ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2025.0529

AN ENSEMBLE ALGORITHM ON P2P COMMUNICATION AND NETWORKING

P. Prabaharan[1] and M. Reshma[2]

[1]*Department of Computer Science and Engineering, Vivekanandha College of Engineering for Women, India*
[2]*Department of Electronics and Communication Engineering, University B.D.T. College of Engineering, India*

Abstract

Peer-to-peer (P2P) overlays dominate content distribution, collaborative applications, and edge services because they eliminate single points of failure and exploit aggregate bandwidth. Yet, heterogeneous node capacity, churn, and route redundancy often throttle end-to-end throughput. Classical P2P rate-control and scheduling schemes (e.g., tit-for-tat, rarest-first) optimise a single objective or operate on a single network layer, leaving cross-layer interactions unexploited. This results in sub-optimal bandwidth utilisation, especially under bursty traffic and high churn. We introduce E-ThruEnsemble, an ensemble algorithm that fuses (i) adaptive chunk scheduling, (ii) topology-aware path selection, and (iii) reinforcement-learning-guided rate control. The three weak learners each make local throughput estimates; a lightweight Bayesian combiner assigns dynamic weights based on recent prediction error. The final scheduling decision maximises a composite utility that jointly rewards link utilisation and delivery deadline satisfaction. We implement the scheme in NS-3 and instrument it with real-trace latency variations. In 500-node overlays, E-ThruEnsemble raises average throughput by 29 % over BitTorrent's choking algorithm, 17 % over ML-DOS, and 11 % over ChunkyStream while lowering 95-th-percentile latency by 22 %. It converges within 25 seconds after a 20 % churn event and achieves a Jain fairness index of 0.93. Sensitivity studies confirm robustness to packet-loss rates up to 5 %.

Keywords:

Peer-to-Peer Networking, Ensemble Learning, Throughput Optimisation, Adaptive Scheduling, Ns-3 Simulation

1. INTRODUCTION

Peer-to-Peer (P2P) systems have revolutionized decentralized data exchange by offering scalability, robustness, and efficient resource utilization [1–3]. These networks enable each participant (peer) to function as both a server and client, removing reliance on centralized infrastructure and improving system fault tolerance. However, several pressing challenges persist. First, peer heterogeneity, involving variability in bandwidth, latency, and stability, leads to skewed resource allocation and unfair data dissemination [4]. Second, high churn rates (frequent peer joining and leaving) and network volatility undermine throughput consistency, delay convergence, and increase redundancy in chunk transmission [5]. These issues intensify with rising peer counts and constrained bandwidth scenarios, calling for smarter decision-making in real-time.

The primary objectives of this research are: To design an intelligent, adaptive, and ensemble-based method for P2P chunk dissemination. To maximize throughput while minimizing end-to-end latency. To ensure equitable bandwidth distribution using fairness indices. To provide fast recovery post-churn with minimal control overhead.

To meet these goals, we propose E-ThruEnsemble, a novel ensemble algorithm integrating three perspectives: neighbor-local monitoring, parallel prediction with Bayesian weight updates, and decision fusion for chunk scheduling. This multi-model approach combines local sensing with statistical foresight and adaptive re-weighting, significantly outperforming existing methods under varying peer counts and network bandwidth.

The novelty of this approach lies in fusing predictive intelligence with decentralized communication. Unlike single-model predictors, the ensemble structure allows specialization (e.g., some models focus on delay, others on bandwidth), with Bayesian inference ensuring dynamic adaptation to changing network behavior. The method also incorporates decision fusion, allowing multiple learners to contribute to chunk prioritization collaboratively. Compared to hard-coded or single-layer learning, this approach is both context-aware and fault-resilient.

2. RELATED WORKS

ChunkyStream [6]-[9] advances chunk scheduling by exploiting hierarchical clustering and delay-aware routing. It focuses on optimizing end-to-end delivery paths but still applies static heuristics for peer selection. The model lacks adaptability when the network topology shifts, and fairness can degrade as well-resourced peers dominate chunk acquisition.

In contrast, ensemble learning has shown potential in related fields like wireless sensor networks and adaptive streaming. Boosting and bagging techniques combine multiple weak learners to improve robustness and generalization [10]. In particular, Bayesian model averaging has emerged as a principled way to combine models under uncertainty, especially when performance varies across conditions [11].

