Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 February 2026

K. Williams
Independent Researcher

17 August 2025

Hierarchical Topology for Language Model Coordination
draft-williams-netmod-lm-hierarchy-topology-01

Abstract

This document defines a YANG data model and reference architecture for a hierarchical topology of language models (LMs), where tiny, small, and large LMs cooperate to perform distributed inference, summarization, and decision-making. The model supports secure inter-node messaging, request escalation, token-based authorization, and decentralized validation using pluggable trust models. This architecture is designed for deployments where computational capabilities vary across nodes, such as edge-to-cloud environments or multi-tier AI systems. The goal is to provide a standards-based mechanism for orchestrating scalable, secure LM interactions across heterogeneous systems.

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), its areas, and its working groups. 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/.

The list of current Internet-Drafts can be accessed at https://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at https://www.ietf.org/shadow.html

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 6 January 2026.

Copyright Notice

Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Recent advancements in machine learning have enabled powerful language models (LMs) to perform complex inference, summarization, and contextual reasoning. However, most production deployments assume centralized access to a single large LM, which is unsuitable for many constrained or distributed environments.

This document proposes a hierarchical model for distributed language model (LM) deployments. In this architecture, lightweight "tiny" LMs operate on constrained devices or at the edge, mid-tier "small" LMs act as aggregators or context enhancers, and a central "large" LM provides global reasoning and escalation handling.

Communication between nodes is structured using a YANG data model that supports:

* Node roles and LM types

* Secure, token-based authorization

* Inter-node RPCs for inference, escalation, and validation

* Notifications for heartbeat and liveliness reporting

This topology enables a more scalable, privacy-preserving, and resilient LM deployment strategy by allowing computation and trust decisions to occur closer to data sources. It also provides a foundation for future interoperability between language model runtimes and network control systems.

The model is inspired by hierarchical network topologies such as those defined in [RFC8345] and extends similar principles to LM-based processing pipelines.

2. Terminology

This document uses the following terminology:

Tiny LM: A lightweight, constrained language model running on edge devices, microcontrollers, or embedded environments. Capable of basic classification, keyword spotting, or template-based NLP tasks.

Small LM: A mid-tier model with enhanced summarization or contextual capabilities. Often deployed in local gateways, routers, or fog compute nodes. Serves as an aggregator and intermediary between Tiny and Large LMs.

Large LM: A centralized, full-scale language model capable of complex inference, reasoning, multi-hop retrieval, and escalation handling. Typically deployed in cloud environments or centralized data centers.

Escalation: The process by which a lower-tier LM defers processing to a higher-tier LM when its local capabilities are insufficient.

Auth Token: A signed token (e.g., JWT or CWT) used to authenticate and authorize requests between nodes. Contains claims such as ‘iss‘, ‘sub‘, ‘scope‘, ‘iat‘, and optionally a ‘nonce‘.

Validate-Token: A YANG-defined ‘action‘ that allows a node to verify the authenticity and authorization scope of a token received from a peer.

Pluggable Token Validation: A YANG ‘feature‘ that indicates support for extensible trust mechanisms (e.g., JWT verification, CBOR/ COSE decoding, external introspection endpoints).

Heartbeat: A periodic ‘notification‘ sent by a Tiny LM to indicate liveness and continued operation.

Hierarchy Topology: A tree-like structure in which Tiny LMs connect to Small LMs, and Small LMs connect to a Large LM, forming a vertical path for data and escalation flow.

3. Model Overview

The data model defined in this document represents a multi-tier topology of language model (LM) nodes, organized hierarchically into three layers: Tiny, Small, and Large. Each node type serves a specific role in the processing pipeline and communicates using a set of well-defined YANG-based interfaces.

The core components of the model include:

* A topology container that describes nodes and their relationships

* RPCs for handling inference requests (‘lm-request‘) and escalation (‘model-escalation‘)

* An ‘action‘ for validating authentication tokens (‘validate-token‘)

* A ‘notification‘ stream for liveness and heartbeat (‘lm-heartbeat‘)

* Features such as ‘pluggable-token-validation‘ to support

extensible security implementations

Tiny LMs typically initiate requests but may escalate to a Small LM if the request exceeds local capacity. Small LMs may respond directly, or further escalate to a Large LM. All communication is authenticated using signed tokens, and authorization is enforced based on node roles, token scopes, and topology position.

