Layer Communication Framework (LCF) For Real-Time, AI-Driven System Communications

Razvan Cristian Voicu†, Muhammad Hassan Tanveer, Yusun Chang
Department of Robotics & Mechatronics Engineering, Kennesaw State University, Atlanta, Georgia
Email: voicu@gatech.edu, mtanveer@kennesaw.edu, ychang7@kennesaw.edu,

***Abstract*—This article introduces the Layer Communication Framework (LCF), a solution designed for modern AI systems, particularly in machine-to-machine and real-time edge environments. Building on programmable networking and parallel communication, LCF streamlines data exchange and improves responsiveness in AI-driven applications. A flexible multi-layer design supports one-to-many, many-to-one, and pass-through messaging patterns, enabling robust interactions while minimizing dependence on specialized prior frameworks. LCF explicitly separates *soft real-time* coordination loops (e.g., high-level robotic task negotiation or distributed perception fusion) from strict hard real-time motor control. These coordination loops typically operate in the 50–150 ms range, where latency on the order of 100 ms remains acceptable for safe and responsive behavior, while lower-level controllers run at millisecond scale on-board the device. By integrating mathematical performance models for delay, throughput, and reliability, LCF demonstrates quantifiable improvements over conventional approaches. Experimental results from NS3 highlight near real-time performance gains under soft real-time constraints and show that LCF can effectively support AI coordination across edge and cloud. Consequently, LCF enables more advanced AI applications at the edge and offers a structured communication substrate for emerging tool-based orchestration protocols.**

***Index Terms*—Programmable Networking, Parallel Communication, Real-time Systems, Multipath Communication, AI Communication**

I. Introduction

Artificial Intelligence (AI) has moved from academic prototypes into deployed systems across healthcare, automotive, telecommunications, and consumer products. With increased computational power and sophisticated models, new forms of intelligent automation and decision-making are now possible. Fully exploiting these capabilities, however, requires communication infrastructures that can keep pace with AI workloads rather than treat them as ordinary traffic.

A critical requirement is efficient data transmission and machine-to-machine interaction under real-time constraints. AI-driven applications often process large data volumes and must respond quickly; frameworks designed for static, best-effort traffic are frequently inadequate. Network architectures must adapt on the fly, support diverse modalities, and maintain sub-second latencies for tasks ranging from autonomous control to near-instant analytics.

To ground the discussion, this work focuses on an autonomous mobile robot fleet that relies on a hierarchy of AI components. Embedded controllers run motor and safety loops at 1–5 ms periods to ensure stability and collision avoidance. Above this, a soft real-time coordination layer performs task assignment, route replanning, and environment-aware behavior by querying local and cloud-hosted AI models. In such settings, end-to-end communication delays in the 50–150 ms range remain acceptable for coordination, as robots can still adjust trajectories or switch tasks while low-level loops guarantee immediate safety. The communication framework must therefore support soft real-time behavior for high-level AI interactions without attempting to replace hard real-time motion control. As generative AI models such as GPT (Generative Pre-trained Transformer) advance, they provide powerful reasoning and interaction capabilities. Delivering timely results from resource-intensive models, however, requires a communication layer that accommodates both local and remote (cloud) decision-making. For example, a robot may query a local model to make a quick go/stop decision while sending richer context to a cloud model for long-horizon planning.

This paper proposes a Layer Communication Framework (LCF) that addresses these realities using a flexible, multi-layer design. LCF distinguishes between a first layer that targets near-instant responses (e.g., local AI or edge decision units) and a second layer for more intensive AI computation (e.g., large cloud-based models). Communication and orchestration mechanisms are tuned so that Layer 1 can consistently operate within soft real-time bounds, while Layer 2 is allowed higher latency but returns more sophisticated responses that refine or override earlier decisions.

Building on programmable and parallel networking, LCF enhances responsiveness and reliability across heterogeneous networks. The framework is protocol-agnostic and can coexist with TCP or UDP, overlaying legacy infrastructure without disruptive changes. The rest of this article is organized as follows: Section II discusses related work; Section III introduces LCF and its relation to CoopNet; Section IV presents delay, throughput, and reliability formulations; Section V details the experimental setup and results, including qualitative comparison; Section VI briefly considers security and deployment; and Section VII concludes.