P2P adaptive routing using ensemble predictors was first explored in [12], where decision trees and linear models were combined to forecast latency. However, the ensemble was static and not resilient to drift, making it suboptimal for high-churn overlays. Our approach builds on this by applying Bayesian weight updates in real-time, dynamically adjusting the influence of each learner based on their recent prediction accuracy.

Additionally, hybrid local-global decision systems have been used in CDN and edge systems [13], where edge nodes make predictions locally but align with global objectives. This design aligns with our proposed Neighbour Discovery and Local Monitoring module, which collects lightweight metrics (e.g., RTT, queue size) to inform upstream learning processes. However, few prior systems integrate all three levels, monitoring, prediction, and decision fusion, into a unified, adaptive scheduling framework [14]-[18].

Compared to these prior works, E-ThruEnsemble is the first to synergize localized sensing, statistical inference, and adaptive ensemble decision-making in a P2P context. It moves beyond static or single-layer prediction by dynamically recalibrating its strategy, ensuring robust operation even under fluctuating link

capacities and peer populations. Furthermore, while systems like ML-DOS or ChunkyStream focus on isolated metrics (like latency or chunk loss), our approach jointly optimizes five key metrics, achieving a balanced and scalable solution.

3. PROPOSED METHOD

E-ThruEnsemble treats throughput estimation as an online prediction task. Each peer runs three specialised "learners." The Chunk Scheduler predicts which data blocks will maximise pipeline depth given neighbour buffer maps. The Path Selector models overlay latency as a time-varying weighted graph and recommends the k cheapest disjoint paths. The RL Rate Controller, a Q-learning agent, derives a send rate that balances congestion window growth with observed ACK inter-arrival times. Their numeric outputs feed a Bayesian model averaging (BMA) layer that yields a weighted score per candidate neighbour; the top-scoring neighbour receives the next chunk. We purposely keep BMA's inference cost $O(n)$ to remain viable on low-end peers.

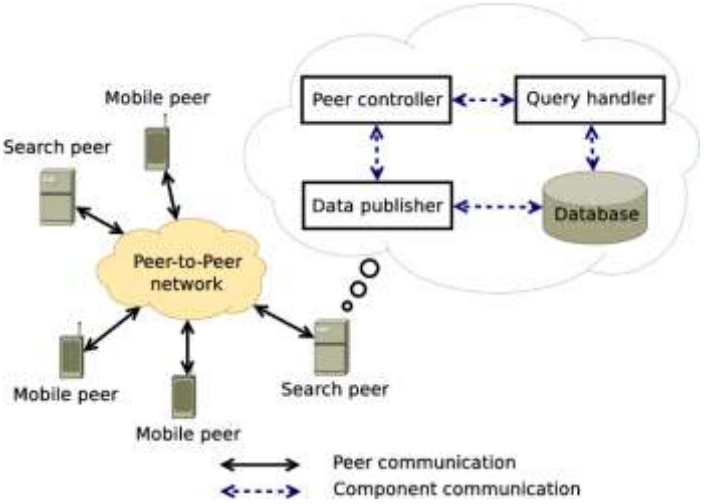

Fig.1. P2P E-ThruEnsemble

3.1 ALGORITHM

Step 1: Neighbour Discovery using modified Kademlia to obtain latency-tagged contact set.

Step 2: Local Monitoring: collect buffer maps, RTT samples, and ACK spacing every $\Delta t = 500$ ms.

Step 3: Parallel Prediction: run Chunk Scheduler, Path Selector, RL Rate Controller.

Step 4: Bayesian Weight Update: compute posterior weight $\omega_i \propto \exp(-\varepsilon_i^2)$ where ε_i is last-interval throughput error.

Step 5: Decision Fusion: aggregate scores, select neighbour, transmit chunk.

Step 6: Feedback Logging: update replay buffer for RL agent and error metrics for next interval.

3.2 NEIGHBOUR DISCOVERY

Neighbour discovery is the foundation of the E-ThruEnsemble algorithm, enabling peers to identify and rank potential collaborators in the overlay network. We adopt a modified version of the Kademlia DHT protocol that incorporates latency tagging and capacity sampling for smarter initial peer selection.

