Nigerian Journal of Technology (NIJOTECH)
SI-2026A: Advances in Modelling, Simulation and AI/ML for Multi-Disciplinary Engineering Applications
2026. pp. 64- 75
www.nijotech.com

Print ISSN: 0331-8443
Electronic ISSN: 2467-8821
https://dx.doi.org/10.4314/njt.v45i1.5s

MESHMIND: DECENTRALIZED LAN-BASED COORDINATION FOR LARGE LANGUAGE MODELS: DESIGN, IMPLEMENTATION, AND EVALUATION

AUTHORS:
*V. U. Rathod[1], A. D. Londhe[2], S. Y. Bobade[3], A. Jagtap[4], S. Dhamdhere[5] and V. C. Todkari[6]

AFFILIATIONS:
[1]Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), Vishwakarma Institute of Technology, Pune, Maharashtra, INDIA
[2]Department of Artificial Intelligence and Data Science, Vishwakarma Institute of Technology, Pune, Maharashtra, INDIA
[3]Department of Computer Science and Engineering (Artificial Intelligence), Vishwakarma Institute of Technology, Pune, Maharashtra, INDIA
[4]Department of Computer Engineering, Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati, Pune, Maharashtra, INDIA
[5]Department of Computer Engineering, Marathwada Mitramandal's Institute of Technology, Lohgaon, Pune, Maharashtra- 411047, INDIA
[6]Department of Mechanical Engineering, Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati, Pune, Maharashtra, INDIA

***CORRESPONDING AUTHOR:**
Email: vijay.rathod25bel@gmail.com

ARTICLE HISTORY:
Received: December 01, 2025.
Revised: February 14, 2026.
Accepted: February 20, 2026.
Published: May 4, 2026.

KEYWORDS:
Large Language Models (LLMs), Edge Intelligence, Distributed Computing Framework, Collaborative Model Processing, Decentralized Artificial Intelligence Systems, Peer-to-Peer Networking.

ARTICLE INCLUDES:
Peer review

DATA AVAILABILITY:
On request from author(s)

EDITORS:
Sagar Shelare
Sameer Sheshrao Gajghate

FUNDING:
None

HOW TO CITE:
Rathod, V. U., Londhe, A. D., Bobade S. Y., Jagtap, A., Dhamdhere, S. and Todkari, V. C. " Meshmind: Decentralized Lan-based Coordination for Large Language Models: Design, Implementation, and Evaluation", *Nigerian Journal of Technology*, SI-2026A: Advances in Modelling, Simulation and AI/ML for Multi-Disciplinary Engineering Applications 2026. pp. 64- 75. https://dx.doi.org/10.4314/njt.v45i1.5s

Abstract
Big developments in large language Models (LLMs) have increased usage of smart (intelligent) apps but have also made it more dependent on cloud infrastructures that have limitations in privacy, latency, costs and network dependence. This paper will propose a new P2P Framework called "MeshMind". MeshMind is a fully decentralized peer-to-peer (P2P) URL Application that provides secure, collaborative access to the LLMs for the purpose of providing extensive access to computational resources or services within LANs. MeshMind allows users to collaboratively utilize multiple heterogeneous devices and enables a distributed architecture for LLM processing. This paper details how a hybrid communication structure incorporates UDP for automatic peer discovery and TCP for establishing reliable synchronized communications, transferring files and updating capability status. Finally, MeshMind employs a multilayered security mechanism, including the use of HMAC- signed protocol messages and JWT-based session control. Ollama-enabled devices can act as hosts for LLMs, while lighter weight devices, referred to as peers, and may utilize authenticated delegation to offload inference requests according to workload heuristics that adapt based on real-time conditions. Experimental evaluation conducted using a multi-device local area network (LAN) test bed showed that MeshMind successfully discovered peers, synchronized devices reliably, and distributed inference processing in an ultimately scalable architecture, resulting in sub-3 second latencies as the network(s) increased in size. The throughput improved significantly with the increasing number of peers while resource usage was evenly distributed across all peer devices by intelligently distributing workload, resulting in significantly improved processing capabilities without increasing costs associated with using additional hardware resources. Mathematical simulations of MeshMind performance confirmed the overall efficiency, high level of network stability, and ability to achieve a near-linear scale within the distributed architecture

1.0 INTRODUCTION
The rapid progress of language model-based AI has improved the abilities of many modern AI systems to conduct highly complex reasoning tasks using sophisticated language crafting and use of language to create contextually relevant writings across many application areas to an extent that had not previously been possible. However, while these recent developments represent vast progress towards creating ever more advanced LLM-based AI

technologies, the reliance on centralized and distributed cloud computing infrastructures remains a significant disadvantage. Specifically, for many edge computing devices and environments where internet connectivity is restricted/limited or where privacy will restrict access to LLM-based AI services, the large computing cost of using LLM-based AI tools, added risk of exposing data/information, and inability to rely on continued availability of the LLM cloud-based service infrastructure makes using LLM-based AI systems very difficult or impossible. Hence, at this point in time, it appears that there will continue to be a large barrier to LLM-based AI technology being utilized within resource- and/or privacy-sensitive scenarios; thereby indicating that there is a need for the creation of new decentralized, local AI architectures [1], [2].