II. Literature Review

Artificial Intelligence (AI) has progressed from foundational theories [1], [2] to systems reshaping multiple industries [3]. Large Language Models (LLMs) such as GPT [4]–[6] can dynamically invoke specialized sub-models, and have shown

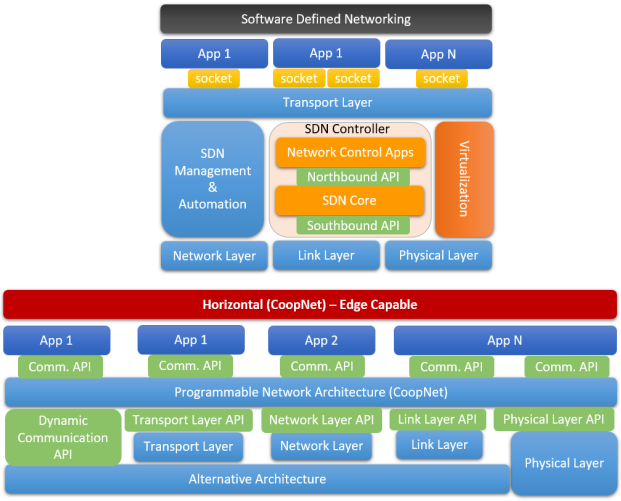


Fig. 1: SDN (top) vs. horizontal programmable networking CoopNet (bottom). Intermediate nodes participate in forwarding and processing.

effectiveness from creative generation to embedded deployments [7], [8]. Their integration into time-sensitive scenarios can improve decision-making and efficiency [9], [10].

Cloud-hosted LLMs handle extensive computation [11], [12], but near-instant responses are vital in domains like robotics and drone navigation [9], [10]. Cloud latencies often remain around two seconds [9], which is inadequate for sub-second control loops. As a result, research has shifted toward edge-based or hybrid architectures [13]–[15] combining localized AI for immediate actions with remote servers for deeper analysis.

Parallel advances in networking—such as programmable control [16] and multi-interface communications [17]—seek to reduce congestion and improve real-time performance. Partitioning AI tasks across localized nodes and cloud back ends addresses growing requirements for speed and scalability.

Figure 1 contrasts a centralized SDN model with a horizontal approach typified by CoopNet [15], where intermediate nodes can adaptively forward or process data for low-latency, parallel transmissions. As shown in Figure 2, real-time AI systems frequently exhibit dynamic topologies; local AI models may handle immediate tasks while cloud models provide deeper analysis, with roles evolving over time.

No single model suits every domain [18], so fine-tuned or domain-specific solutions [8] often coexist with broader LLMs [4], [19]. This diversity demands robust, tiered communication strategies uniting edge devices and large-scale clouds. Many existing frameworks focus on isolated metrics such as throughput or reliability, while CoopNet [15] illustrates benefits of in-network processing. CoopNet, however, primarily addresses dynamic link establishment and multipath routing; it does not provide explicit layered AI semantics nor integrated delay/throughput/reliability modeling.

LCF responds to this gap by providing a unified, layered framework that connects edge, intermediate nodes, and cloud, with explicit support for soft real-time AI coordination and analytical performance modeling. Rather than compete with fieldbus or deterministic Ethernet technologies that target sub-millisecond motion control, LCF is intended to sit above those mechanisms and orchestrate higher-level AI services that tolerate tens to hundreds of milliseconds but demand flexibility, heterogeneity, and the ability to incorporate new models and tools over time. In this sense, LCF complements lower-level real-time networks by handling the AI-intensive part of the stack that traditional control-oriented protocols were never designed to support.

III. Proposed Solution: Layer Communication Framework (LCF)

Building on earlier work on multipath parallel communication for AI-driven robotic control, we separated AI queries from routine commands and used multiple interfaces to prioritize critical tasks. A dual-model strategy—a "turbo" model for rapid responses and a more thorough model for optimization—allowed fast immediate actions and slower refinement. A buffering mechanism queued refined instructions for later execution. Nonetheless, local AI outputs were still underutilized and request separation was imperfect, motivating the development of LCF.

A. System Context and Motivating Use Case

In a typical deployment, LCF operates between an AI-enabled real-time system (RTS) and a collection of heterogeneous processing endpoints. The RTS generates requests such as sensor summaries, intent descriptions, or safety constraints. These requests must reach both a fast, local decision module (Layer 1) and a more capable but slower AI engine (Layer 2), which might reside in the cloud or a nearby data center.