The model is inspired by the YANG network topology architecture defined in [RFC8345], but adapted to reflect the unique needs of language model interaction across a distributed system. It is designed to support:

* Flexible security policy enforcement

* Modular trust and validation strategies

* Constrained environments with limited resources

* Scalable coordination of inference and summarization workloads

4. Use Case Flow

This section illustrates key operational behaviors of a hierarchical language model (LM) system using the data model defined in this document. Three common interaction patterns are described: inference escalation, heartbeat reporting, and summary aggregation.

4.1. Inference Request Escalation: Tiny LM > Small LM > Large LM

Actors:

* tiny-lm-089: A constrained edge LM deployed in a local sensor

* small-lm-042: An intermediate aggregator LM with limited inference ability

* large-lm-001: A central high-capacity LM responsible for complex reasoning

Scenario:

A user inputs a query via a constrained device running tiny-lm-089. The device is unable to resolve the meaning of the input and escalates the request through its hierarchy.

Step-by-Step Flow:

1. Tiny LM Initiates Request

   Calls lm-request RPC to small-lm-042

   Includes: auth-token, source-node, request-type, payload

2. Small LM Attempts to Resolve

   Verifies auth-token via validate-token

   If unable to respond, it prepares an escalation

3. Small LM Escalates to Large LM

   Calls model-escalation RPC to large-lm-001

   Includes original-payload, reason, and its own auth-token

4. Large LM Responds

   Performs inference and returns enriched result

5. Small LM Relays Result

   Responds to the original lm-request

6. Tiny LM Displays Output

   Presents the result to the user

Security:

* Token validation at each hop

* Token scopes enforced (e.g., only small LMs can escalate)

4.2. Heartbeat Broadcast: Tiny LM > Topology

Actors:

* tiny-lm-089: An edge device LM

* small-lm-042: Its supervising node

Scenario:

To maintain system health, each tiny LM emits a periodic heartbeat signal to its parent.

Step-by-Step Flow:

1. Tiny LM Sends Heartbeat

   Emits lm-heartbeat notification with timestamp and status

2. Small LM Receives Notification

   Subscribed to lm-heartbeat stream

   Updates health status table or triggers alert on timeout

Security:

Not signed by default, but implementations MAY correlate with recent authenticated activity

4.3. Summary Aggregation: Small LM > Large LM

Actors:

* tiny-lm-089, tiny-lm-090, tiny-lm-091: Sensor LMs

* small-lm-042: Aggregator

* large-lm-001: Reasoning LM

Scenario:

Multiple tiny LMs submit observations. The small LM combines and escalates them.

Step-by-Step Flow:

1. Tiny LMs Submit Observations

   Each sends lm-request to small-lm-042 with its own auth-token

2. Small LM Aggregates Input

   Locally summarizes data from tiny LMs

3. Optional Escalation

   Sends model-escalation RPC to large-lm-001 with the summary

4. Large LM Enhances Result

   Returns executive-level context or response

5. Small LM Caches + Responds

   Updates cache and optionally forwards summary or alert

Security:

* All requests must carry valid, scoped auth-tokens

* Escalation privileges restricted to authorized node types

5. Data Model Summary and Usage

The data model defined by this document describes a hierarchical topology of language model (LM) nodes and the interfaces through which they communicate, authorize, and escalate inference operations. It is expressed using the YANG 1.1 data modeling language [RFC7950].

The model includes the following key elements:

* A ‘lm-node‘ container with identity-based classification (‘tiny‘, ‘small‘, ‘large‘)

* RPCs:

  - ‘lm-request‘: Initiates an inference or summarization task

  - ‘model-escalation‘: Forwards requests upward in the LM hierarchy

* Actions:

- ‘validate-token‘: Verifies token authenticity and scope

* Notifications:

- ‘lm-heartbeat‘: Indicates node liveness and status

* Groupings for ‘auth-token‘, trust metadata, and request payloads

* A ‘feature‘ flag (‘pluggable-token-validation‘) to support modular trust infrastructure

All inter-node requests include a signed ‘auth-token‘, which may be validated locally via ‘validate-token‘ or externally if the feature is supported.