Prior research has placed more focus on implementing federated and edge-based learning paradigms to alleviate these issues by retaining data on local devices while distributing computational workloads amongst participating nodes. For example, [3] has shown that such architecture reduces latency and energy consumption by placing inference and learning processes nearest end users. In addition, [4, 5] indicates that supported by Efficient Scheduling and Collaborative Processing Strategies, the potential for Coordinated Edge-Based Inference to achieve the same performance as Current Cloud Infrastructures is real.

Even so, many current solutions rely on a centralized host, or cloud-based aggregator, which opens these solutions to a single point of failure as well as limits their capacity for device independent functionality. As such, many of these architectures are devoid of real time peer discovery, efficient local resource sharing and seamless ability to operate offline; thus, limiting their potential use in privacy-sensitive or low-resource environments. Furthermore, achieving a proper balance of privacy preservation, communication overhead and computation load remains a challenge for any fully decentralized system that would coordinate without using trusted central entities [6].

MeshMind solves all these issues with our decentralized P2P architecture. We will establish a P2P framework where multiple devices can collaborate on the execution of LLMs in their own local environment by sharing and/or transferring files from one node to another or others and securely invoking execution results from an LLM. Each device will communicate using both hybrid (TCP and UDP) communication protocols to discover peers and provide state synchronization and secure file transfers. Security of the P2P network and individual node(s) will be enhanced using HMAC-based authentication of the network nodes. Only devices that successfully authenticate will be permitted to participate in the P2P network. The peer-to-peer architecture of the MeshMind provides a means for each device to either function as an LLM host where it can run an LLM using the Ollama for on-device model execution or as a lightweight client that securely delegates inference tasks to peers that can execute those tasks. This will enable resilient, privacy-preserving, resource efficient and efficient collaboration among many different types of commodity hardware for an LLM.

This work offers the subsequent key contributions

- A fully decentralized peer-to-peer framework that facilitates collaborative LLM inference and secure local file and resource sharing without reliance on any central coordinating entity.

- A hybrid UDP/TCP communication architecture that enables autonomous peer discovery, reliable state synchronization, and consistent mesh formation across heterogeneous devices.

- Privacy-preserving execution and authenticated task delegation mechanisms, supporting both on-device inference and secure offloading to trusted peers while maintaining the integrity of collaborative processing.

- A comprehensive empirical evaluation, supported by mathematical modeling, which quantifies system performance in terms of latency, throughput, resource utilization, and scalability within local network environments.

2.0 RELATED WORKS

The increasing shift of machine learning towards decentralized and collaborative approaches that do not rely on cloud architecture for operation, cloud-based LLM pipelines typically result in high operational costs, potential for data leakage, and high latency; these issues are exacerbated if the user has limited or intermittent connectivity [7]. The use of edge computing and federated learning reduces some of these impacts by moving the computation into the end-user's device and providing increased responsiveness to users, reducing the amount of sensitive data available for potential misuse believe that creating resilient, trusted, and governed distributed AI systems is important for the success of these systems, has shown that coordinated edge-

based inference can provide an equivalent level of performance to that of cloud systems, provided that efficient task scheduling and collaboration are in place [8].

Although novel technologies have evolved to retain the centralized aggregation server as an experimental component of peer-to-peer artificial intelligence, many other currently available peer-to-peer artificial intelligence technologies still relies on a centralized server to collect and aggregate data [9]. These servers create workflow dependencies, bottlenecks, and interfere with the independent operation of each peer. Recent investigations into this concept have examined the development of peer-to-peer controllers and controllers without controllers to provide a framework for peer-to-peer intelligence that removes the need for central orchestration [10]. Initial theoretical research conducted by van der Schaar Lab describes the fundamental principles of collaboration, incentive design, and stability in P2P Networks [11]. In addition, multi-LLM orchestration and Distributed Inference have been the focus of some research efforts, with the goal of increasing the efficiency of distributed processing through the investigation of routing methodologies, model partitioning strategies, and adaptive scheduling techniques. Additionally, Parallax created a decentralized LLM Service Architecture to facilitate Quality-Inference using low-latency communication and Dynamic Scheduling between nodes [12, 13].

The current body of research on localized mesh-based architecture lacks adequate practical examples of autonomous peer discovery and secure file transfer, as well as the ability to execute collaborative LLMs without relying on the internet. This indicates the need for appropriately designed, lightweight, self-sustaining systems that can perform independently from an internet connection and operate completely within a local area or similarly confined environment. MeshMind has created a solution to address this requirement by creating an eco-friendly and light-weight means of integrating the items above into a single local-first platform with resource-efficient capabilities, a backend focused on supporting high-performance computing using Rust (Actix), the use of UDP and TCP for peer discovery and collaborative synchronization purposes, and Ollama technology on the device itself for performing on-device LLM inference, thus allowing a seamless decentralized way of working together on LLMs [14].

To better understand the niche MeshMind fits within the larger environment of decentralized and distributed AI systems, a comparative table (Table I) has been created to compare MeshMind with other already established Distributed AI frameworks (Parallax, Petals, Hive Mind, Ray Serve, Microsoft SWARM). As can be seen in the table, there is considerable variation between the previously established Distributed AI Frameworks in terms of architectures used, development environments, routing models, how trust and security are being built, and the hardware the framework will depend on. This comparison shows how MeshMind offers an alternative to the other Distributed AI frameworks through its local-area only capability, privacy and security, as well as enabling decentralized collaborative LLM resource sharing without the need for being connected to a wider-area network or being dependent upon centralized coordination [15].