When a mobile robot detects an obstacle and must choose between slowing down, rerouting, or negotiating passage with nearby robots, LCF sends a compact state description to a local policy model at Layer 1 and, in parallel, to a large-scale model at Layer 2 that considers longer-term objectives and fleet-wide optimization. The Layer 1 response is expected

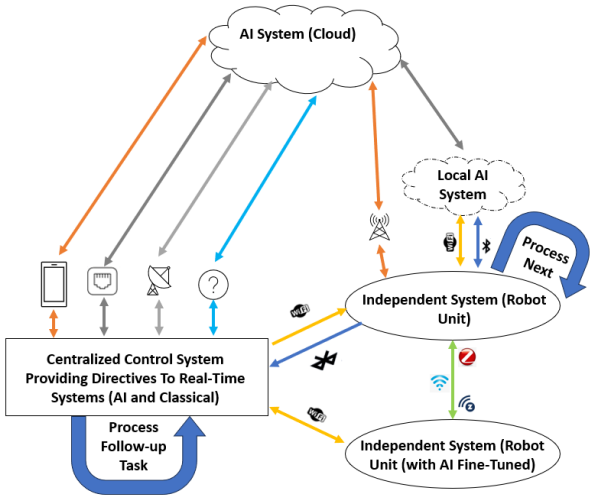


Fig. 2: Topology dynamics for a real-time AI application combining edge and cloud intelligence.

within roughly 100 ms, providing an immediate, safe action, while the Layer 2 response may arrive in about 1.2 s and refine subsequent behavior. This illustrates how soft real-time communication with AI models supports high-level decision-making without assuming millisecond-scale LLM responses.

B. Framework Overview and Relation to CoopNet

LCF integrates simultaneous multi-node messaging, dynamic data routing via CoopNet, and decentralized data handling to optimize responsiveness in real-time systems. Most devices rely on TCP sockets (often with TLS) for cloud-based communication, while edge systems (e.g., drones, IoT devices) use similar protocols or domain-specific options such as Modbus TCP. These layered and sometimes legacy approaches can become problematic in delay-sensitive scenarios.

To address this, we use a bio-inspired messaging model that minimizes overhead, remains protocol-agnostic, and supports parallel transmissions across diverse interfaces. As illustrated in Algorithm 1, LCF combines programmable networking APIs with ephemeral links created by CoopNet to coordinate multiple nodes at once. CoopNet provides the substrate for dynamic links, while LCF defines layered semantics and AI-driven orchestration on top.

LCF is particularly beneficial for applications requiring real-time data processing and decision-making. A single message can be delivered concurrently to multiple systems (e.g., local and cloud AI), avoiding sequential transmission delays. Figure 3 shows Nodes L1 and LN receiving and replying to the same message, while Figure 4 depicts forwarding and pass-through via intermediary nodes. In LCF, a unique block of identifiers recognizes receivers, and need-centric identifiers (akin to data-centric naming) are left for future work.

C. Orchestration Algorithm and Function Definitions

Algorithm 1 summarizes LCF orchestration. The system collects node state (e.g., queues, reliability, delay), assigns requests to nodes, and then uses single or multiple paths for transmission. Responses are processed such that the first usable one is executed immediately, while later responses refine prior decisions.

The key functions are:

- `ComputeMatching`: pairs requests with nodes according to a cost function w (Section IV).
- `DefaultAssign`: provides a simpler policy when full matching is unnecessary.
- `SelectParallelLinks`: queries CoopNet for candidate paths and selects a subset.
- `SendRequestOverPath`/ `SendRequestOverSingleLink`: encapsulate underlying packetization and transport.
- `AwaitResponse`, `ExecuteImmediateAction`, and `RefineDecision`: manage response timing and use.

Algorithm 1: LCF Orchestrator with Matching-Based Resource Assignment

Require: **Inputs:**
- $\mathcal{R}$: Set of requests (sensor, inference, etc.)
- $\mathcal{N}$: Nodes (Layer 1, Layer 2, etc.) with reliability and queue states
- *ParallelMode*: Enable/disable multipath transmissions
- *MatchingEnabled*: Use matching or scheduling

Ensure: Real-time responses, efficient data routing

Gather Node States (queues, reliability, delays).
if *MatchingEnabled* **then**
  pairs ← ComputeMatching($\mathcal{R}, \mathcal{N}, w$)
  AssignedRequests ← map each request to its matched node in *pairs*
else
  AssignedRequests ← DefaultAssign($\mathcal{R}, \mathcal{N}$)
end if
for all *(req, node)* in *AssignedRequests* **do**
  if *ParallelMode* **then**
    paths ← SelectParallelLinks($req, node$)
    for all *path* in *paths* **do**
      SendRequestOverPath(req, path) {Use CoopNet to instantiate path}
    end for
  else
    SendRequestOverSingleLink(req,node)
  end if
end for
for all *(req,node)* in *AssignedRequests* **do**
  resp ← AwaitResponse($req, node$)
  if *IsFirstResponse(resp)* **then**
    ExecuteImmediateAction(resp) {Soft real-time decision}
  else
    RefineDecision(resp) {Update internal plan or future actions}
  end if
end for
PurgeExpired(*staleResponses*)

