrementally, under rapidly changing threat models, and without a clear, standardized interface through which external parties could express intent or request authorization. In this exploratory phase, it was not possible to specify a correct, general-purpose mid-layer admission primitive in advance. Applications therefore adapted bottom-up, discovering workable behavior through rendezvous services, relays, tunnels, and heuristic traversal.

These compensations were successful: often remarkably so. They allowed services to remain available, masked failures, and enabled deployment at global scale. Over time, however, their role shifted. What were originally fallback mechanisms hardened into steady-state infrastructure. Relay paths became primary paths; failure masking replaced explicit admission decisions; and forwarding increasingly occurred after control-plane authority had been bypassed rather than coordinated.

This transition introduces a characteristic systems failure mode. When fallback mechanisms become dominant, traffic converges onto a small number of shared relay endpoints, locality is lost, correlated failure domains expand, and load increases non-linearly. This does not reflect a defect in TURN or similar protocols. It reflects a change in operating regime: compensatory mechanisms are being asked to substitute permanently for a missing lower-layer primitive.

In architectural terms, this represents a too-successful bottom-up stabilization. The compensations worked well enough to suppress pressure to introduce an explicit, policy-respecting admission and intent control plane at the actual locus of authority. As a result, applications continue to embed routing and reachability logic above the transport layer, reproducing precisely the form of cross-layer coupling and hidden failure warned against in earlier end-to-end analyses.

6.2. Local Compensation vs. System Stability

STUN/TURN are effective compensatory mechanisms when loss of direct L4 reachability is occasional. They assume that relaying is rare, short-lived, and bounded. As network conditions evolve such that an increasing fraction of sessions require relaying, the compensatory traffic itself becomes a dominant load. This introduces positive feedback: increased relaying removes additional viable paths, which in turn increases reliance on relaying.

This is not a protocol failure; it is a phase transition from fallback behavior to steady-state operation.

6.3. An STP Misconfiguration Analogy

A close analogue exists in L2 networks employing the Spanning Tree Protocol (STP). STP is designed to prevent forwarding loops by constraining data-plane forwarding based on a control plane that reflects actual path viability. A known failure mode occurs when ports forward traffic but do not correctly participate in STP control (e.g., misconfigured edge/portfast settings, one-way BPDU handling, or links degraded by congestion rather than hard failure).

In this state:

* The data plane continues to forward frames.
* The control plane has lost authoritative visibility over viable paths.
* Forwarding persists after coordination has failed.
* Traffic reconverges onto fewer remaining links.
* Path stretch increases, locality is lost, and load concentrates.

STP may still converge correctly on a loop-free topology, yet the network becomes unusable as more links become effectively unavailable (due to congestion or policy), forcing edge routing through a narrowing core. The protocol is correct; the substrate is no longer viable.

6.4. Mapping to Application-Layer Relaying

The same failure class appears when application-layer relays are used as a steady-state substitute for missing L4 admission:

* The data plane continues to carry payloads (via relays).
* The L4 control plane no longer authoritatively constrains forwarding (due to default-deny policy, NAT behavior, or depeering).
* Applications forward after control-plane authority has been bypassed.
* Many independent flows collapse onto a small set of known-good relay endpoints.
* Path diversity shrinks; retries and duplication amplify load.

At scale, this behavior is equivalent to collapsing destination addressing toward a single forwarding identity (analogous to sending all frames toward one working port in a degraded L2 fabric). While individual sessions succeed, the shared substrate degrades for all participants.

The effectiveness of application-layer relays suggests that the system has reached a locally stable operating regime. However, that stability is maintained by continued compensatory mechanisms distributed across layers, administrative domains, and economic actors, rather than by a small set of globally enforcing primitives. In computer engineering terms, this is consistent with a metastable regime: one that can persist for long periods under prevailing conditions, yet lacks strong restoring forces should those conditions shift. Because the governing pressures span many interacting and largely exogenous degrees of freedom, no reliable prediction can be made about persistence or transition; the observation is therefore classificatory rather than prognostic.