Table 1: The comparison of MeshMind with existing decentralized and distributed AI systems

Feature	MeshMind	Parallax	Petals	Hive mind	Ray Serve	Microsoft SWARM
Deployment Model	LAN-only, fully offline	WAN / multi-network clusters	Global volunteer network	Decentralized P2P training	Distributed Cloud or cluster	Agent-based coordination Layer
Primary Goal	Private LAN LLM collaboration	Scalable decentralized inference	Crowd sourced LLM inference	Decentralized optimizer sync	High throughput model serving	Multi-agent distributed workflows
Peer Discovery	Automatic UDP broadcast	Known node registry or routing tables	Server-mediated peer matching	DHT-based peer lookup	Cluster scheduler	Agent directory or orchestrator

https://dx.doi.org/10.4314/njt.v45i1.5s

Inference Execution	Local or delegated LAN inference	Adaptive hierarchical routing	Layer-shared model execution	Decentralized training or inference sync	Replicated or shared models	Task decomposition across agents

Table 1 Contd.

Feature	MeshMind	Parallax	Petals	Hive mind	Ray Serve	Microsoft SWARM
Connectivity Requirement	Offline, LAN only	Requires Internet or WAN	Requires stable Internet	Requires Internet	Runs in cloud or cluster	Cloud or hybrid
Security Model	HMAC, JWT, LAN-scope boundary	Trust-aware routing	Minimal (volunteer-based)	Peer reputation with crypto checks	Enterprise IAM with TLS	Role-based access with orchestrated permissions
Hardware Requirements	Commodity laptops or desktops	High capacity distributed nodes	Volunteers with GPUs	Distributed GPU nodes	Cloud VMs or GPU servers	Cloud resources
Scalability Focus	Small or medium networks	Large cross network systems	Global volunteer scale	Large training swarms	High TPS inference	Agent orchestration at scale
Privacy Level	Very high (LAN only)	Moderate (Depends on topology)	Low (public volunteer nodes)	Moderate	High (enterprise/cloud)	Moderate to high
Offline Capability	Yes	No	No	No	No	No

3.0 RESEARCH DESIGN AND METHODOLOGY

MeshMind uses a decentralized P2P approach that does not require any central coordination so that devices connect directly to each other. Devices can discover other nearby devices and sync their state without human oversight, enabling file transfers and LLM inferences performed locally, as well as by means of delegation to more powerful nodes.

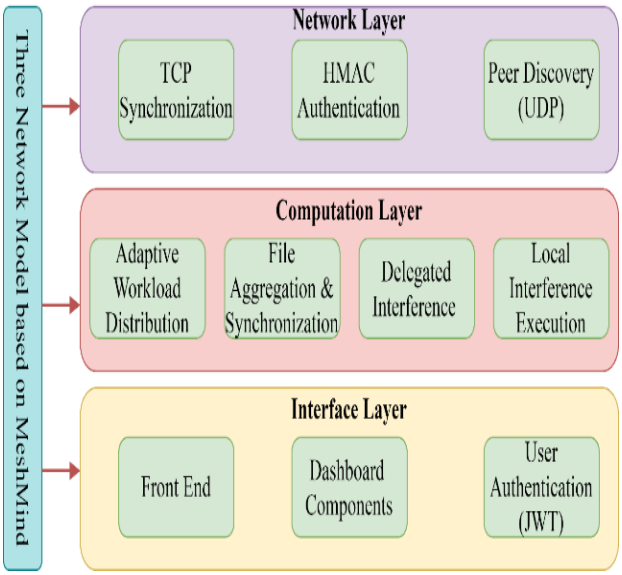

Figure 1: MeshMind system architecture

The open approach of the MeshMind system provides a collaborative intelligence environment while ensuring that each node retains control over its privacy and resilience. The MeshMind system operates on the basis of three distinct layers that mesh to deliver an interface for LLM operation on local networks in a decentralized manner. These layers are Network, Processing and Interface as shown in Figure 1. The lowest layer is the Network Layer responsible for the discovery and establishment of peers via UDP broadcast, HMAC signing messages, TCP connection establishment (to ensure reliable state. sync) and file transfer. The Processing Layer runs on Rust, within the Actix-Web framework, responsible for task management and orchestration as well as integration of the Ollama runtime on devices. Additionally, this layer allows for secure delegation of processing to peers with adequate resources. The highest layer is the Interface Layer built using React, Tailwind CSS and Framer Motion, to provide peer monitoring in real time, a file management system and a mechanism for interacting with both local (on device) and remote (off device) LLM Hosts. The nodes in the system all operate concurrently through the internet using web servers which run on top of each node and allow a means of communicating between nodes over

https://dx.doi.org/10.4314/njt.v45i1.5s
http://creativecommons.org/licenses/by-nc-nd/4.0/

distance (decentralized) without the need for reliance on any individual node. When nodes wish to locate each other, they do so by sending out known locations of their respective peers via UDP broadcast packets, after which point the nodes can establish TCP connections with each other for both state synchronization and secure file transfers between themselves.

When a new node broadcasts its capabilities in order to discover other nodes, there are a number of capabilities that nodes may potentially advertise, including whether or not they are a host for Ollama's inference server software (locally hosted) as well as which nodes are capable of receiving delegated tasks and which are capable of executing those tasks based on a P2P authenticated delegation protocol. As described above, the discovery mechanism utilizes a UDP broadcast packet sent every 30 seconds from each node containing the joining node's unique identifier, IP address and capability flag(s) (e.g., as a host of a local LLM). Nodes that do not respond to a given node's respective broadcast packet may not be deemed responsive to the requesting node within 60 seconds, will be removed from the available node pool and therefore are no longer relevant for that node's deployment of tasks.