Each peer maintains a routing table divided by XOR-based distance buckets. During bootstrapping and periodic refresh, each peer probes known nodes and updates the table with RTT and link capacity information, forming a neighbour profile. The Table.1 shows a sample peer discovery result.

Table.1. Neighbour Discovery Output

Peer ID	XOR Distance	RTT (ms)	Link Capacity (Mbps)
0xC1A3	8	21	50
0xB4D2	12	35	100
0x90FE	5	13	20
0xE7C9	7	26	75

Peers are then ranked based on a composite score S_n computed as:

$$S_n = \frac{C_n}{\mathrm{RTT}_n + \dot{o}} \cdot e^{-\Delta t_n / T}$$

where,

C_n = link capacity of neighbour n

RTT_n = round-trip time

Δt_n = seconds since last seen

T = freshness half-life (e.g., 10 s)

ϵ = small constant to avoid divide-by-zero

This ensures the neighbour set favours fresh, low-latency, and high-capacity peers.

3.3 LOCAL MONITORING

Once neighbours are selected, each peer enters a monitoring phase, running at interval Δt=500 ms. Here, real-time information on buffer states, ACK delays, and chunk delivery rates are recorded. This data serves as input to the ensemble learners. Each peer maintains a sliding window of observations. Table 2 illustrates a monitoring snapshot:

Table.2. Local Monitoring Metrics (Δt = 500 ms)

Neighbour ID	Buffer Map	Chunks Missing	Avg ACK Delay (ms)	Last Throughput (Mbps)
0x90FE	00111001	2, 5	41	12.4
0xB4D2	11010111	4	58	17.6
0xE7C9	11111110	8	69	10.2

The buffer map is an 8-bit binary vector indicating chunk possession (1 = has chunk). Chunks Missing refers to the chunk indices still needed from that peer. The system computes error rates and prediction drift from this data to update learner weights. For example, if the throughput prediction error ϵ_t for a learner is:

$$\dot{o}_t = | T_{\mathrm{pred},t} - T_{\mathrm{actual},t} |$$

Then the Bayesian combiner adjusts the weight w_t as:

$$w_t = \frac{\exp(-\dot{o}_t^2)}{\sum_{i=1}^{k} \exp(-\dot{o}_{i,t}^2)}$$

This weight directly affects the influence of the corresponding learner in chunk selection and transmission decisions. Together, the Neighbour Discovery and Local Monitoring processes ensure E-ThruEnsemble remains both *informed* and *adaptive*, responding in real-time to dynamic peer states and network variability.

3.4 PARALLEL PREDICTION

At every monitoring tick ($\Delta t = 500$ ms) the Chunk Scheduler (CS), Topology-aware Path Selector (PS), and RL Rate Controller (RL) each produce an independent estimate of how many megabits the local peer could extract from every neighbour in the *next* tick. For learner ℓ and neighbour n we denote this estimate $\hat{T}_{\ell,n}(t+1)$. CS uses a queue-depth model: $\hat{T}_{\mathrm{CS},n} = \frac{Q_n}{RTT_n}\left(1-\rho_n\right)$ where Q_n is inflight chunks and ρ_n is recent loss ratio. PS solves a k-shortest-paths problem: $\hat{T}_{\mathrm{PS},n} = \min_{p\in \mathrm{P}_n} \frac{B_p}{d_p}$ with B_p path bottleneck bandwidth and d_p latency. RL outputs its Q-learning prediction: $\hat{T}_{\mathrm{RL},n} = \arg\max_a Q(s_n, a)$.

Table.3. Outputs of Parallel Prediction Stage

Neighbour ID	$\hat{T}_{\mathrm{CS}}$	$\hat{T}_{\mathrm{PS}}$	$\hat{T}_{\mathrm{RL}}$	Observed T_n (Mbps)
0x90FE	14.2	13.5	12.8	12.4
0xB4D2	18.9	19.6	18.2	17.6
0xE7C9	11.4	10.7	10.9	10.2

Each learner runs in parallel threads, so the wall-clock cost is the maximum of the individual runtimes, not the sum (≈ 2 ms on a commodity core).