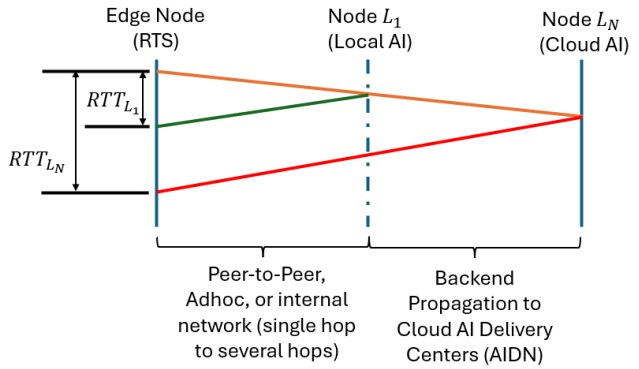


Fig. 3: Layered Communication Framework: RTS at the edge interacts concurrently with Layer 1 and Layer 2 AI components.

LCF leverages CoopNet for efficient management of network resources, enabling simultaneous data requests to specific nodes, distinct from general broadcasting. Responses are integrated into the decision-making process, allowing nodes to adapt based on real-time data from multiple sources and improving robustness in complex environments such as robotic navigation.

D. Delay Modeling for Layer Communication Framework

To represent LCF performance, we develop models for delay, throughput, and reliability.

The end-to-end delay D_{e2e} is:

$$D_{\text{e2e}} = D_{\text{proc}} + D_{\text{prop}} + D_{\text{trans}} + D_{\text{queue}} \tag{1}$$

where D_{proc} is processing delay, D_{prop} propagation delay, D_{trans} transmission delay, and D_{queue} queuing delay.

Processing Delay (D_{proc}):

$$D_{\text{proc}} = D_{\text{txrx}} + \sum_{i=1}^{k} \frac{\Gamma + \Gamma_{\text{TTL}} + \Gamma_{\text{HR}} + \Gamma_{\text{DF}} + \Gamma_{\text{IO}}}{f} \quad (2)$$

where D_{txrx} is sender/receiver processing, Γ terms are CPU cycles for specific operations, f is CPU frequency, and k is hop count.

Propagation, Transmission, and Queuing Delays:

$$D_{\text{prop}} = \sum_{i=1}^{k} \frac{d_i}{s_i},\ D_{\text{trans}} = \sum_{i=1}^{k} \frac{L_i}{R_i},\ D_{\text{queue}} = \sum_{i=1}^{k} \frac{\lambda_i}{\mu_i - \lambda_i} \quad (3)$$

with d_i and s_i denoting distance and propagation speed, L_i and R_i packet length and bandwidth, and λ_i, μ_i arrival and service rates.

Although some components may be small, LCF exposes intermediate nodes as virtual networking elements, making processing and queuing effects more prominent. We therefore incorporate hop-dependent factors $\beta(h)$, $\phi(h)$, and $\gamma(h)$:

Propagation Delay Factor ($\beta(h)$):

$$\beta(h) = \frac{1}{2}\left(\tanh\left(\kappa(h - h_0)\right) + 1\right) \quad (4)$$

Transmission Delay Factor ($\phi(h)$):

$$\phi(h) = 1 - e^{-\tau h} + \epsilon(h), \quad \epsilon(h) \sim \mathcal{N}(0, \sigma^2_{\phi(h)}) \quad (5)$$

Queuing Delay Factor ($\gamma(h)$):

$$\gamma(h) \sim \mathcal{N}(1 + \alpha h, \zeta h) \quad (6)$$

Total Delay in LCF (D_{LCF}):

$$D_{\text{LCF}} = D_{\text{proc}} + \phi(h) \cdot D_{\text{trans}} + \beta(h) \cdot D_{\text{prop}} + \gamma(h) \cdot D_{\text{queue}} \quad (7)$$

LCF can satisfy soft real-time requirements even when full end-to-end delay to distant nodes is large, because responses may be generated at earlier nodes.