6.5. Correctness Does Not Imply Viability

This analogy clarifies why "it works" is not a sufficient criterion. In all of the following cases, the mechanism operates as designed:

* STP converging on a loop-free tree after multiple link degradations.
* OSPF reconverging onto backup links sized for failure, not steady-state.
* Store-and-forward systems (e.g., UUCP/SMTP) delivering messages when neighbours disappear.
* TURN establishing sessions when direct reachability is unavailable.

In each case, correctness of the fallback does not imply that the resulting steady state is viable. When fallback becomes primary, capacity concentrates, locality is lost, and correlated failures emerge.

6.6. Implication

This section does not argue against relays or fallback mechanisms. It documents a known systems effect: forwarding that continues after the relevant control plane has lost authority produces instability, even when all participating protocols are correct. Restoring explicit admission at the boundary allows payload traffic to return to native forwarding, preserving path diversity and preventing the positive feedback loops described above.

7. Equilibrium and Architectural Response

The behaviors described above represent a second-order effect of earlier architectural shifts. As direct L4 reachability is withdrawn by default, the Internet’s connectivity mechanisms are reaching a new equilibrium in which application-layer relaying is no longer exceptional but structural. At this equilibrium point, a choice exists that is independent of any particular protocol design:

Accept the equilibrium:
: If application-layer relaying is treated as steady-state infrastructure rather than fallback, the network must be provisioned for sustained relay traffic. This includes accommodating persistent path convergence toward relay hubs, reduced locality, and correlated load and failure domains.

Alter the equilibrium:
: If such a steady state is not acceptable, the forces driving it must change. In practice, this requires mechanisms that restore explicit admission and authorization at the network boundary, allowing payload traffic to resume native forwarding once authorized and returning relays to an exceptional role.

This framing makes explicit that inaction is itself a decision. The system will continue to settle at the equilibrium defined by current constraints unless those constraints are altered. Subsequent sections discuss the failure modes and architectural implications that arise when fallback mechanisms become primary paths under this equilibrium.

8. Failure Modes and Implications

Given the equilibrium and second-order failure modes described above, the original end-to-end argument leads to different implications today than when it was first articulated. The original end-to-end paper documents a concrete failure in which partial guarantees at the wrong layer induced applications to assume correctness that did not exist, leading to rare but systematic corruption. The lesson is that partial guarantees at the wrong layer are worse than none.

Modern application-layer relays reproduce this failure at scale. By buffering, retrying, and synthesizing success signals above the true point of authority, they obscure routing, policy, and cost failures, defer detection, and amplify damage.

Applying the end-to-end argument correctly today therefore implies:

* treating policy-enforcing edges as endpoints for admission;
* placing authentication, authorization, and accounting at those boundaries;
* avoiding application-layer routing and relays as steady-state mechanisms; and
* providing explicit mechanisms for expressing intent to the proximate endpoint.

9. Derived Architectural Requirements

From the analysis above, any mechanism intended to preserve correct end-to-end semantics under modern conditions must satisfy the following constraints:

a. Semantic authority remains at the application or host.
b. Admission authority resides at the policy boundary.
c. Intent must be explicit.
d. Authorization must precede forwarding state.
e. Policy must not be mutable by untrusted actors.
f. No steady-state intermediaries are introduced.
g. Failure must be explicit and honest.

These requirements are architectural rather than protocol-specific.

10. Durable Proxy Authenticators

Because the proximate endpoint differs on either side of an administrative boundary, the parties cannot safely identify one another by network-layer attributes (addresses, ports, or routes). Any practical mechanism therefore requires a durable proxy authenticator: a verifiable artifact that represents identity and intent across address churn, translation, and policy mediation.

A suitable proxy authenticator has the following properties:

Durable across address instability:
: It identifies who is requesting access and what service is requested, independent of the requester’s current IP address.

Time-limited:
: Bounds authorization lifetime.

Verifiable at the boundary:
: The policy authority can validate it locally and deterministically.

Non-forgeable (accountable):
: It is attributable to an authenticated principal in a way that supports audit and dispute resolution. This property is often referred to as "non-repudiation" in PKI literature; it is used here strictly in the operational and audit sense, not as a claim of legal effect.