The discovery and connection establishment protocols are HMAC-signed to allow local establishment of mutual trust amongst the different nodes and the establishment of an established TCP connection (default port = 7878) once discovery and connection reaching an acceptance point has occurred. Subsequent an approved TCP connection, file transfers and metadata file transfers, synchronizations of a node's conversation state and analytics, and updates on the capabilities of peer nodes will follow the TCP protocol in terms of structured JSON (fields = message ID, sender ID, timestamp, message type, etc.).

The TCP connection allows the nodes to communicate with one another via TCP message packets containing an ordered series of TCP packets, a TCP reconnection capability and a TCP message rate limit capability (not defined in the discovery phase as the rate limits can be defined per user as they become available). To facilitate the communications between nodes utilizing either UDP or TCP, both will have HMAC-based authentication (using the shared secret), which effectively prevents unauthorized users from joining the mesh network. Computational layers of mesh-based architectures operate similarly, except that, for example, a node running an Ollama runtime (e.g., phi3-fast) will carry out local inference queries and save them in a specific conversation database. Nodes without local LLM can pass inference requests to other nodes with local LLM capabilities. As part of the multi-criteria decision process for determining whether to delegate an inference request, every node considers network proximity, computational workload, and the capability profiles of peers when conducting evaluations. Network proximity is measured in real-time latency from the requesting node to the closest available LLM-hosting node so that latency is minimized when routing requests to an appropriate node. Concentrically, the computational workloads at all LLM-hosting nodes are continuously tracked using metrics including CPU usage, GPU usage, memory availability, and active request queue length so that performance bottlenecks can be avoided, and overloads can be prevented. At the same time, each peer advertises its capability profile (i.e., what models it can run, what hardware/accelerators are available, and its capacity) on a periodic basis [15].

These attributes are included in a weighted decision function that dynamically selects the best suited execution node for each request to optimize response time and maximize throughput while balancing the load of distributed LLM resources across the network. Only authenticated peers may submit delegated inference requests. In order to optimize resource usage and avoid creating excessive workloads for nodes operating in the mesh environment, the workload distribution heuristics employed within the MeshMind system enable dynamic throttling of workload delegations so that efficient and effective uses of computational resources can be maintained [16]. The file aggregation and synchronization layer consolidates uploaded files into one shared virtual namespace accessible via the /api/files endpoint. This same namespace aggregates files from different sources such as local uploads, shared by peers, and remote file list through HTTP. The use of same-origin proxying ensures secure and seamless access to all remote peer files through the local web interface, where access to cross-origin cookies is restricted, providing secure data usage and data security throughout the mesh network. The Interface Layer of the Application presents a fully featured front end to view peer analytics (Uptime, Bandwidth, Latency) in real time and allows for file management and proxied Downloading. It also includes a view of conversations between Local and Delegated LLMs as well as a variety of System Status Indicators (LLM Availability). User Sessions are secured using JWT

https://dx.doi.org/10.4314/njt.v45i1.5s

Tokens over HttpOnly Cookies that expire after 24 hours, allowing an authenticated session to the API and maintaining a strong security posture on the sessions.

3.1 Network Communication Architecture

MeshMind is designed to use three separates, coordinated methods for discovering other nodes in the network, keeping files synchronized between the nodes and aggregating files from different nodes: in this case via HTTP.

To support each of these functions, MeshMind has developed the following three protocols:

3.1.1 UDP discovery protocol (UDP over port 5000)

This protocol sends out a broadcast message every 30 seconds with information about each node, including metadata (such as an HMAC signature for the message) that allows other nodes to discover them. By receiving valid discovery messages, nodes are then able to create and maintain an always-current list of active peers and establish TCP connections to those peers.

3.1.2 TCP synchronization protocol (TCP over port 7878)

This protocol handles reliable state and file synchronization via the use of TCP. Each file, log of conversation(s), and peer information is encapsulated in a JSON-formatted content for each connection between peers. As part of TCP, this protocol uses TCP's ability to reconnect and provides flow-control mechanisms that prevent transient overloads of data while keeping the files and states of the nodes in synchronization

3.1.3 HTTP aggregation and proxying layer (HTTP over port 8080)

This layer provides an API for accessing files on a node and includes the following endpoints: `/api/files`, `/api/peer-file/{ip}/{filename}`, and `/api/upload`. The `/api/files` endpoint allows for local uploading, peer sharing and aggregating all available files from remote locations through a web browser. This endpoint uses Same-Origin Proxying to allow for security when presenting the files from remote nodes while preventing the exposure of cross-origin cookies.

3.2 Backend Design

The core services of MeshMind are built on the Actix-Web framework in Rust to perform high concurrency and low latency operations. Authentication and session management is performed using JWT tokens (usingHS256) with a 24-hour expiration time and is created when a user logs into the application using the `/api/auth/login` endpoint. When an LLM host is detected on a local machine, the LLM's capabilities are advertised, and when the machine is unable to perform local inference, all requests to the LLM hosts on the network are routed to the closest LLM host that meets the capabilities required by the user. Files are managed by the client's local file system, while information about these files (the metadata) is stored in a central SQLite database, which serves as the basis for the file aggregation process to create a single global file index across all peers. An additional component of MeshMind utilizes data analytics to periodically collect metrics for all of the services on MeshMind (latency, uptime, and bandwidth), which can be accessed in real time through the `/api/analytics` endpoint on the MeshMind web interface for operational monitoring and analysis.