The full YANG module is provided in Appendix A.

6. Implementation Status

NOTE TO RFC EDITOR: This section is to be removed before publication.

This section documents the current implementation efforts related to the YANG model and architecture described in this draft. It is included to inform reviewers and working group participants of the maturity and deployment experience of this specification.

Title: UniLoRa Mesh LM Hierarchy Prototype
Authors: Keenan Williams
Maturity Level: Early Prototype
Development Status: Active

Description:

A functional prototype of the hierarchical LM topology described in this document has been implemented as part of the UniLoRa Mesh project. The prototype demonstrates communication between Tiny, Small, and Large LM nodes using the defined YANG data model over a LoRa-based mesh transport layer.

Key features supported:

* ‘lm-request‘: Implemented on Tiny and Small LMs to initiate and forward inference requests

* ‘model-escalation‘: Fully implemented for upward delegation to a central reasoning engine

* ‘validate-token‘: Implemented on Small and Large LMs using JWT-based verification with public key validation

* ‘lm-heartbeat‘: Actively used to track the liveliness of edge nodes

* ‘pluggable-token-validation‘: Enabled; token verification can be swapped between local crypto module or cloud introspection service

* YANG model validated with ‘pyang‘ and integrated with a prototype RESTCONF endpoint

Deployment:

* Tiny LM: ESP32-based TTGO T-Beam devices running lightweight keyword spotting and rule-based summarization

* Small LM: Raspberry Pi 5 devices running a lightweight Python LM with summarization and caching logic

* Large LM: Central node hosted on a cloud container, running GPT-style inference with trust policy enforcement

The system supports real-time inference routing from edge to core, token-authenticated message passing, and topology-driven trust enforcement as defined in this draft. The goal is to refine this prototype into a reference implementation that can be used for interoperability testing and NETCONF/RESTCONF YANG validation tooling.

Source Code Repository: [to be published]
License: Apache 2.0

Feedback and collaboration are welcomed to further validate this model across constrained and distributed environments.

7. IANA Considerations

This document registers one URI in the " XML Registry" [RFC3688] and one YANG module name in the "YANG Module Names" registry [RFC6020].

7.1. XML Namespace Registration

URI: urn:ietf:params:xml:ns:yang:ietf-lm-hierarchy
Registrant Contact: NETMOD Working Group
XML:

```
<namespace>

  urn:ietf:params:xml:ns:yang:ietf-lm-hierarchy
</namespace>
```

7.2. YANG Module Name Registration

Name: ietf-lm-hierarchy
Namespace: urn:ietf:params:xml:ns:yang:ietf-lm-hierarchy
Prefix: lm
Reference: This document (draft-williams-netmod-lm-hierarchy-topology)

8. Security Considerations

This document defines a hierarchical topology model for distributed language models (LMs), where communication occurs between nodes of differing capabilities and privileges (Tiny LMs, Small LMs, and a central Large LM). To maintain the integrity, trustworthiness, and isolation of operations within such a topology, security is critical.

8.1. Authentication and Authorization

All inter-node communication is required to include an ‘auth-token‘, as defined in the data model. These tokens may be bearer tokens, CBOR Web Tokens (CWTs), or JWTs signed by trusted issuers (e.g., Large LMs or centralized authorities). The model assumes a shared trust infrastructure wherein Large LMs issue or delegate trust tokens to downstream nodes.

To support flexible, decentralized validation, this YANG module defines a node-level ‘validate-token‘ action. This action enables nodes to verify the authenticity and scope of received tokens at runtime. While token fields alone do not enforce authorization, this action provides a behavioral interface for systems to verify and act on trust decisions.

This approach avoids reliance on centralized introspection services, which may not be suitable for bandwidth-constrained or delay-sensitive environments (e.g., when Tiny LMs operate at the edge).