In practice this role may be realized using signed authorization tokens or certificate-bound grants. The document does not mandate a particular format; the architectural point is that addresses are routing hints, not identities, and the boundary requires a cryptographic or equivalent basis for trustworthy attribution.

11. Authorization-Driven Ephemeral Circuits: Architectural Context and Scope

Having derived the architectural requirements in the preceding sections, we now describe one possible architectural realization that satisfies those constraints. This section re-establishes context and scope for the architectural material that follows, without introducing new premises.

The following is an architectural model for establishing authenticated, short-lived, service-scoped forwarding state in environments lacking ambient end-to-end reachability. The model separates a long-lived control plane from short-lived data-plane forwarding state, and draws on established practices from routing, telephony signaling, and centralized authorization systems.

Rather than relying on heuristic NAT or firewall traversal, permissive default behavior, or long-lived tunnels and relays, the architecture installs narrowly scoped forwarding state only after explicit authorization by the policy authority identified earlier in this document.

The following should not be read as a return to virtual circuits or path-coupled data-plane state. Authorization affects admission, not the forwarding model itself.

This architecture is illustrative, not prescriptive. It demonstrates one way to satisfy the derived requirements, without defining a wire protocol, cryptographic scheme, or message format.

11.1. Proposal as Middle-Out Rectification of Reachability Semantics

The architecture proposed in this document should be understood not as a replacement for existing traversal or relay mechanisms, but as a middle-out consolidation of reachability semantics once the exploratory phase has ended.

The present system exhibits a clear structural split: application semantics remain end-to-end, while admission authority resides at policy-enforcing boundaries. What is missing is an explicit, standardized control plane that allows semantic endpoints to express intent and obtain authorization through those boundaries, prior to data-plane forwarding. In the absence of such a mechanism, applications have been forced to compensate by re-implementing reachability above the network and transport layers.

The proposed architecture introduces the missing mid-layer primitive directly at the locus of admission authority:

* Explicit intent expression to the policy boundary.
* Authentication and authorization prior to forwarding.
* Time-bounded, service-scoped forwarding state installed only after consent.
* Direct data-plane forwarding once authorized, without steady-state intermediaries.

Crucially, this approach does not require a flag day, nor does it invalidate existing mechanisms. Relay-based connectivity remains available as a fallback when direct authorization cannot be obtained. However, where both endpoints and the boundary support the proposed control plane, authorized native forwarding becomes the preferred path, returning relays to their intended exceptional role.

This is the defining characteristic of a middle-out repair. Existing compensatory mechanisms at the mid-level (application-layer routing, relaying, and traversal heuristics) are not abruptly removed. Instead, they are progressively displaced in the common case by a more correct lower-layer primitive, while the higher-layer application structure remains intact.

The result is a gradual cleanup rather than a redesign: fallback paths stop being structural, authority is re-aligned with enforcement, failures become explicit again, and transport semantics are restored without weakening boundary policy.

In this sense, the proposal does not alter the Internet’s security posture or attempt to resurrect ambient reachability. It completes the architectural work that exploratory, bottom-up adaptation necessarily deferred, by installing a clear, explicit interface between semantic endpoints and the policy boundaries that already govern admission.

11.2. Purpose and Scope of the Architectural Realization

The purpose of this architectural realization is to close the gap between the semantic endpoint and the proximate endpoint identified earlier in this document.

Under contemporary network practice, application semantics and correctness remain end-to-end, while admission authority has moved to policy-enforcing boundaries. This split occurred for valid operational reasons, but it was not accompanied by a corresponding control plane that allowed semantic endpoints to express intent and obtain consent through those boundaries. As a result, applications and external parties were left without an explicit signaling path to the locus of admission authority.

The architecture described here addresses that gap directly. It introduces a control plane whose sole purpose is to convey authenticated intent across the boundary separating semantic and proximate endpoints, allowing admission decisions to be made explicitly rather than inferred from data-plane behavior.