E. Round Trip Time (RTT)

The Round Trip Time (RTT) to node L is:

$$RTT_L = 2 \cdot (D_{\text{prop},L} + D_{\text{trans},L} + D_{\text{proc},L}) \quad (8)$$

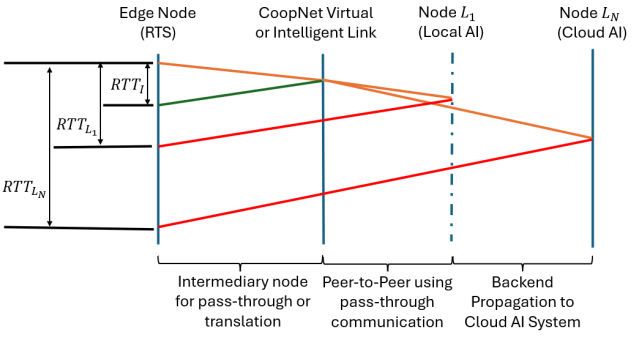


Fig. 4: Intermediary virtual link with LCF using CoopNet-managed paths for parallel and pass-through communication.

For layered exchanges, the Complete Transaction Delay for layer L is:

$$CTD_L = D_{\text{request\_e2e}} + \sum_{i=1}^{m} D_{\text{response\_e2e},L_i} + \sum_{i=1}^{m} D_{\text{misc},L_i} \quad (9)$$

where m is the number of message parts and D_{misc,L_i} captures acknowledgments or protocol overhead.

F. Throughput Modeling

Throughput T can be modeled as:

$$T = \frac{C}{D_{\text{LCF}}} \quad (10)$$

or, explicitly including the delay factors:

$$T = \frac{C}{1 + \phi(h) \cdot D_{\text{trans}} + \beta(h) \cdot D_{\text{prop}} + \gamma(h) \cdot D_{\text{queue}}} \quad (11)$$

As delays increase, throughput decreases. In LCF, parallel paths and matching-based assignment help keep D_{LCF} bounded for Layer 1 paths that must meet soft real-time constraints.

IV. Reliability Analysis for LCF

Transforming standard nodes into active networking nodes necessitates a reliability model that captures stochastic behavior.

A. Probabilistic Reliability at Each Node

Node reliability p_i is modeled as:

$$p_i = \eta \exp(-\Psi x_i) \quad (12)$$

where η is baseline reliability and Ψ controls decay with load or stress x_i.

B. Overall Network Reliability

Overall reliability R along a path of n nodes is:

$$R = \prod_{i=1}^{n} p_i \quad (13)$$

emphasizing how each node's reliability affects end-to-end behavior when nodes perform networking functions under LCF.

C. Extended Modeling: Matching-Based Resource Assignment

Beyond delay and reliability, LCF must allocate tasks across nodes efficiently.

1) Bipartite Matching Formulation

We model this as a bipartite graph:

$$G = (U, V, E),$$

where U is the set of requests and V the set of nodes. An edge $(u, v) \in E$ indicates that v can serve u, with cost $w(u, v)$ encoding communication delay, reliability, or capacity.

The objective is:

$$\sum_{(u,v) \in M} w(u, v) \to \min \quad (14)$$

subject to M being a matching. For example:

$$w(u, v) = \psi \cdot D_{\text{LCF}}(u, v) \; - \; \varphi \cdot p_v,$$

with ψ, φ weighting delay and reliability.

2) Modeling Considerations

The Hungarian Method [20] yields optimal matchings for moderate sizes, while larger systems may rely on heuristics or approximate solvers. By integrating LCF's delay and reliability models into $w(u, v)$, resource assignment becomes aligned with soft real-time constraints and robustness goals.

3) Dynamic Queueing Models

Queue dynamics can be captured as:

$$Q_i[t+1] \; = \; \max\big(Q_i[t] - \mu_i[t],\, 0\big) \; + \; \lambda_i[t],$$

where $Q_i[t]$ is backlog, $\lambda_i[t]$ arrivals, and $\mu_i[t]$ service. Lyapunov drift techniques [21] can then guide scheduling policies that keep queues stable and delays bounded.

4) Multi-Objective Optimization

When trade-offs are needed, a multi-objective function can be used:

$$\text{Objective} = \theta\, D_{\text{LCF}} \; + \; \delta\,(1 - R) \; + \; \xi\, \Pi,$$

where θ, δ, ξ weight delay, reliability, and a load/energy metric Π.