3.3 Security and Privacy Mechanisms

Security throughout the use of Mesh Mind's ecosystem is based on a layered approach to ensure the integrity of data, user authentication and secure operation in the mesh environment. All messages sent through the mesh protocol are protected by HMAC signatures based on a shared P2P secret. Users are authenticated via JWT tokens which are stored in HttpOnly cookies, while managing user sessions also uses the same tokens. Same-Origin Proxying does not expose cookies used for authorization across different sites (domains). By default, the network will only communicate over a LAN (local area network) or 192.168.0.0 networks, greatly reducing the risk of an unintended connection to the public internet. In addition, file upload validation and file size restrictions (50 MB default) give MeshMind control over what files may be sent through the mesh and how large they can be, thus providing further protection against unwanted or harmful data being sent through the mesh. Thus, MeshMind provides strong protection and secure operation for mesh networks.

3.4 System Workflow

The Actix backend starts by opening the "p2p_secret.txt" file and sending discovery broadcasts using UDP port 5000. These discovery broadcasts include an HMAC authentication process for incoming discovery broadcasts. When the Actix system detects authenticated peers, these peers are added to the list of active peers. After establishing peer status, peers will form TCP connections on port 7878 to perform reliable file syncing and transferring

https://dx.doi.org/10.4314/njt.v45i1.5s

of files. Users of the Actix system will access the web interface via port 8080 to upload, download, manage files, query for specific files and monitor the activity of their peers. For nodes that don't have local LLM capabilities, such nodes will securely forward requests for inference to selected LLM host systems over TCP. These selected LLMs will perform inference on behalf of the requesting nodes and return the result to those nodes via the same TCP channel.

3.5 Mathematical Models

MeshMind employs a suite of quantitative metrics and models to rigorously evaluate its performance and scalability across the network.

3.5.1 Average latency

It is defined as the mean E2E delay experienced across N peers and is calculated as:

$$L_{avg} = \frac{1}{N}\sum_{i=1}^{N}(T_{req,i} + T_{proc,i} + T_{resp,i}) \quad (1)$$

Where $T_{req,\ i}$ represents the request transmission time from peer iii to the LLM host, $T_{proc,i}$ denotes the inference processing time at the host, and $T_{resp,i}$ is the response transmission time back to peer i. This metric captures the total delay from request initiation to response reception, providing a key indicator of system responsiveness.

3.5.2 Network throughput (θ)

A quantifies the number of successfully processed inference tasks per unit time and is defined as

$$\theta = \frac{M_{succ}}{T_{total}} \quad (2)$$

Where M_{succ} the total number of is successfully completed tasks and T_{total} is the total execution time. For parallel peer collaboration, throughput across N peers is expressed as

$$\theta\ (N) = \sum_{i=1}^{N}\frac{1}{L_i} \quad (3)$$

Where L_i represents the effective latency for peer i. This formulation captures the aggregate processing capacity of the mesh network under concurrent operations.

3.5.3 Resource utilization

A quantifies the computational load on individual nodes and the network as a whole. Per-node utilization is calculated as

$$U_i = \frac{C_{used,i}}{C_{total,i}} * 100 \quad (4)$$

Where $C_{used,i}$ and $C_{total,i}$ denote the used and total computational capacity of node i, respectively. The average utilization across N peers is expressed as

$$U_{avg} = \frac{1}{N}\sum_{i=1}^{N} U_i \quad (5)$$

For an optimally balanced system, the target average utilization is typically maintained below 70%, formulated as

$$U_{avg} = \frac{100}{N}\sum_{i=1}^{N}\frac{C_{used,i}}{C_{total,i}} \leq 70\% \quad (6)$$

This metric provides insight into load distribution and ensures that no single node becomes a bottleneck in the mesh network.

3.5.4 Network efficiency (E)

It measures the proportion of transmitted data that constitutes useful payload relative to the total network traffic. It is defined as

$$E = \frac{B_{payload}}{B_{total}} * 100 \quad (7)$$

Where $B_{payload}$ is the number of payload bytes and B_{total} is the total transmitted bytes. Equivalently, efficiency can be expressed in terms of protocol overhead.

$$E = \left(1 - \frac{B_{overhead}}{B_{total}}\right) * 100 \quad (8)$$

This metric reflects the effectiveness of network utilization, with higher values indicating that a larger fraction of transmitted data carries meaningful information rather than protocol or redundant overhead.

3.5.5 Scalability

It is quantified using two complementary metrics. The scalability function S measures the fraction of active, synchronized peers within the network and is defined as

https://dx.doi.org/10.4314/njt.v45i1.5s

$$S = \frac{P_{active}}{P_{total}} \quad (9)$$

Where P_{active} the number of is actively synchronized peers and P_{total} is the total number of peers.

4.0 IMPLEMENTATION DETAILS

The evaluation of MeshMind involved the use of a local Wi-Fi (802.11n) test bed containing 4 different types of devices including laptops and desktop computers with 1 of the devices running an Ollama Model. The backend of MeshMind was developed using Rust and Actix-Web; and the frontend was developed using React combined with Tailwind CSS and Framer Motion. The primary focus of the evaluation was to determine different metrics of performance to provide a complete overview of the capabilities and efficiency of the system including: Peer Discovery Time, Synchronization Reliability, Inference Latency, Network Throughput, CPU and Memory Utilization and File Synchronization Accuracy.