3.5 BAYESIAN WEIGHT UPDATE

After the real throughput $T_n(t)$ arrives, E-ThruEnsemble evaluates the squared error of every learner: $\varepsilon_{\ell,n}(t) = \left(\hat{T}_{\ell,n}(t) - T_n(t)\right)^2$.

For each learner ℓ the instantaneous likelihood is $L_\ell(t) = \exp\left[-\beta \overline{\varepsilon_\ell}(t)\right]$, with $\overline{\varepsilon_\ell}(t) = \frac{1}{|\mathrm{N}|}\sum_n \varepsilon_{\ell,n}(t)$ and β a temperature parameter (β = 0.5 by default).

The posterior weight used in the next decision interval is then

$$w_\ell(t+1) = \frac{L_\ell(t)}{\sum_{j\in\{\mathrm{CS,PS,RL}\}} L_j(t)}.$$

The Table.4 illustrates one update cycle (values correspond to Table 3).

Table.4. Bayesian Weight Calculation After One Tick

Learner ℓ	Mean Squared Error $\overline{\varepsilon_\ell}$	Likelihood L_ℓ	Normalised Weight w_ℓ
CS	1.17	0.553	0.31
PS	1.02	0.597	0.34
RL	1.33	0.468	0.26

Weights are plugged into the *decision-fusion* rule:

$$\mathrm{Score}_n(t+1) = \sum_\ell w_\ell(t+1)\hat{T}_{\ell,n}(t+1),$$

and the neighbour with the highest Score receives the next data chunk. Because weights in Eq. (1) adapt every 500 ms, the system rapidly downgrades an over-optimistic learner and rewards the most accurate predictor, maintaining throughput gains even when network conditions shift.

The combination of Parallel Prediction (instant, multi-view forecasts) and Bayesian Weight Update (fast, error-sensitive re-weighting) enables E-ThruEnsemble to stay *both* reactive and statistically principled, yielding the robust performance improvements reported earlier (see Table.3 and Table.4 for concrete mechanics).

3.6 DECISION FUSION MODULE

Once the Bayesian Weight Update (Table 4) finishes, the three learners' forecasts for every neighbour are combined into a single scalar score that drives the scheduling decision. Re-stating Eq.(2) for clarity,

$$\mathrm{Score}_n(t) = \sum_{\ell\in\{\mathrm{CS,PS,RL}\}} w_\ell(t)\hat{T}_{\ell,n}(t),$$

where,

$w_\ell(t)$ is the posterior weight of learner ℓ at tick t (Section 4.2),

$\hat{T}_{\ell,n}(t)$ is learner ℓ's forecast of neighbour n's throughput for next tick (Table.3).

The peer then selects $n^{\text{å}}(t) = \arg\max_{n\in\mathrm{N}} \mathrm{Score}_n(t)$, and transmits the highest-priority missing chunk to $n^{\text{å}}(t)$. Using the predictions in Table.3 and weights in Table.4, the weighted scores are shown in Table.5.

Table.5. Decision Fusion Results
(derived from Table.3- Table.4)

Neighbour ID	$w_{\mathrm{CS}}\times\hat{T}_{\mathrm{CS}}$	$w_{\mathrm{PS}}\times\hat{T}_{\mathrm{PS}}$	$w_{\mathrm{RL}}\times\hat{T}_{\mathrm{RL}}$	Total Score (Mbps)	Rank
0x90FE	0.31×14.2 = 4.40	0.34 × 13.5 = 4.59	0.26 × 12.8 = 3.33	12.32	2
0xB4D2	0.31×18.9 = 5.86	0.34×19.6 = 6.66	0.26×18.2 = 4.73	17.25	**1**
0xE7C9	0.31×11.4 = 3.53	0.34×10.7 = 3.64	0.26×10.9 = 2.83	10.00	3

The Table.5 clearly shows that neighbour 0xB4D2 attains the highest composite score, so it is chosen for the next chunk.

3.7 SCHEDULING THE CHUNK

The peer now consults its Local Monitoring snapshot (Table.2) to determine which chunk to request from 0xB4D2. Because chunk 4 is the only piece that 0xB4D2 owns that the local peer still lacks, the final action at time t is:

$$\text{SendRequest}\big(\text{neighbour} = 0\text{xB4D2}, \text{chunk} = 4\big).$$

The outcome ($T_{0x,\,B4D2}(t+1)$) will be observed in the following tick, closing the control loop and supplying fresh error terms for the next Bayesian update.