5) MDP-Based Dynamic Routing or Scheduling

With stochastic link states and arrivals, task assignment can be modeled as an MDP. States include queue lengths and link reliabilities; actions choose paths or nodes; rewards reflect latency and reliability. Standard methods (e.g., value iteration or Q-learning) can produce adaptive policies [22].

6) Parallelization Overhead Modeling

Parallel transmissions introduce coordination overhead. If $\Omega(\#\text{Paths})$ denotes this overhead:

$$D_{\text{e2e}}^{(\text{parallel})} = D_{\text{e2e}} + \Omega(\#\text{Paths}),$$

capturing diminishing returns when too many paths are used simultaneously.

V. Implementation and Results

To evaluate LCF, we used NS3 v3.41 with multiple interfaces, including edge Wi-Fi under the FriisPropagationLossModel. The linear topology (Figure 6) comprised seven nodes at 50 m intervals: Node 0 as the Real-Time System (RTS), Node 1 a programmable networking node implementing prior work, Node 2 as Layer 1, Nodes 3–5 as Internet components, and Node 6 as Layer 2. Average processing times were derived from real GPT systems [9]. This configuration emulates a robotics scenario where the RTS communicates with a nearby edge node and a more distant cloud node through intermediate routers.

Figure 5 summarizes delay, throughput, and message reception. Layer 1 demonstrated near 100 ms latency at the RTS under the considered load, suitable for soft real-time tasks such as coordination, route selection, and safety-envelope decisions. Layer 2 incurred a 1.2 s reception delay due to dual-reception overhead and more complex processing for remote AI components. This higher latency is acceptable for background refinement or long-horizon planning but not for immediate obstacle avoidance.

LCF's throughput largely paralleled conventional frameworks, but reception rate improved, especially when Layer 1 actively handled local AI tasks. Horizontal programmability mitigated packet loss via parallel routes: when a link degraded, CoopNet instantiated alternative paths and LCF shifted traffic without RTS-level reconfiguration. A simple matching-based method (Section III) hinted at further gains by routing requests to underutilized nodes. LCF maintained consistent performance under varying loads, important for edge scenarios where bursts of data arise, and reduced single-path congestion by splitting traffic across interfaces.

Overall, these results show that LCF's multipath strategies and integration with CoopNet address latency demands in soft real-time AI scenarios. The improved delay, throughput, and reception rates highlight its potential for robotics, autonomous vehicles, drones, and other mission-critical applications where hard real-time control and soft real-time AI coordination are separated.

A. Real-Time Latency Regimes and 10 ms Constraints

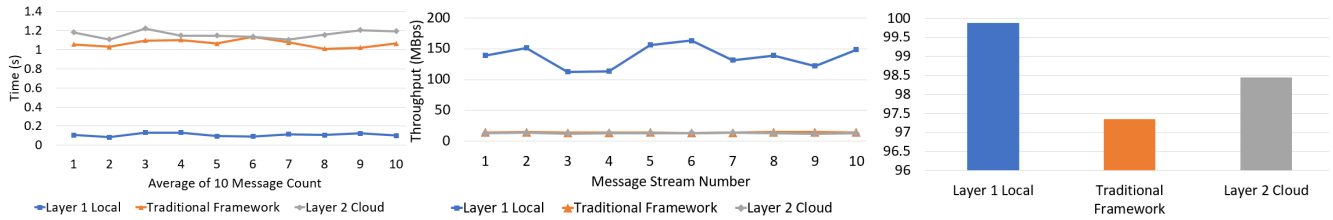
The notion of "real-time" can refer to very different latency regimes. Many industrial control applications require cycle times in the 1–10 ms range for inner feedback loops. These loops are typically implemented on dedicated hardware co-located with sensors and actuators, with only minimal communication and virtually no complex inference in the loop. In contrast, LCF is designed to orchestrate AI workloads that may involve natural language or high-level planning, where each request often triggers an LLM completion or a sequence of tool calls.

Under favorable wireless or wired conditions, the *communication-only* component between the RTS and a nearby Layer 1 node can be driven into the tens-of-milliseconds range via appropriate modulation schemes, link-layer optimizations, and short-distance paths. However, the total end-to-end latency for a full request–inference–response cycle also includes:

- the time to serialize the request into an LLM- or tool-compatible format;
- the actual inference time at the edge or cloud model; and
- the time to post-process and apply the result.

With current model sizes and hardware, these processing components dominate the budget, making a complete 10 ms cycle unrealistic when an LLM-style completion is in the loop, even if pure communication delay is substantially lower.

