Computing-Aware Traffic Steering Q. Li
Internet-Draft T. gao
Intended status: Informational Pengcheng Laboratory
Expires: 2 September 2026 Y. Jiang
Tsinghua Shenzhen International Graduate School & Pengcheng Laboratory
1 March 2026

IntelliNode: In-Network Intelligent Scheduling Extensions for CATS
draft-li-cats-intellinode-network-scheduling-00

Abstract

This document introduces IntelliNode, an in-network intelligent scheduling mechanism built upon the Computing-Aware Traffic Steering (CATS) framework. Modern large-scale AI training and inference heavily rely on distributed heterogeneous clusters (GPU/CPU/FPGA). However, existing networks lack awareness of tensor semantics, training phases, and heterogeneous computing capabilities, leading to high communication latency, low resource utilization, and pipeline stalls.

IntelliNode shifts away from the traditional passive scheduling paradigms that rely on probes and controllers. By bypassing traditional paths and integrating FPGAs alongside programmable Switch ASICs, it constructs a rapid data-plane closed loop of "Perception-Inference-Decision-Execution". This architecture performs feature extraction at line rate, leverages lightweight prediction models to infer short-term network behavior, and drives real-time heuristic scheduling decisions (e.g., path selection, tensor slicing, and compute matching). This document defines the four core functional layers and extension signaling that support this architecture, laying the foundation for an AI-native, scalable distributed computing network.

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/.

Copyright Notice

Table of Contents

1. Introduction

The CATS framework primarily addresses the selection of service instances and computing-aware traffic steering in general distributed systems. However, when confronting large-scale AI training, distributed inference, and heterogeneous computing clusters, AI workloads exhibit traffic dynamics on the order of microseconds to milliseconds, accompanied by highly diverse tensor types (e.g., gradients, activations, parameters).

Traditional CATS models (control-plane decisions combined with service-level steering) are inadequate for these next-generation computing workloads. IntelliNode proposes an extended architecture deeply embedded in the data plane. It not only natively processes

RoCEv2 protocol semantics but also transforms the network from a "passive data pipe" into an "active, computing-aware collaborative engine".

2. Problem Statement

Applying existing network scheduling mechanisms to AI training and heterogeneous computing networks reveals the following fundamental limitations:

* Tensor Semantic Blind Spot: Existing mechanisms cannot distinguish specific semantics within data streams, such as gradients, activations, or parameter updates.

* Lag in End-to-End Feedback: Mechanisms like ECN misinterpret "passive feedback" as "active scheduling" and assume that "rate reduction is the correct response." This completely ignores the computing semantics in AI inference, where certain flows "cannot slow down, but must wait or degrade precision."

* Excessive Control-Plane Decision Latency: Control-plane routing updates, which take hundreds of milliseconds to seconds, cannot handle transient congestion or iterative bursts within a 1-5 millisecond window.

* Conflict between Isomorphic Assumptions and Heterogeneous Reality: In cross-domain computing networks, node capabilities are highly uneven (e.g., GPU/FPGA hybrids). The network must possess global state awareness to accurately match computing power with communication workloads rather than relying on isomorphic computing assumptions.

3. Architecture

The IntelliNode architecture consists of four tightly coordinated functional layers that perfectly align with CATS’s abstractions for information collection, decision engine, and steering plane. This architecture fuses the capabilities of programmable switches, FPGAs, and CPUs at the local node, enabling a microsecond-level closed loop without interrupting the packet forwarding path.

3.1. Feature Extraction Layer (Switch ASIC)

Deployed on Tofino-class programmable switch ASICs, this layer actively participates in RoCEv2 traffic management. It maintains a high-performance Queue Pair (QP) flow state machine. The switch collects and parses real-time features at line-rate, including:

* Basic Network Features: Ingress port, transmission rate, flow size, queue depth, and link utilization.

* AI Semantic Features: Tensor type (gradient / activation / normal traffic), tensor position within a batch/iteration, the stage of the model-parallel pipeline, and whether it is cross-node gradient-sync traffic.