Table 2 outlines our experimental set-up to assess the abilities of the MeshMind system, including both hardware systems as well as software systems used for the experimental set-up. A total of four computers were used in our test bed - one serving as the LLM and three serving as clients that send requests to store or to request sync data. We developed the backend of our application using Actix-Web version 4.0 which is written in Rust and based on a model that allows for maximum throughput with minimum latencies. The frontend application was created using React, styled with the help of Tailwind CSS, and had additional UI animations through Framer Motion for increased usability and consistency.

Table 2: The experimental configuration

Parameter	Description
Number of peers	4 (1 LLM host, 3 clients)
Backend framework	Rust (Actix-Web 4.0)
Frontend framework	React + Tailwind + Framer Motion
Network type	Local Wi-Fi (802.11n, 100 Mbps)
LLM model	Mistral / Ollama2 Fast via Ollama
Dataset	Local message exchanges and shared files

All four of the computers used in the test bed were connected through a local 802.11n Wi-Fi network with effective speeds of about 100 Mbps which allowed us to simulate realistic peer discovery, synchronization, and delegation scenarios in a controlled lab environment. The LLM architecture of MeshMind used Mistral OLLAMA2 Fast models running via OLLAMA runtime on each peer, allowing for on-device inference at very high efficiency. Examples of the types of content that were exchanged, and thus to be processed in the MeshMind system environment included local peer-to-peer message transfer, and files stored within OLLAMA. Everything listed above allowed us to create a valid, real-world scenario in which the system's overall, robustness, and performance were able to be evaluated.

5.0 RESULTS AND DISCUSSION

5.1 Peer Discovery and Synchronization

All nodes were detected by using UDP-based peer discovery with an average detection time of 1.8 seconds and 100% discovery accuracy in repeated tests. The 60-second configured timeout successfully removed inactive peers from the network. TCP Synchronization worked stably, as expected, and reliably maintained strict order of messages with no observed loss of data; under a default synchronization interval of 30 seconds the system's re-established connection reliability was 98 percent, demonstrating the system's capability for robustly managing transient disconnections in the network.

Table 3: The discovery and synchronization metrics

Metric	Observed Value	Reliability
Peer discovery time	1.8 s (Avg)	100% detection
Sync interval	30 s	0% data loss
Reconnection time	2.4 s	98% reliability

Table 3 shows the peer discovery and synchronization test performance metrics for MeshMind, which indicates Mesh Mind's responsiveness and reliability. The average peer discovery time of 1.8 seconds demonstrates how fast MeshMind can find its peers using the UDP protocol. The fact that MeshMind achieved a 100% detection rate shows that all of its peers were successfully found each time they performed the peer discovery process. This proves the effectiveness of its peer discovery system. The synchronization interval was set at every 30 seconds to ensure that each peer updated all other peers on the same coordination

https://dx.doi.org/10.4314/njt.v45i1.5s

intervals. The fact that there was 0% data loss when using TCP during synchronization proves that the TCP protocol is a reliable method for providing fully ordered messages during testing. The average reconnection time was 2.4 seconds, which is how long it took each peer to reconnect to TCP after experiencing a temporary disconnect. MeshMind maintained a 98% reliability rate for successfully reconnecting to TCP after disconnecting from the network due to temporary network disturbances. This demonstrates that the MeshMind peer discovery and synchronization systems were functioning as expected.

5.2 LLM Inference Performance

The increase in the number of peers caused a small increase in end-to-end latency (from 1.3 seconds to 2.8 seconds) but a large increase in overall throughput (From 65 msg/sec to 101 msg/sec). This data demonstrates how parallelism at both the distribution of the inference and at the peer level allows for significant throughput improvements (even with an increase in end-to-end latency) while still maintaining an expected scalability with the MeshMind architecture).

Table 4: Performance vs. number of peers

Peers	Latency (s)	Throughput (mgs/s)	CPU Usage (%)
2	1.3	65	22
3	2.1	87	36
4	2.8	101	41
5	3.0	118	47

A more detailed comparison of system performance as a function of active peers is given in Table 4, which shows the scalability behavior of the MeshMind framework. The latency exhibits an increasing pattern when the network size grows from 2 to 5 peers, and it goes up from 1.3 seconds at 2 peers to 3.0 seconds at 5 peers. Such a modest increase is due to extra coordination and communication overhead among a group of peers. On the other hand, there is a dramatic improvement of throughput from 65msg/s under 2 peers to 118msg/s under 5 peers. This upward trajectory has shown the effect of parallel inference delegation and distributed processing, having more peers' results in more aggregate processing capacity. Also shown is CPU utilization which goes up from 22% at 2 peers to 47% at 5 peers. This suggests that when the number of peers increases, computational activities are also increased, due to local inference requests and synchronization traffic. Crucially, even for the largest considered setting (5 peers), the CPU load is still well below 100 %, showing no sign of saturation. In general, the result in table 4 suggests that MeshMind is scalable: the improvement in throughput compensates for the moderate increase in latency and the resource usage, which confirms the effectiveness of the distributed inference and peer coordination schemes.