LCF therefore intentionally targets the 50–150 ms regime for soft real-time coordination tasks, while leaving sub-10 ms hard real-time control to local controllers or microcontrollers that do not depend on remote inference. Ongoing work explores combining LCF with compressed or distilled edge



(a) Delay (b) Throughput (c) Message Reception

Fig. 5: Network performance metrics under LCF. (a) End-to-end delay observed at the RTS for Layer 1 and Layer 2; (b) throughput evolution; and (c) message reception ratios.

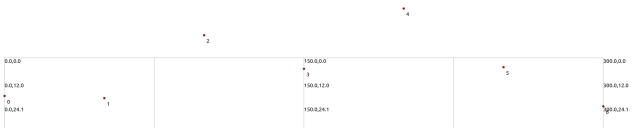


Fig. 6: NS3 network animator topology used for LCF evaluation. Nodes are spaced 50 m apart between the RTS and Layer 2.

TABLE I: Qualitative Comparison of LCF with Representative Frameworks

Feature	TCP/IP	MQTT-like	CoopNet	LCF
AI-oriented layered semantics (L1/L2)	No	No	No	Yes
Built-in multipath orchestration	Limited	No	Yes	Yes
Integration of delay/throughput models	No	No	Partial	Yes
Reliability modeling at node level	No	No	Partial	Yes
Matching-based request assignment	No	No	No	Yes
Protocol-agnostic operation	Partial	Partial	Yes	Yes
Soft real-time AI coordination focus	No	Limited	Partial	Yes

models, quantized LLMs, and tighter integration with Model Context Protocol (MCP)-style predefined tools. By shortening prompts, standardizing tool schemas, and executing more logic locally at the edge, these approaches aim to reduce both communication and processing overhead, gradually pushing effective response times closer to the lower end of the soft real-time range without over-promising 10 ms end-to-end AI responses.

B. Comparative Analysis with Existing Frameworks

To position LCF among existing solutions, Table I qualitatively compares it with representative baselines. TCP/IP provides reliable end-to-end transport but no AI-oriented layers or explicit multipath orchestration. Message brokers such as MQTT-style publish/subscribe systems offer flexible topics but typically lack link-level parallelism and formal performance models. CoopNet [15] supports in-network cooperation and dynamic virtual links, yet does not define layered AI semantics or matching-based request assignment.

LCF does not replace the underlying transport protocols. Instead, it layers AI-centric orchestration on top, introducing explicit layered semantics, integrated performance modeling, and AI-aware assignment that are absent in typical stacks.

C. Integration with Tool-Based Orchestration (MCP)

LCF can also accelerate workflows that rely on tool-based orchestration frameworks, such as recent Model Context Protocol (MCP) designs where tools (e.g., perception, control, logging) are predefined and registered with schemas. In this setting, each tool becomes a known endpoint or service in the LCF layer graph. Because the tools and their arguments are predefined, LCF does not need to negotiate capabilities at runtime and can directly route requests to the appropriate Layer 1 or Layer 2 nodes.

For example, a robot may expose a “navigate,” “inspect,” or “log-event” tool through MCP. LCF can map each tool call to a specific node or path, using matching-based assignment to choose low-latency or high-reliability nodes, and use parallel links when necessary. The combination of predefined tools (via MCP) and LCF’s communication orchestration reduces overhead and shortens effective response times, particularly for repeated or predictable call patterns. In multi-robot scenarios, the same tool catalog can be shared across a fleet, while LCF decides in real time whether a given request is best served locally (on-board or at a nearby edge node) or remotely (at a cloud or data center node), effectively turning tools into a distributed service fabric anchored by the communication layer.

D. Discussion and Limitations

The current evaluation relies on NS3 simulations. This allows controlled variation of link characteristics and repeatable experiments, but does not fully capture real-world wireless interference or hardware-specific timing. Absolute latency values may therefore differ from those in a physical deployment.

Our goal is to show that:

1) Under a realistic model, LCF can achieve soft real-time latency for Layer 1 tasks while supporting higher-latency Layer 2 processing; and
2) Structural features such as multipath transmission and matching-based assignment translate into improved delay and reception compared to a single-path baseline.

Future work will deploy LCF on a physical robotic testbed and study the impact of hardware acceleration for both communication and AI processing. Additional research directions include quantifying fairness across multiple competing RTS

instances, exploring energy–latency trade-offs when choosing between edge and cloud inference, and evaluating LCF in heterogeneous settings where devices range from low-power microcontrollers to GPU-equipped edge servers. These experiments will help determine how the framework behaves under mixed workloads and how its parameters should be tuned for different application domains (e.g., logistics robots, assistive home robots, or industrial inspection).