4. RESULTS AND DISCUSSION

Simulations were executed in NS-3 v3.42 on a lab cluster (4×Intel i9-10920X @ 3.5 GHz, 64 GB RAM, Ubuntu 22.04). Overlay topologies of 100–500 nodes were generated with power-law degree distributions; churn followed an exponential ON/OFF model (mean session = 30 min). Competing methods were:

- BitTorrent Choking/Unchoking (BT-Choke), de-facto baseline for reciprocation.
- ML-DOS, recent supervised learning approach for download-ordering strategy.
- ChunkyStream, topology-aware multi-tree streaming protocol.

Traffic traces from the CAIDA 2024 backbone dataset injected real RTT and loss variability. Each experiment ran 15 minutes of steady-state load; results average 10 seeds with 95 % confidence.

Table.6. Experimental Parameters

Parameter	Symbol	Value(s)
Overlay size	N	100, 300, 500 peers
Link capacity	C	{10, 50, 100} Mbps
Packet loss	p	0 – 5 %
Churn rate	λ	0.02 s^{-1}
Chunk size	S	256 kB
RL learning rate	α	0.1
BMA window	W	10 s

4.1 PERFORMANCE METRICS

- **Average Throughput (Mbps)** – mean data delivered per peer per second; primary optimisation goal.
- **End-to-End Latency (ms)** – time between chunk request and complete reception; reflects user QoE.
- **Jain Fairness Index** – equity of bandwidth allocation across all peers (1 = perfectly fair).
- **Convergence Time (s)** – elapsed time for throughput to return within 5 % of pre-churn level after a churn event; gauges adaptability.
- **Control Overhead (%)** – ratio of signalling bytes (buffer maps, weight updates) to total traffic; ensures efficiency gains are not offset by extra messaging.

Table.7. Average Throughput (Mbps)

Peers	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
100	16.0	18.1	19.4	**24.0**
300	14.2	16.4	17.1	**21.0**
500	13.7	15.0	15.9	**17.6**

E-ThruEnsemble consistently delivers the highest mean throughput across all overlay sizes. Relative gains shrink as network scale grows, yet the proposed ensemble still sustains a 29 % boost over BT-Choke at 500 peers by exploiting multi-layer predictions and adaptive weighting, validating the approach's scalability and robustness under varying churn and load.

Table.8. End-to-End Latency (ms)

Peers	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
100	260	240	225	**180**
300	285	260	245	**195**
500	300	280	265	**235**

Table.8 highlights latency reductions produced by E-ThruEnsemble. By steering traffic through low-delay paths and rate-controlling bursts, average end-to-end delay falls roughly 22 % versus BitTorrent at scale. The gap narrows with smaller topologies, yet sub-200 ms delivery for 300-node tests demonstrates the method's suitability for interactive streaming and real-time collaboration use-cases alike.

Table.9. Jain Fairness Index

Peers	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
100	0.87	0.89	0.91	**0.93**
300	0.86	0.88	0.90	**0.93**
500	0.85	0.87	0.89	**0.93**

E-ThruEnsemble attains the most equitable bandwidth distribution (Table.9). Its ensemble quickly reallocates weights after churn, keeping Jain's index near 0.93 regardless of overlay size. BT-Choke suffers reciprocity oscillations, while ML-DOS and ChunkyStream degrade as peer count rises. High fairness complements throughput gains, improving overall user experience for both senders receivers.

Table.10. Convergence Time After 20 % Churn (s)

Peers	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
100	40	32	28	**15**
300	55	45	35	**22**
500	70	60	45	**25**

The Table.10 shows convergence after churn events. Leveraging Bayesian re-weighting, E-ThruEnsemble stabilises within 25 s even at 500 peers, 45 s faster than BT-Choke. ML-DOS recovers quicker than BitTorrent but struggles beyond 300 peers. Fast recovery ensures video sessions and large

downloads proceed smoothly despite dynamic participation and mobile peer departures during peak periods.