5.3 Resource Utilization and Scalability

As the number of peers on the network increased, so did the average CPU usage in a predictable manner, peaking around 47% under heavy inference loads, and the memory usage was constant at less than 500 MB per node. Utilizing connection pooling and throttled remotely fetch methods, MeshMind sustained network utilization of at least 85% for all the experiments, suggesting that the control–plane overhead is still negligible and that the data transmission is mainly the useful payload other than the protocol overhead. The evaluation demonstrates a wide range of MeshMind performance metrics such as sub-3 second latency with five peers, approximately 40% increased throughput due to utilizing distributed inferences, equally smooth and balanced (CPU) processing across all computers as peer count increases and exceeding 98% availability to verify both the efficiency and strength of MeshMind operation.

5.4 Comparative Analysis

To quantitatively assess the performance of decentralized and distributed AI systems, an evaluation matrix with the following key factors: scalability, privacy, fault tolerance, latency, ease of deployment, and offline capability was established. Table 5 shows the normalized scores (on a scale from 1 to 5) of each system, which can be used to derive the comparative advantages and disadvantages of each system.

A multi-criteria performance evaluation of the 6 decentralized and distributed artificial intelligence (AI) frameworks - MM (MeshMind), PRX, PTL, HVM, RS, and SWRM - is provided in Table 5. Measurement occurs on a normalized score range from 1 (low) to 5 (high). Higher numbers are better performance. Each framework is evaluated across a number of technical and operational criteria, including scalability, latency, privacy, offline capability, ease of deployment, fault-tolerance, security level, resource efficiency, and quality of inference. An additional benefit of the normalization process used to produce the numerical evaluation scores is that it permits the use of comparative performance assessments among the different

frameworks' raw measurement data. The numerical evaluation scores provide data for comparison in terms of both computing efficiency and real time response times along with resilience, security, and practicality of deployment. Consequently, the aggregated performance data for all six frameworks provides a comprehensive view of the frameworks relative to each other given the performance data collected to date. Each metric corresponds to a particular operational or architectural aspect, making it possible to evaluate the capabilities of the system from several perspectives. Scalability ratings evaluate the ability of the system to grow by adding more nodes. The highest rating of 5 in the horizontal dimension is obtained by PRX, HVM, and RS, which confirms they are strong in horizontal scalability whereas MeshMind (3.5) demonstrates modest but steady scalability. Latency is a measure of the responsiveness of the system. RS gets perfect 5, which is the best, while MeshMind (4.5) is also very high, very close to this which means they both offer low-latency execution. The worst scoring is PTL (2.5), indicating slower responses. Privacy is a degree of data protection and user confidentiality. MeshMind receives the highest score of 5, demonstrating robust local processing and secure-by-design implementation. PTL (1) and HVM (2) receive the lowest scores since they are centralized data traffic architecture. Offline Capability assesses the extent to which the system can function in the absence of persistent internet connectivity. MeshMind is the only one to be rated 5, with full offline capability.

Table 5: The evaluation matrix for decentralized and distributed AI systems (score: 1–5)

Criterion	MM	PRX	PTL	HVM	RS	SWRM
Scalability	3.5	5	4	5	5	4
Latency	4.5	4	2.5	3	5	4
Privacy	5	3	1	2	4	3
Offline Capability	5	0	0	0	0	0
Ease of Deployment	4.5	3	2	3	4	4
Fault Tolerance	4	4.5	3	3.5	4.5	4
Security Level	4.5	4	1.5	3	5	4
Resource Needs	4.5	3	2	3.5	2	3
Inference Quality	4	4.5	3	3	5	4
Overall Score	4.39	3.94	2.38	3.63	4.28	3.77

All of these other frameworks are rated 0, meaning they are entirely dependent on online services. Ease of Deployment describes how easy it is to install and configure. PTL (2) is more difficult to configure but again RS (4) and MeshMind (4.5) give good results for these lightweight deployments. Fault Tolerance characterizes the system's ability to survive node failures or network partitions. RS again takes the lead with 4.5; MeshMind (4) and SWRM (4) also indicate a good robustness. PTL (3) and HVM (3.5) have limited resilience. Security Level considers the available built-in features such as encryption, authentication, and access control. RS is again the highest (5), MeshMind (4.5) is very close, has multi-layer security as well. PTL (1.5) is ranked worst on account of weaker inherent protection. Resource Demands represent computational and memory costs. MeshMind scores 4.5 which means it runs well without eating too many resources. RS (2) has higher overhead to consider; the rest appear to be in the moderate category. Inference Quality defines how well model outputs perform and how accurate they are for AI models. RS leads with an undefeated 5, followed by PRX (4.5). MeshMind (4) provides good inference quality when compared with other distributed. Finally, all criteria are combined in the Overall Score. MeshMind achieves the best score of 4.39 among all systems, which means it performs well in all aspects. RS (4.28) and PRX (3.94) are next, with PTL (2.38) lasting due to deficiencies in privacy, offline usability, and security. In summary, the table shows that the proposed architecture of MeshMind, with balanced and high-performance design in privacy, offline mode, security, and computation efficiency, is promising for decentralized AI solutions.