This realization is deliberately agnostic about the location of policy and enforcement. The policy authority may be centralized or distributed, co-located with enforcement points or separate from them. What matters is not physical topology, but functional placement: the control plane operates at the layer boundary where application intent meets admission policy.

Similarly, references to certificates or signed artifacts are illustrative rather than prescriptive. Such mechanisms are cited because they satisfy the architectural requirement for a durable, verifiable, and non-forgeable identifier that survives address churn and policy mediation. Any mechanism providing equivalent properties is compatible with this architecture.

In this sense, the architecture does not relocate application semantics into the network, nor does it weaken boundary policy. Instead, it restores alignment between semantic authority and admission control by providing the explicit signaling path that was missing when the endpoint split occurred. This completes, rather than contradicts, the end-to-end argument as reformulated in this document.

12. Architecture Overview

The architecture is structured around three conceptual elements:

* a policy authority at the administrative boundary;
* a control plane used to convey authenticated intent and authorization decisions; and
* a data plane consisting of ephemeral forwarding state ("circuits").

The control plane and data plane are strictly separated. Control messages never carry application data, and data flows do not require ongoing participation by the control plane once authorization has been granted.

12.1. Informational Precedents (Non-Normative)

This document does not introduce the concept of circuit establishment, authorization-before-forwarding, or control/data separation in isolation. Similar patterns have existed in prior systems and standards. The following examples are cited for instruction and architectural context only, not as templates or protocol dependencies:

IS-IS ISO10589:
Separates authenticated adjacencies (control relationships) from forwarding state. An adjacency permits the exchange of control information but does not itself create forwarding behavior; forwarding entries are installed only after control-plane convergence.

ISDN [ISDN]:
Explicitly separates signaling from bearer channels. Call setup and authorization occur on a dedicated signaling path, after which time-bounded bearer circuits are established and the signaling function withdraws from the data path. This provides a clear precedent for authorization-before-forwarding and control/data separation.

These examples are intentionally diverse and historical. They are included to show that the architectural pattern described here is well precedented, even though the specific mechanisms and environments differ.

13. Policy Authority and Interfaces

A single policy authority exists at the administrative boundary. This and the enforcement point that installs forwarding state may be co-located or separate; this document distinguishes decision-making from enforcement even when implemented within the same device. The authority is the functional realization of the proximate endpoint identified earlier in this document. It already exists in contemporary networks and routinely makes admission decisions governing reachability, authentication, authorization, accounting, and exposure.

This architecture does not introduce a new decision-maker. Instead, it adds an explicit outside-the-boundary interface that allows untrusted external parties to express intent without requiring trusted status or the ability to mutate policy. All authority remains with the boundary; only the means of expressing intent becomes explicit and policy-respecting.

Conceptually, this role is analogous to AAA-style authorization systems long used for access control, in which a centralized decision point evaluates identity and policy while distributed enforcement elements apply those decisions mechanically. This document applies the same separation of concerns to reachability and forwarding state rather than to user or device access.

Two distinct interfaces may therefore exist to this authority:

* an inside-the-boundary interface (for example, PCP [RFC6887]), used by trusted internal actors to request changes to reachability policy; and
* an outside-the-boundary intent interface (this mechanism), used by untrusted external actors to request communication without modifying policy.

These interfaces terminate at the same decision point. They differ only in trust position and privilege. All policy ownership and enforcement remain with the boundary.

13.1. Implementation Notes (Non-Normative)

This document is intentionally vague about the concrete realization of the policy authority. No assumption is made about where policy is stored, how it is expressed, or which component ultimately evaluates it. In practice, the authority may be implemented as a firewall, gateway, controller, distributed service, or composition of such elements, provided it has the ability to authenticate intent and control forwarding state.

Similarly, references to certificates or signed artifacts are illustrative rather than prescriptive. Certificates are cited because they satisfy the architectural requirements of a durable, verifiable, and non-forgeable identifier that survives address churn and supports auditability. Other mechanisms that provide equivalent properties, such as capability tokens, attribute certificates, or locally trusted credentials, are equally compatible with this architecture.