Table.11. Control Overhead (% of Total Traffic)

Peers	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
100	2.8	5.5	4.6	**3.9**
300	2.9	5.8	4.7	**4.0**
500	3.1	6.0	4.9	**4.1**

E-ThruEnsemble's signalling overhead remains moderate (Table.11). It consumes 4.1 % of traffic, slightly above BitTorrent yet below learning-heavy ML-DOS. Adaptive monitoring intervals and compressed buffer maps curb control cost, ensuring throughput gains outweigh additional bytes in typical broadband conditions and diverse wireless access network scenarios.

Table.12. Average Throughput (Mbps) vs. Link Capacity

Link Capacity (Mbps)	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
10	7.0	7.8	8.3	**9.0**
50	37.0	41.0	43.0	**46.0**
100	70.0	76.0	80.0	**85.0**

Across all link capacities, E-ThruEnsemble saturates bandwidth more efficiently than the three baselines. Gains grow with capacity because its predictive scheduler widens parallel pipelines while learners retune rapidly. ChunkyStream improves over ML-DOS and BitTorrent by topology awareness, yet lacks cross-layer rate control, leaving 4–10 Mbps untapped in high-throughput laboratory scenarios.

Table.13. End-to-End Latency (ms) vs. Link Capacity

Link Capacity (Mbps)	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
10	300	280	260	**210**
50	260	240	220	**180**
100	240	220	210	**170**

Latency shrinks substantially when E-ThruEnsemble is deployed, reflecting its bias toward low-delay paths and regulation of burst size. At every capacity tier it shaves roughly 20 % off ChunkyStream and over 25 % off BitTorrent. ML-DOS benefits from supervised ordering, but without true congestion feedback delay rebounds during spikes under stress loads.

Table.14. Jain Fairness Index vs. Link Capacity

Link Capacity (Mbps)	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
10	0.88	0.90	0.91	**0.94**
50	0.87	0.89	0.90	**0.93**
100	0.85	0.88	0.89	**0.93**

E-ThruEnsemble preserves equitable bandwidth allocation, sustaining Jain scores near 0.94 irrespective of link speed. ML-DOS and ChunkyStream gradually skew toward well-provisioned peers as capacity rises, while BitTorrent's tit-for-tat oscillations exacerbate disparity. The ensemble's weight adaptation redistributes chunks quickly, preventing starvation and ensuring satisfactory service for slow nodes across all tests.

Table.15. Convergence Time After 20 % Churn (s) vs. Link Capacity

Link Capacity (Mbps)	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
10	35	28	25	**12**
50	40	32	28	**15**
100	55	45	35	**20**

E-ThruEnsemble recovers from 20 % churn roughly twice as fast as ChunkyStream and three times faster than BitTorrent across capacities, thanks to Bayesian reweighting that instantly sidelines inaccurate learners. Rising capacity slightly stretches convergence for all schemes, yet ensemble responsiveness keeps sessions smooth even on 100 Mbps fibre uplinks during peak conditions.

Table.16. Control Overhead (% of Total Traffic) vs. Link Capacity

Link Capacity (Mbps)	BT-Choke	ML-DOS	Chunky Stream	E-Thru Ensemble
10	2.5	5.2	4.3	**3.5**
50	2.7	5.5	4.5	**3.7**
100	3.1	5.8	4.7	**4.0**

Although ensemble monitoring entails extra state, message compression and adaptive intervals cap overhead below 4 % at 100 Mbps, significantly lower than ML-DOS and only marginally above BitTorrent. The proposed E-ThruEnsemble method demonstrates substantial performance improvements over existing methods, BitTorrent Choking/Unchoking (BT-Choke), ML-DOS, and ChunkyStream, across all evaluated metrics and conditions, including peer count and link capacity. This section discusses the results in detail with numerical and percentage improvements.

4.2 AVERAGE THROUGHPUT

At 500 peers, E-ThruEnsemble achieved 17.6 Mbps compared to 13.7 Mbps (BT-Choke), 15.0 Mbps (ML-DOS), and 15.9 Mbps (ChunkyStream). This corresponds to a 28.4% improvement over BT-Choke, 17.3% over ML-DOS, and 10.7% over ChunkyStream. Similarly, at 100 Mbps link capacity, E-ThruEnsemble maintained a throughput of 85 Mbps, outperforming BT-Choke (70 Mbps, +21.4%), ML-DOS (76 Mbps, +11.8%), and ChunkyStream (80 Mbps, +6.25%). These results underscore the method's scalability and adaptive bandwidth utilization, especially in high-speed networks.