* Flow State Classification: The hardware identifies the flow's current state as UNALLOCATED, SMALL_FLOW (delay-sensitive/ control), LARGE_FLOW (high-bandwidth parameter synchronization), or DRAINING (tail-end flushing).

These features are extracted, normalized, and encoded at line-rate, then written into a high-speed featureFIFO to be sent directly to the onboard FPGA. Simultaneously, the pipeline incorporates real-time checksum updates and validation logic for mutable fields in RoCEv2 (such as ECN and TTL markings) to ensure protocol legitimacy.

3.2. State Prediction Layer (FPGA)

The FPGA reads features from the featureFIFO and executes an ultra-low latency, lightweight prediction model (e.g., State-GNN based on Graph Neural Networks, linear regression, or heuristic models). This layer focuses on predicting short-term network and load states within the next 1-5 milliseconds (ms):

* Network State Prediction: Imminent congestion risks on switch ports and the available bandwidth of candidate routing paths in the next window.

* Computing Load Prediction: The arrival time of the next batch of periodic tensor traffic, and the probability of queuing backlogs or pipeline stalls at the downstream GPU.

These forward-looking prediction fields serve as the core input for the subsequent scheduling engine.

3.3. Heuristic Scheduling Layer

The scheduling engine integrates the currently extracted AI semantics with the predicted states output by the FPGA, approximating Pareto Optimality amidst conflicting multi-objective goals (e.g., computing latency vs. communication overhead). The decision logic is based on:

* Tensor type and structural priority.

* Operator dependency.

* Heterogeneous computing capabilities of target nodes.

* 1-5ms network and congestion predictions.

The decision outputs (execution actions) include:

* Path and Priority: Outputs the optimal path set. SMALL_FLOWs are prioritized based on arrival rate, while LARGE_FLOWs dynamically allocate bandwidth based on target computing power using a Weighted Deficit Round Robin (WDRR) policy.

* Tensor Slicing: Determines if tensor slicing is necessary, defining the number of slices and the independent routing path for each.

* Multipath Aggregation: Decides whether to enable data-plane multipath aggregation.

* In-Network Offloading: Decides whether to offload specific operators (e.g., Sum/Reduce) to in-network FPGAs or edge nodes.

3.4. Steering Plane

The output of the heuristic scheduling must be applied to the entire network data plane via a lightweight signaling mechanism (potentially as an extension to CATS-SR or CATS-Overlay):

* Control Plane Interface: Triggers the local CPU to update the routing/forwarding tables, applying the latest policies to the next batch of traffic automatically.

* Data Plane Labels / TLVs: Pushes extended Metadata TLVs carrying tensor types, training phases, and compute resource requests into the packet header.

* Fragment Routing Encapsulation: Provides necessary Encapsulation information for traffic requiring Tensor Slicing.

4. Security Considerations

Given that IntelliNode introduces granular TLV fields for tensor semantics and active data-plane scheduling, the system MUST:

* Provide integrity protection for TLV fields to prevent malicious nodes from tampering with "Tensor Types" to preempt high-priority queues.

* Introduce encrypted control-plane channels for telemetry and configuration.

* Implement authentication to prevent unauthorized nodes from falsely broadcasting their Compute-Capability within the computing network.

5. IANA Considerations

This document requests that IANA allocate new TLV types for AI-native CATS deployments, including but not limited to:

* TENSOR_TYPE TLV

* TRAINING_PHASE TLV

* COMPUTE_CAPABILITY TLV

* PATH_PREDICTION TLV

Authors' Addresses

Qing Li
Pengcheng Laboratory
Email: liq@pcl.ac.cn

Teng gao
Pengcheng Laboratory
Email: gaot@pcl.ac.cn

Yong Jiang
Tsinghua Shenzhen International Graduate School & Pengcheng Laboratory
Email: jiangy@sz.tsinghua.edu.cn