One illustrative realization uses attribute certificates for authorization (see [RFC5755]), though the specific credential format is not material to the architectural argument here.

The essential requirement is not the use of a specific technology, but the presence of a trustworthy identifier and authorization artifact that can be evaluated by the policy authority independently of network-layer addressing.

14. Control Plane

The control plane is responsible for:

* authenticating the requesting party;
* identifying the requested service;
* evaluating policy;
* issuing authorization decisions; and

* determining the lifetime and scope of any resulting forwarding state.

The control plane may be centralized or distributed, but it is logically distinct from the data plane and does not participate in steady-state data forwarding.

15. Authorization Artifacts

Authorization decisions are conveyed as authorization artifacts that permit, but do not themselves perform, the installation of forwarding state.

These artifacts are intentionally narrow in scope and limited in time. They are bound to authenticated identity, named service intent, and explicit validity periods. They do not represent sessions, transports, or application state.

15.1. Illustrative Identity Binding

One common realization binds authorization to a cryptographic principal (for example, a certificate-bearing identity) rather than to an address. In this model:

* the requester authenticates to the boundary using a durable identity credential;
* the policy authority issues or validates a short-lived grant scoped to a named service; and
* forwarding state is installed based on the observed flow at connection time, while authorization remains bound to the requester’s identity.

This allows authorized communication to survive address churn and NAT rebinding without treating addresses as stable identifiers. It also supports auditability because grants can be attributed to authenticated principals.

This subsection is illustrative and does not define a specific credential format, handshake, or PKI model.

16. Data Plane (Ephemeral Circuits)

Upon successful authorization, the policy authority installs short-lived forwarding state, referred to here as "ephemeral circuits" (authorization-scoped direct forwarding state). These are time-bounded authorization artifacts that permit data forwarding, not tunnels, sessions, or transport-visible constructs.

An ephemeral circuit is:

* temporally limited: authorization expires;
* identity-scoped: bound to authenticated principals;
* policy-bound: subject to the authority's decisions;
* non-enumerable by default; and
* removed automatically upon expiration or loss of authorization.

Ephemeral circuits represent permission to forward data, not commitment to carry it. They may be instantiated at any enforcement point for which the policy authority controls authorization decisions. In this sense, end-to-end refers not to a physical location or topology, but to a position in the network stack where application semantics meet authorization, consistent with the original end-to-end argument.

Once installed, data flows directly between endpoints without further mediation by the control plane.

16.1. Conceptual Diagram (Textual)

The following textual diagram illustrates the architecture and the placement of control and data functions. It is intended to clarify roles rather than depict physical topology.

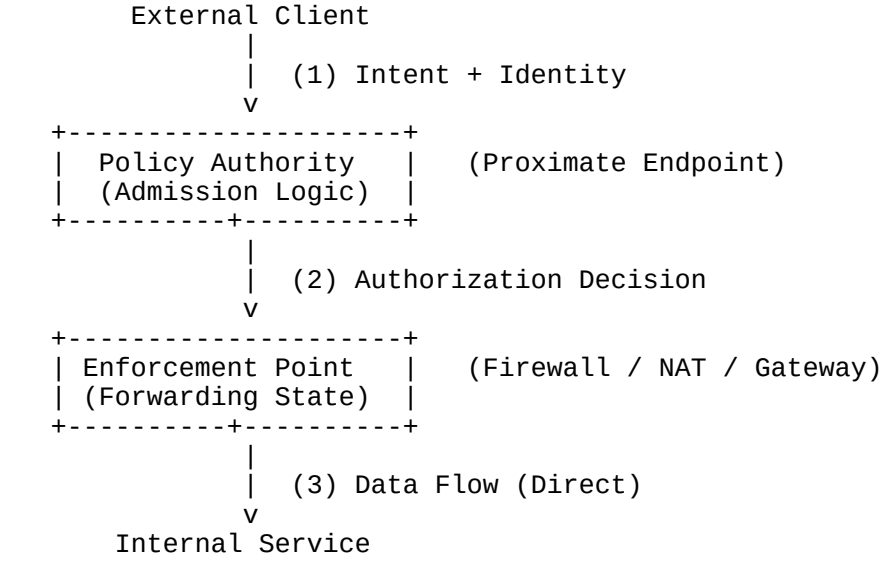

Figure 1

1. An external client expresses intent and presents a durable identity to the policy authority.
2. The policy authority evaluates policy and, if permitted, authorizes forwarding.
3. The enforcement point installs ephemeral forwarding state. Data then flows directly between endpoints.