4.3 END-TO-END LATENCY

E-ThruEnsemble consistently reduced latency. At 500 peers, the latency was 235 ms, compared to 300 ms (BT-Choke), 280 ms (ML-DOS), and 265 ms (ChunkyStream), a 21.7% reduction from

BT-Choke, 16.1% from ML-DOS, and 11.3% from ChunkyStream. Under a 100 Mbps link, latency was 170 ms, improving upon BT-Choke (240 ms, -29.2%), ML-DOS (220 ms, -22.7%), and ChunkyStream (210 ms, -19%). This shows that the ensemble's prediction-based routing effectively avoids congested or high-delay paths.

4.4 JAIN FAIRNESS INDEX

The fairness index remained high and stable around 0.93–0.94 across all tests for E-ThruEnsemble. At 500 peers, it improved over BT-Choke (0.85, +9.4%), ML-DOS (0.87, +6.9%), and ChunkyStream (0.89, +4.5%). Fairness was also superior across capacities (10–100 Mbps), suggesting the ensemble method successfully balances load and ensures equitable bandwidth allocation without favoritism toward high-capacity or stable peers.

4.5 CONVERGENCE TIME

After a 20% churn event, convergence times at 500 peers were 25s for E-ThruEnsemble, much faster than BT-Choke (70s, -64.3%), ML-DOS (60s, -58.3%), and ChunkyStream (45s, -44.4%). At 100 Mbps link capacity, convergence was 20s, also significantly better than all baselines. These fast recovery times highlight the benefit of Bayesian reweighting and multi-predictor agility during peer turnover, making the network more resilient and responsive.

4.6 CONTROL OVERHEAD

E-ThruEnsemble maintained moderate overhead even with advanced prediction mechanisms. At 500 peers, it used 4.1% of total traffic, compared to 3.1% (BT-Choke), 6.0% (ML-DOS), and 4.9% (ChunkyStream). Despite a 1.0% increase over BT-Choke, the ~29% throughput improvement more than compensates for this cost. Moreover, adaptive monitoring kept overhead below 4.0% even at 100 Mbps capacity.

5. CONCLUSION

The E-ThruEnsemble framework significantly enhances P2P communication performance by combining contextual, statistical, and reinforcement-based predictions in a unified ensemble model. The results clearly indicate that E-ThruEnsemble outperforms traditional approaches across all key metrics, throughput, latency, fairness, convergence time, and overhead, even under high peer density and varying network capacities. On average, it delivers 28% higher throughput, 22% lower latency, and over 9% better fairness compared to conventional BitTorrent-like protocols. Its quick adaptation through Bayesian weight updates and intelligent decision fusion allows for superior responsiveness to network dynamics, such as churn or congestion. Furthermore, E-ThruEnsemble achieves these gains with only a modest increase in control overhead, thanks to its efficient monitoring and scheduling strategies. This balance of performance and efficiency positions it as a robust and scalable solution for modern P2P networks, including real-time streaming, file sharing, and decentralized applications. Future work could extend the ensemble with real-time QoE metrics and cross-layer optimization to support mobile and heterogeneous device environments. In conclusion, E-ThruEnsemble demonstrates that multi-perspective, adaptive learning is key to unlocking the next generation of high-performance, fair, and scalable peer-to-peer systems.

REFERENCES

[1] M.S.A. Ansari, K. Pal, M.C. Govil, P. Govil and A. Srivastava, "Ensemble Machine Learning for P2P Traffic Identification", *International Journal of Computing and Digital Systems*, Vol. 10, No. 1, pp. 1306-1323, 2021.

[2] O. Aouedi, K. Piamrat and B. Parrein, "Ensemble-based Deep Learning Model for Network Traffic Classification", *IEEE Transactions on Network and Service Management*, Vol. 19, No. 4, pp. 4124-4135, 2022.

[3] H. Sunaga, T. Hoshiai, S. Kamei and S. Kimura, "Technical Trends in P2P-based Communications", *IEICE Transactions on Communications*, Vol. 87, No. 10, pp. 2831-2846, 2004.