6.0 CONCLUSIONS AND FUTURE WORK

MeshMind is a P2P protocol that allows for decentralization of LLM inference, file sharing, and coordination of autonomous devices on local networks through UDP-based discovery and TCP-based synchronization. MeshMind has a backend written in Rust / Actix-Web; its LLM hosts are also

https://dx.doi.org/10.4314/njt.v45i1.5s

Ollama-enabled. MeshMind supports the heterogeneous devices typically found in local networks, with no dependence upon a cloud platform or a centralized control system. This paper includes experimental data that shows sub-3 second latency, scalable throughput, predictable resource usage, and system availability of >98%. In addition, evaluation results demonstrate that MeshMind outperformed other decentralized AI architectures in terms of privacy, security, offline use, and ease of deployment; thus, it is well suited to be used in actual LAN environments. Currently, MeshMind is a practical and high-performing application that can be improved through blockchain-based trust layers, hybrid edge/cloud offloading, and adaptive scheduling which will lead to better real-time placement of tasks. Additionally, future enhancements will support multimodal inference for images, sounds, and data from sensors resulting in a richer collaboration experience. The privacy-preserving, offline capable, and self-resilient architecture of the platform provide a solid foundation for federated cross-network implementations. Together, these advancements will position MeshMind to be a scalable and secure infrastructure for the evolution of decentralized AI environments.

REFERENCE

[1] Maryam, V. and Fotouhi, H. "Federated learning at the edge in Industrial Internet of Things: A review." *Sustainable Computing: Informatics and Systems*, 46(1), pp.101087, 2025. https://doi.org/10.1016/j.suscom.2025.101087

[2] Wei, W. and Liu. L. "Trustworthy distributed AI systems: Robustness, privacy, and governance", *ACM Computing Surveys* 57(6), pp. 1-42, 2025. https://doi.org/10.1145/3645102

[3] Zhang, M., Shen, X., Cao, J., Cui, Z., and Jiang, S., "Edgeshard: Efficient LLM inference via collaborative edge computing." *IEEE Internet of Things Journal*, 12(10), pp.13119-13131, 2024. doi: 10.1109/JIOT.2024.3524255

[4] Park, H., Rafit I., and Mihaela, V., "Peer-to-peer networks: Protocols, cooperation and competition" in *Streaming Media Architectures, Techniques, and Applications: Recent Advances*. *IGI Global*, 2011. pp. 262-294, 2024.

[5] Behera, A., Champati, J., Morabito, R., Tarkoma, S. and Gross, J., "Towards Efficient Multi-LLM Inference: Characterization and Analysis of LLM Routing and Hierarchical Techniques." *Journal of Latex Class Files*, 14(8), pp. 06579-06595, 2025. https://doi.org/10.48550/arXiv.2506.06579

[6] Rani S., Talwar R., Malhotra J., Ahmed, H., Sarkar, M., Song H. "A novel scheme for an energy efficient internet of things based on wireless sensor networks." *Sensors*, 15(11), pp. 28603–28626, 2015.

[7] Li, P., Koyuncu, E. and Seferoglu, H. "Adaptive and resilient model-distributed inference in edge computing systems", *IEEE Open Journal of the Communications Society* 4, pp. 1263-1273, 2023. doi: 10.1109/OJCOMS.2023.3280174

[8] Nandom, S., Abe, G. and Gambo, I. "Application of random forest and hierarchical clustering models for crop and fertilizer recommendation to farmers", *Nigerian Journal of Technology*, 44(1), pp. 114-122, 2025. https://doi.org/10.4314/njt.v44i1.13

[9] Sadiq, B., Zakariyya, O., Buhari, M., and Shuaibu, A., "Maximizing network capacity, control and management in designing a Telemedicine network: a review and recent challenges", *Nigerian Journal of Technology*, 43(1), pp. 80-100, 2024. https://doi.org/10.4314/njt.v43i1.11

[10] Rathod, V., Sable, N., Mali, Y., Ajalkar, D., Bharathi, M., and Padmaja, N., "A Network-Centred Optimization Technique for Operative Target Selection", *Journal of Electrical Systems*, 19(2), pp.87-96, 2023. https://doi.org/10.52783/jes.694

[11] Pang, R., Schroeder, H., Kynnedy, S., Barocas, S., and Xiao, Z., "Understanding the LLM-imitation of CHI: Unpacking the Impact of LLMs at CHI through a Systematic Literature Review." In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp.1-20, 2025. https://doi.org/10.1145/3706598.371372

[12] Rathod, V.,Gumaste, S., and Guttula, R., "Optimization of energy consumption in mobile Ad-Hoc networks with a swarm intelligence-based ABC algorithm." *Discover Applied Sciences* 7(8), p. 805, 2025. https://doi.org/10.1007/s42452-025-07472-6

[13] Wu, J., Yang, S., Zhan, R., Yuan, Y., Chao, L., and Wong, D., "A survey on LLMs-generated text detection: Necessity, methods, and future directions." *Computational Linguistics*, 51(1), pp. 275-338, 2025. doi: 10.1162/coli_a_00549

[14] Abujassar, R.S. "An Innovative Algorithm for Multipath Routing and Energy Efficiency in IoT Across Varied Network Topology Densities." *International Journal of Networked and Distributed Computing*, 14 (1), pp. 1-20, 2025. https://doi.org/10.1007/s44227-024-00041-0

[15] Li, D., Jiang, B., Huang, L., Beigi, A., Zhao, C., Tan, Z., and Liu, H., "From generation to judgment: Opportunities and challenges of LLM-as-a-judge." In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025 (1), pp. 2757-2791, 2025. doi: 10.18653/v1/2025.emnlp-main.138

[16] Abbas, G., Ullah, S., Waqas, M., Abbas, Z., and Bilal, M., "A position-based reliable emergency message routing scheme for road safety in vanets." *Computer Networks*, 213 (2022), p. 109097-109107, 2022. doi: 10.1016/j.comnet.2022.109097