No control traffic traverses the data path after authorization. The policy authority does not proxy, relay, or observe application data.

This diagram is conceptual: the policy authority and enforcement point may be co-located or distributed, and the authorization decision may be enforced at multiple points. The defining characteristic is the separation of intent and authorization from data forwarding, not the physical arrangement of components.

17. Circuit Lifecycle

A typical circuit lifecycle consists of the following phases:

1. Intent expression: A request to communicate with a named service is presented to the policy authority.
2. Authorization: Policy is evaluated and a decision is made.
3. Commit: Forwarding state is installed at the boundary.
4. Use: Data flows directly between endpoints.
5. Expiration: Forwarding state is removed when authorization ends.

Forwarding state is treated as soft state and may be removed without coordination.

18. Not a Tunnel, Not a Proxy

This architecture deliberately avoids long-lived tunnels and application-layer proxies.

Tunnels require intermediaries to remain continuously involved in the data path, tightly coupling authorization, transport, and availability. Proxies require protocol awareness and terminate and re-originate connections, embedding application semantics into the network.

In contrast, this architecture installs forwarding state and then exits the data path. It scales with authorization events rather than throughput and avoids application-layer routing and relaying.

19. Operation Under Strict NAT Semantics

The architecture is explicitly designed to operate under strict NAT behavior, including symmetric NAT.

No ambient inbound reachability is assumed or created. Each authorized communication results in explicit, time-bounded forwarding state. Reconnection within an authorization window does not require persistent tunnels.

20. Applicability and Limits

This architecture is best suited for environments requiring strict policy enforcement, explicit authorization, and predictable behavior, such as enterprise and service-provider networks. It is also applicable in end-user environments where admission authority does not reside within the user’s administrative control.

In many contemporary deployments, admission authority for inbound communication resides outside the end user’s domain of control. This may result from access-network architecture, shared infrastructure, mobility, policy enforcement, or other operational considerations. The specific motivations for these arrangements are not material to this discussion. What matters architecturally is that end users cannot reliably create, modify, or signal reachability policy on their own behalf using host-local mechanisms alone.

The absence of an explicit control-plane mechanism for expressing intent across this boundary forces applications serving end users into compensatory designs, such as embedding rendezvous, relaying, or routing logic above the network layer. These designs are not chosen because they are optimal, but because they provide the only available means to bridge the gap between semantic endpoints and the locus of admission authority.

The architecture described in this document does not seek to alter or override such operational arrangements. Instead, it identifies the missing signaling function required to allow intent to be expressed and evaluated even when admission authority lies outside the end user’s administrative domain.

It is not intended to replace all tunneling or proxy mechanisms, nor to provide ad-hoc NAT traversal in unmanaged environments.

21. Conclusion

The end-to-end argument was originally articulated in terms of correctness and semantics precisely because it addressed why certain functions cannot be implemented elsewhere. The difficulties examined here arise from subsequent semantic drift that treated "end-to-end" as a statement about connectivity rather than authority. Absent a renewed respect for these constraints, future mechanisms, regardless of their novelty, are likely to fail along the same lines.

Modern networks no longer provide durable, ambient end-to-end reachability. This is not a transient failure of deployment or a consequence of address scarcity, but the result of deliberate and widely adopted operational practices: default-deny firewalls, centralized policy enforcement, mobility, address translation, and administrative control. Treating these boundaries as transparent has produced a structural mismatch between application expectations and network reality, forcing applications to compensate through heuristic traversal, relaying, and tunnel-centric designs that reproduce precisely the failure modes warned against in the original end-to-end argument.