VI. Security and Deployment Considerations

While LCF focuses on performance and reliability, security is critical for deployment. Multi-node, multi-path transmissions enlarge the attack surface and require confidentiality, integrity, and authentication.

LCF can incorporate common measures with limited changes to the core model: nodes carry cryptographic credentials that are verified before establishing CoopNet-managed paths; application- or transport-layer encryption (e.g., TLS/DTLS) protects each parallel path; nonces or sequence numbers prevent replay; and matching policies can enforce security labels (e.g., trusted edge vs. semi-trusted cloud). These mechanisms add processing overhead that is captured in D_{proc} but do not alter the overall framework.

VII. Conclusion

This article introduced the Layer Communication Framework (LCF), which extends existing programmable networking approaches by enabling dynamic, simultaneous data exchanges for AI-driven, real-time systems. LCF integrates multipath parallel communication, a dual-layer structure, buffering mechanisms, and a horizontal architecture to provide reliable, low-latency operation in scenarios such as robotic control and distributed perception.

A key contribution is the explicit separation between soft real-time coordination tasks and higher-latency, compute-intensive AI processing. By keeping Layer 1 within approximately 100 ms while allowing Layer 2 to perform more sophisticated reasoning over longer time scales, LCF aligns communication behavior with practical requirements in robotics and edge AI. Integrated delay, throughput, and reliability models, together with matching-based resource assignment, provide a basis for understanding and tuning this behavior. LCF also complements emerging tool-based orchestration frameworks (e.g., MCP) by offering a communication substrate that can quickly route predefined tool calls to appropriate nodes and by making it easier to deploy new AI capabilities without redesigning the network.

Future work will refine context-aware AI interactions, extend buffering and queuing strategies, and evaluate hardware acceleration and security mechanisms in physical deployments. In the longer term, we envision LCF as part of a broader ecosystem in which real-time communication, AI inference, and tool-based orchestration are co-designed, enabling intelligent spaces, collaborative robot fleets, and cyber-physical systems that can adapt their behavior in real time to evolving conditions and user needs. The framework's flexibility across edge, fog, and cloud infrastructures positions LCF as a foundation for next-generation AI communication systems.

References

[1] F. De Saussure, *Cours de linguistique générale*. Otto Harrassowitz Verlag, 1989, vol. 1.

[2] N. Chomsky, *Syntactic structures*. Mouton de Gruyter, 2002.

[3] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 604–624, 2020.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, 2020.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, vol. 30, 2017.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[8] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," 2023.

[9] R. C. Voicu, A. K. Pande, M. H. Tanveer, and Y. Chang, "Communication interchange for artificial intelligence systems," in *2024 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, Accepted.

[10] A. K. Pande, P. Brantley, M. H. Tanveer, and R. C. Voicu, "From ai to agi: The evolution of real-time systems with gpt integration," *IEEE SouthEast Conference 2024*, 2024.

[11] C. Hsu, C. Fannjiang, and J. Listgarten, "Generative models for protein structures and sequences," *nature biotechnology*, vol. 42, no. 2, pp. 196–199, 2024.

[12] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, "Everybody dance now," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5933–5942.

[13] R. C. Voicu, S. Steele, J. D. Rodriguez, Y. Chang, and C. Ham, "Advanced biomedical laboratory (abl) synergy with communication, robotics, and iot," in *SoutheastCon 2023*. IEEE, 2023, pp. 590–595.

[14] R. C. Voicu, J. A. Copeland, and Y. Chang, "Multiple path infrastructure-less networks a cooperative approach," in *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019, pp. 835–841.

[15] R. C. Voicu and Y. Chang, "Cooperative networking using passive discovery for dynamic environments," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2022, pp. 1279–1284.

[16] ——, "Towards programmable networking with coopnet: A horizontal parallel multipath approach," in *2023 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2023, pp. 111–116.

[17] ——, "Stages of coopnet: A multipath parallel link architecture for next-gen networks," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2021, pp. 592–597.

[18] K. N. Qureshi and T. Newe, *Artificial Intelligence of Things (AIoT): New Standards, Technologies and Communication Systems*. CRC Press, 2024.

[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[20] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[21] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.

[22] C. Wang, S. Lei, P. Ju, C. Chen, C. Peng, and Y. Hou, "Mdp-based distribution network reconfiguration with renewable distributed generation: Approximate dynamic programming approach," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3620–3631, 2020.