8.2. Pluggable Trust Model

A YANG ‘feature‘, ‘pluggable-token-validation‘, is defined to indicate support for extensible validation backends (e.g., certificate chains, OAuth2 introspection endpoints, COSE/CBOR decoders, etc.). This allows implementations to declare advanced trust handling capabilities without forcing them into minimal deployments.

8.3. Replay and Scope Protection

Tokens should include ‘exp‘ (expiration) and ‘iat‘ (issued at) claims to protect against replay. Where possible, ‘nonce‘ or one-time identifiers should be used to detect message duplication. Token ‘scope‘ claims (e.g., ‘inference‘, ‘summarization‘) should be enforced by recipient nodes to prevent privilege escalation across the hierarchy.

8.4. Topological Access Controls

Nodes should reject incoming requests from unauthorized peers based on:

* Node type (e.g., a Tiny LM may not issue requests to another Tiny LM),
* Issuer identity,
* Token scope.

Large LMs SHOULD enforce policies on which nodes may act as intermediaries (e.g., only trusted Small LMs may escalate).

This layered security model ensures each node in the hierarchy enforces local trust decisions, minimizing blast radius in the event of compromise and allowing granular control over inter-node permissions.

8.5. Token Format (Informative)

This document assumes the use of signed tokens to authorize inter-node communication. While the token format is implementation-specific, systems are RECOMMENDED to use existing standards such as:

* JSON Web Tokens (JWT) [RFC7519]

* CBOR Web Tokens (CWT) [RFC8392]

Example minimal JWT claims:

```
{
  "iss": "large-lm-001",
  "sub": "tiny-lm-089",
  "scope": ["inference", "summarization"],
  "exp": "2025-07-06T16:00:00Z",
  "iat": "2025-07-06T15:00:00Z",
  "nonce": "3e8f5b5b-c21e-47a0-92a2-1f6ad919ef55"
}
```


Tokens MAY be passed in cleartext (if signed) or encrypted (if confidentiality is required). Nonce tracking, token expiration, and scope enforcement SHOULD be implemented at all receiving nodes.

9. References

9.1. Normative References

[RFC3688] Mealling, M., "The XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

9.2. Informative References

This document has no informative references.