Revisiting the end-to-end argument in its original, functional sense clarifies that the principle is about where authority and semantics belong, not about preserving host-to-host reachability at all costs. Application meaning and correctness remain end-to-end, but admission, authentication, authorization, and accounting now reside at policy-enforcing boundaries. Correct application of the end-to-end argument therefore requires explicit mechanisms that allow semantic endpoints to express intent and obtain consent through those proximate endpoints, rather than bypassing or repairing them.

This document derived a set of architectural requirements from that analysis and presented one concrete realization that satisfies them: an authorization-driven model in which authenticated, time-limited, non-forgeable grants permit the temporary installation of narrowly scoped forwarding state. By separating a long-lived control plane from ephemeral data-plane state, and by binding authorization to durable identity rather than to unstable addresses, the architecture enables direct data flow without steady-state intermediaries, tunnels, or application-layer routing.

The result is not a restoration of ambient reachability, but a disciplined replacement for it: explicit intent instead of inference, authorization instead of exposure, and deterministic behavior instead of heuristic repair. In doing so, the approach preserves the spirit of the end-to-end argument under modern operational reality, aligns application design with contemporary security and policy constraints,

and offers a coherent architectural foundation for future standardization, without mandating specific protocols or deployment models.

22. IANA Considerations

This document has no IANA actions.

23. Security Considerations

This document is informational and descriptive. It does not define or modify any protocol behavior, nor does it advocate, recommend, or endorse any particular security methodologies, techniques, or deployment practices.

General guidance on certificate usage and secure transport configuration is available elsewhere [BCP195] [AuthCred]; however, it is orthogonal to the architectural questions examined here.

24. Notes on References

The following references are provided for architectural background and historical context; they should not be considered as normative implementation directions.

25. Informative References

25.1. Foundational and Architectural

[DHCPRelay]
Wimer, W., "Clarifications and Extensions for the Bootstrap Protocol", RFC 1542, DOI 10.17487/RFC1542, October 1993, <https://www.rfc-editor.org/info/rfc1542>.

Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <https://www.rfc-editor.org/info/rfc2131>.

Patrick, M., "DHCP Relay Agent Information Option", RFC 3046, DOI 10.17487/RFC3046, January 2001, <https://www.rfc-editor.org/info/rfc3046>.

[DNSFoundations]
Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <https://www.rfc-editor.org/info/rfc882>.

Mockapetris, P., "Domain names: Implementation specification", RFC 883, DOI 10.17487/RFC0883, November 1983, <https://www.rfc-editor.org/info/rfc883>.

Postel, J. and J. Reynolds, "Domain requirements", RFC 920, DOI 10.17487/RFC0920, October 1984, <https://www.rfc-editor.org/info/rfc920>.

[DNSPriorArt]
"CHAOSnet", an early packet-switched network developed at MIT that incorporated integrated naming and service discovery tightly coupled to its host and network model, 1978.

"Hesiod Name Service", a distributed naming service developed as part of Project Athena, implemented using the DNS HS class to provide directory-style lookups within IP-based environments, 1987.

Postel, J., "Internet Name Server", consolidates earlier ARPANET-era naming work beginning with IEN 61 (1978) and obsoletes several prior Internet Experiment Notes. Included for historical context, IEN 116, August 1979.

Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, DOI 10.17487/RFC0952, October 1985, <https://www.rfc-editor.org/info/rfc952>.

[ISDN] ITU-T, "ISDN Basic Rate User-Network Interface", defining the ISDN Basic Rate Interface (BRI), establishing explicit separation between a low-bandwidth D channel (architecturally treated as a control plane) and higher-bandwidth B (bearer) channels (used as the data plane) at the access edge, ITU-T I.430, 1995.

ISO/IEC, "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol (IS-IS)", defining the IS-IS link-state routing protocol originally developed for OSI CLNP; revision and consolidation of the original 1990 specification, ISO/IEC 10589, 2002.

ITU-T, "ISDN Primary Rate User-Network Interface", defining the ISDN Primary Rate Interface (PRI), extending the same low-bandwidth D channel and high-bandwidth data-plane bearer separation to trunk and carrier-scale interfaces, ITU-T I.431, 1995.