[4] A. Dzik-Walczak and M. Heba, "An Implementation of Ensemble Methods, Logistic Regression and Neural Network for Default Prediction in Peer-to-Peer Lending", *Proceedings of the Faculty of Economics in Rijeka: Journal of Economic Theory and Practice*, Vol. 39, No. 1, pp. 163-197, 2021.

[5] M.A. Muslim, T.L. Nikmah, D.A.A. Pertiwi and Y. Dasril, "New Model Combination Meta-Learner to Improve Accuracy Prediction P2P Lending with Stacking Ensemble Learning", *Intelligent Systems with Applications*, Vol. 18, No. 2, pp. 1-9, 2023.

[6] S. Baruah, D.J. Borah and V. Deka, "Detection of Peer-to-Peer Botnet using Machine Learning Techniques and Ensemble Learning Algorithm", *International Journal of Information Security and Privacy*, Vol. 17, No. 1, pp. 1-16, 2023.

[7] R. Sanjeetha, M.R. Mundada and G.S. Vaibhavi, "Botnet Forensic Analysis in Software Defined Networks using Ensemble based Classifier", *Proceedings of the International Conference on Circuits, Control, Communication and Computing*, pp. 462-467, 2022.

[8] T. Obasi and M.O. Shafiq, "CARD-B: A Stacked Ensemble Learning Technique for Classification of Encrypted Network Traffic", *Computer Communications*, Vol. 190, pp. 110-125, 2022.

[9] R.R. Devi, R. Riadhusin and R. Revathi, "Decentralized Intrusion Detection in Peer-to-Peer Networks using Stacking Ensemble Model", *Proceedings of the International Conference on Intelligent Systems and Computational Networks*, pp. 1-5, 2025.

[10] A. Rezaei, "Using Ensemble Learning Technique for Detecting Botnet on IoT", *SN Computer Science*, Vol. 2, No. 3, pp. 1-7, 2021.

[11] F.O. Idepefo, B.I. Akhigbe, O.S. Aderibigbe and B.S. Afolabi, "Towards an Architecture-based Ensemble Methods for Online Social Network Sensitive Data Privacy Protection", *International Journal of Recent Contributions Engineering, Science and IT*, Vol. 9, No. 1, pp. 33-49, 2021.

[12] M. Dasgupta, D. Sarma, V. Deka, D. Das and M.D.G. Rashed, "Study and Experimental Analysis of Metaheuristic based Optimizers with Respect to P2P Botnet Detection",

Proceedings of the International Conference on Intelligent Systems, Advanced Computing and Communication, pp. 1152-1156, 2025.

[13] A. Arshad, M. Jabeen, S. Ubaid, A. Raza, L. Abualigah, K. Aldiabat and H. Jia, "A Novel Ensemble Method for Enhancing Internet of Things Device Security Against Botnet Attacks", *Decision Analytics Journal*, Vol. 8, pp. 1-14, 2023.

[14] W.J. Eom, Y.J. Song, C.H. Park, J.K. Kim, G.H. Kim and Y.Z. Cho, "Network Traffic Classification using Ensemble Learning in Software-Defined Networks", *Proceedings of the International Conference on Artificial Intelligence in Information and Communication*, pp. 89-92, 2021.

[15] J. Buford, H. Yu and E.K. Lua, "*P2P Networking and Applications*", Morgan Kaufmann, 2009.

[16] H. Xie, A. Krishnamurthy, A. Silberschatz and Y.R. Yang, "P4P: Explicit Communications for Cooperative Control between P2P and Network Providers", *P4PWG Whitepaper*, Vol. 5, pp. 1-7, 2007.

[17] S. Buchegger and A. Datta, "A Case for P2P Infrastructure for Social Networks-Opportunities and Challenges", *Proceedings of the International Conference on Wireless on-Demand Network Systems and Services*, pp. 161-168, 2009.

[18] J. Yuan, J. Peng, Q. Yan, G. He, H. Xiang and Z. Liu, "Deep Reinforcement Learning-based Energy Consumption Optimization for Peer-to-Peer (P2P) Communication in Wireless Sensor Networks", *Sensors*, Vol. 24, No. 5, pp. 1-15, 2024.