Appendix A. YANG Module

<CODE BEGINS> file "ietf-lm-hierarchy@2025-07-06.yang"

```
module ietf-lm-hierarchy {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-lm-hierarchy";
  prefix lm;

  import ietf-yang-types { prefix yang; }
  // NOTE: Removed unused import of ietf-inet-types

  organization
    "IETF NETMOD (Network Modeling) Working Group";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author: Kevin Williams
             <mailto:telesis001@icloud.com>";

  description
    "This module defines a YANG data model for hierarchical Language
Model (LM) nodes that coordinate inference, summarization and
escalation across tiny/small/large LM tiers.

Copyright (c) 2025 IETF Trust and the persons identified as the
document authors.  All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal
Provisions Relating to IETF Documents
(https://trustee.ietf.org/license-info) in effect on the date of
publication of this document.  Please review these documents
carefully, as they describe your rights and restrictions with
respect to this document.  Code Components extracted from this
document must include Revised BSD License text as described in
Section 4.e of the Trust Legal Provisions and are provided without
warranty as described in the Revised BSD License.

This version of this YANG module is part of draft-williams-netmod-lm-hierarchy-topology-0
1;
see the Internet-Draft itself for full legal notices.";
  reference
    "RFC 7950: The YANG 1.1 Data Modeling Language
     RFC 8407: Guidelines for Authors and Reviewers of YANG Data Model Documents";

  revision 2025-07-06 {
    description
      "Initial revision for I-D draft-williams-netmod-lm-hierarchy-topology-01.";
    reference
      "I-D: draft-williams-netmod-lm-hierarchy-topology-01";
  }

  feature pluggable-token-validation {
    description
      "Indicates support for a pluggable token validation mechanism for
       requests handled by LM nodes.";
  }
```

```
identity lm-node-type {
  description "Base identity for LM node types.";
}

identity tiny-lm {
  base lm-node-type;
  description "A lightweight edge-deployed language model.";
}

identity small-lm {
  base lm-node-type;
  description "A mid-tier aggregator or summarizer.";
}

identity large-lm {
  base lm-node-type;
  description "A central reasoning or escalation endpoint.";
}

grouping auth-token-grouping {
  description "Reusable auth-token structure.";
  leaf auth-token {
    type string;
    description "A signed authentication/authorization token.";
  }
}

// Top-level configuration for LM hierarchy
container lm {
  presence
    "Enable LM hierarchy configuration on this device.";
  description
    "Presence of this container enables configuration and state for
     the Language Model (LM) hierarchy on the device.";

  container node {
    description
      "Attributes describing this node within the LM hierarchy.";

    leaf node-id {
      type string;
      description
        "Implementation-specific identifier for this LM node.";
    }

    leaf node-type {
      type identityref { base lm-node-type; }
      description
        "Classification of this node (tiny, small, large).";
    }
  }

  container trust {
    if-feature pluggable-token-validation;
    description
      "Configuration for token validation used to authorize LM
       requests and actions.";

    leaf trust-anchor {
      type string;
      description "Root or public key used for token validation.";
    }

    leaf token-scope-enforced {
      type boolean;
```

```
      default true;
      description "Whether to enforce scope claims in tokens.";
    }

    action validate-token {
      description
        "Validate a presented token against the active trust policy.";
      input {
        leaf token {
          type string;
          description
            "Opaque bearer or structured token presented by a caller.";
        }
      }
      output {
        leaf valid {
          type boolean;
          description
            "True if the token is valid per current trust policy.";
        }
        leaf reason {
          type string;
          description
            "Human-readable reason when validation fails, or an
             implementation-specific note when it succeeds.";
        }
      }
    }
  }
}

rpc lm-request {
  description
    "Submit an LM request for processing within the hierarchy.";

  input {
    uses auth-token-grouping;
    leaf source-node {
      type string;
      description
        "Identifier of the node submitting the request.";
    }
    leaf target-node {
      type string;
      description
        "Desired target node identifier; implementations may ignore
         and route based on policy and availability.";
    }
    leaf request-type {
      type enumeration {
        enum inference {
          description "Perform inference over provided payload.";
        }
        enum summarization {
          description "Summarize the provided payload.";
        }
      }
      description
        "The kind of operation requested.";
    }
    leaf payload {
      type string;
      description
        "The operation payload. Encoding is implementation-specific
         (e.g., JSON).";
    }
```

```
  }
  output {
    leaf result {
      type string;
      description
        "The result produced by the LM operation (implementation-
         specific encoding, e.g., JSON).";
    }
    leaf status {
      type enumeration {
        enum ok {
          description "The request completed successfully.";
        }
        enum error {
          description "The request failed; see result for details.";
        }
      }
      description
        "Status of the processed request.";
    }
  }
}

rpc model-escalation {
  description
    "Request an escalation to a different LM tier (e.g., from tiny to
     small/large) with rationale.";

  input {
    uses auth-token-grouping;
    leaf original-payload {
      type string;
      description
        "Original payload requiring escalation.";
    }
    leaf reason {
      type string;
      description
        "Reason for escalation (e.g., insufficient context, model
         limitations, or policy).";
    }
  }
  output {
    leaf resolution {
      type string;
      description
        "Resolution or answer produced after escalation.";
    }
    leaf downstream-directive {
      type string;
      description
        "Optional directive for downstream workers/coordinators.";
    }
  }
}

notification lm-heartbeat {
  description
    "Periodic heartbeat emitted by LM nodes to advertise presence and
     health.";

  leaf sender-node {
    type string;
    description
      "Identifier of the node emitting the heartbeat.";
  }
```

```
        leaf status {
          type enumeration {
            enum alive {
              description "Node is reachable and operating nominally.";
            }
            enum degraded {
              description "Node is reachable but operating with reduced capacity.";
            }
            enum unreachable {
              description "Node is currently unreachable.";
            }
          }
          description
            "Operational status reported by the node.";
        }
        leaf timestamp {
          type yang:date-and-time;
          description
            "Timestamp for the emitted heartbeat.";
        }
      }
    }

    <CODE ENDS>
```

Author’s Address

Keenan Williams
Independent Researcher
Email: telesis001@icloud.com