[JF] Raymond, E. S., Ed., "The Jargon File", entry "firewall machine", historical online glossary of hacker slang. Term coined mid-1980s, editorial commentary updated 1999 noting subsequent uptake and semantic drift, 1999, <http://catb.org/jargon/html/F/firewall-machine.html>.

[MBONE] Diot, C., Levine, B., Lyles, B., Kassem, H., and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture", IEEE Network, Vol. 14, No. 1, January/February 2000, pp. 78-88., DOI 10.1109/65.81917, 2000, <https://doi.org/10.1109/65.81917>.

Floyd, S., Jacobson, V., Liu, C.-G., McCanne, S., and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing", IEEE/ACM Transactions on Networking, Vol. 5, No. 6, December 1997, pp. 784-803., DOI 10.1109/90.650139, 1997, <https://doi.org/10.1109/90.650139>.

[OSI] ISO/IEC, "Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model", defining a generalized reference model for systems interconnection, not limited to computer networking; jointly developed with ITU-T, ISO/IEC 7498-1, 1994.

[SRC84] Saltzer, J. H., Reed, D. P., and D. D. Clark, "End-to-End Arguments in System Design", ACM Transactions on Computer Systems, Vol. 2, No. 4, November 1984, pp. 277-288. Revised version of a paper from the Second International Conference on Distributed Computing Systems, Paris, April 1981, DOI 10.1145/357401.357402, 1984, <https://web.mit.edu/saltzer/www/publications/endtoend/endtoend.pdf>.

[TR37] ECMA International, "Framework for OSI Management", ECMA TR/37, January 1987, <https://ecma-international.org/publications-and-standards/technical-reports/ecma-tr-37/>.

[RFC903] Finlayson, R., Mann, T., Mogul, J., and M. Theimer, "A Reverse Address Resolution Protocol", STD 38, RFC 903, DOI 10.17487/RFC0903, June 1984, <https://www.rfc-editor.org/info/rfc903>.

[RFC951] Croft, W. and J. Gilmore, "Bootstrap Protocol", RFC 951, DOI 10.17487/RFC0951, September 1985, <https://www.rfc-editor.org/info/rfc951>.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <https://www.rfc-editor.org/info/rfc3234>.

[RFC3439] Bush, R. and D. Meyer, "Some Internet Architectural Guidelines and Philosophy", RFC 3439, DOI 10.17487/RFC3439, December 2002, <https://www.rfc-editor.org/info/rfc3439>.

[STD13] Internet Standard 13, <https://www.rfc-editor.org/info/std13>. At the time of writing, this STD comprises the following:

Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

25.2. Boundary Policy and Authorization

[RFC5755] Farrell, S., Housley, R., and S. Turner, "An Internet Attribute Certificate Profile for Authorization", RFC 5755, DOI 10.17487/RFC5755, January 2010, <https://www.rfc-editor.org/info/rfc5755>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <https://www.rfc-editor.org/info/rfc6887>.

25.3. Traversal and Relay Mechanisms (Explicit Non-Goals)

[TraversalNonGoals] UPnP Forum, "Internet Gateway Device (IGD) Architecture", version 2 specification defining the UPnP Internet Gateway Device control and service model, 2015, <https://openconnectivity.org/specs/upnp/>.

Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <https://www.rfc-editor.org/info/rfc5389>.

Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <https://www.rfc-editor.org/info/rfc5766>.

Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

25.4. Transport and Security Guidance

[AuthCred] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[BCP195] Best Current Practice 195, <https://www.rfc-editor.org/info/bcp195>. At the time of writing, this BCP comprises the following:

Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <https://www.rfc-editor.org/info/rfc8996>.

Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

Appendix A. Note on Non-Goals References

The references in the "Traversal and Relay Mechanisms (Explicit Non-Goals)" section describe mechanisms intentionally not adopted by this architecture. They are cited to clarify scope and contrast design choices, not as endorsements.

Author’s Address

Erik Fjeldstrom
Independent
Email: erik_fjeldstrom@yahoo.ca