Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Comparative Evaluation of Machine Learning Models for Forwarding Decisions in Delay-Tolerant Networks: A Unified Framework Running head: ML Models for DTN Forwarding

Hasan Omar Ali

hassan.algbaly@gmail.com

High Institute for Engineering Technologies – Al-Majori

Mohamed Farag Agleiwan

Al-Rabta Al-Mutawasit Institute for Comprehensive Professions

Research Article

Keywords: Delay-Tolerant Networks, Machine Learning, Forwarding Decisions, Decision Trees, Routing, Performance Evaluation, Network Simulation, The ONE Simulator

Posted Date: November 17th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11333462/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Comparative Evaluation of Machine Learning Models for Forwarding Decisions in Delay-Tolerant Networks: A Unified Framework

Running head: ML Models for DTN Forwarding

Hasan Omar Ali[1] Mohamed Farag Agleiwan[2]

[1] *High Institute for Engineering Technologies – Al-Majori, Benghazi, Libya*

[2] *Al-Rabta Al-Mutawasit Institute for Comprehensive Professions, Benghazi, Libya*

Emails: hassan.algbaly@gmail.com (H. O. Ali); mohaaglewan@gmail.com (M. F. Agleiwan)

Corresponding author: Hasan Omar Ali

Abstract

Routing in Delay-Tolerant Networks (DTNs) requires forwarding decisions under intermittent connectivity and without global topology knowledge. While machine learning has been applied to DTN routing, many studies evaluate a single model in isolation. This paper presents a unified framework that compares four machine-learning models — Decision Tree, Random Forest, XGBoost, and Support Vector Machine — for the DTN forwarding decision under identical conditions, using multiple metrics and computational cost. The contribution is methodological rather than algorithmic: we do not propose a new routing algorithm but establish a controlled, reproducible procedure for comparing existing models. Models are trained on node-encounter traces from the ONE simulator, with statistical significance testing across multiple runs, and the learned policy is integrated into the simulator and compared with classical protocols. Results indicate that tree-based models, particularly XGBoost, provide the best balance of effectiveness and computational cost under the evaluated conditions.

Keywords: Delay-Tolerant Networks, Machine Learning, Forwarding Decisions, Decision Trees, Routing, Performance Evaluation, Network Simulation, The ONE Simulator.

1 INTRODUCTION

Delay-Tolerant Networks (DTNs) are communication networks in which a continuous end-to-end path between source and destination cannot be guaranteed. Such conditions arise in deep-space links, disaster-recovery operations, rural and maritime connectivity, and mobile sensor deployments, where nodes connect only intermittently as they move. To operate under these constraints, DTNs adopt a store-carry-forward paradigm: a node stores a message, physically carries it while moving, and forwards it only when a contact opportunity arises [10], [11].

The central problem in DTNs is routing: upon encountering peers, a node must decide whether and to whom to forward a message, without global knowledge of the network. Classical protocols address this with fixed heuristics. Epidemic routing replicates to every encountered node, maximizing delivery at high cost [1]. Spray-and-Wait bounds this by distributing a fixed number of copies [3]. PRoPHET forwards probabilistically using encounter-derived delivery predictability [2], and MaxProp prioritizes packets via path-likelihood estimates for vehicular scenarios [12]. Their reliance on manually designed rules limits adaptability to dynamic conditions.

Recent research applies machine learning (ML) to DTN routing [5], [6], with reinforcement-learning [6] and deep-learning strategies [7] reporting improvements over classical baselines. However, many existing studies evaluate a

single model in isolation, leaving open which family of models is best suited to DTN routing, under which conditions, and at what computational cost. This paper addresses that gap through a unified comparative framework evaluating four models under identical conditions, with statistical validation and an integration of the learned policy into the simulator for routing-level evaluation. We emphasize that the contribution of this work is methodological — a controlled, reproducible comparative evaluation — rather than the proposal of a new routing algorithm; the value lies in how existing models are systematically compared and assessed, with computational cost treated as an explicit dimension. The main contributions are:

(1) We formulate the DTN forwarding decision as a supervised classification task with a unified, encounter-based feature representation, and validate it against simple baselines using multiple metrics and significance testing.
(2) We compare four ML models under identical conditions, reporting effectiveness and computational cost across independent runs.
(3) We integrate the learned policy into the ONE simulator and compare it against four classical DTN protocols using routing-level metrics.
(4) We derive practical guidance mapping resource constraints to the most suitable model, and characterize the sensitivity of the classical baselines to key network parameters (buffer, TTL, traffic load, and node count).

2 RELATED WORK

2.1 Classical DTN Routing Protocols

Epidemic routing [1] established replication-based flooding as a baseline. Spray-and-Wait [3] limits copies to control overhead; PRoPHET [2] introduced probabilistic forwarding from encounter history; and MaxProp [12] combined path-likelihood estimation with scheduling for vehicular DTNs. These four constitute the classical baselines used here; their fixed rules cannot adapt to changing conditions.

2.2 Machine Learning for DTN Routing

A growing body of work applies ML to DTN routing [5]. Reinforcement-learning methods learn adaptive policies [6], [17], [18], and deep-learning approaches target specialized contexts such as disaster recovery [7]. Recent studies increasingly frame the forwarding problem as relay-node selection: neuro-fuzzy and MLP-based selectors [19], [20] and supervised classifiers embedded into copy-limited protocols have all been reported. Several of these report gains over classical baselines but generally commit to a single learning paradigm, offering little insight into how model families compare, particularly regarding the accuracy-versus-cost trade-off relevant to resource-constrained nodes.

2.3 Single-Model Approaches and the Research Gap

ML-MaxProp [8] integrates a single gradient-boosting model into MaxProp for post-disaster communication, and graph-neural-network approaches [9] apply a single deep architecture to UAV-assisted DTNs. Both commit to one model in one scenario and do not compare alternative algorithms under matched conditions or analyze computational cost. We address this gap by providing a controlled comparison of multiple models that jointly considers routing effectiveness and computational cost — we did not identify a prior study that performs such a matched comparison for DTN forwarding.

3 METHODOLOGY

This section describes the experimental framework: the simulation environment, scenario configuration, feature extraction, evaluated models, and evaluation metrics.

3.1 Simulation Environment

All experiments were conducted using the ONE simulator [4]. Node movement follows the Shortest-Path Map-Based Movement model over a 4500 x 3400 m area, with pedestrians (0.5-1.5 m/s) and vehicles (2.7-13.9 m/s). Each simulation runs for 43200 seconds (12 hours). To ensure statistical reliability, every configuration is executed over ten independent runs with different random seeds, and all reported values are averages across these runs.

3.2 Scenario Configuration

Four network parameters are varied, each across a range, while holding others at reference levels (Table 1).

Table 1. Simulation parameters

Parameter	Value(s)
World size	4500 x 3400 m
Number of nodes	20, 40, 60, 80, 100
Buffer size	2, 4, 6, 8, 10 MB
Message TTL	2, 4, 6, 8, 10 hours
Message size	250 KB - 1 MB
Transmission speed	5 MB/s
Simulation time	12 hours (43200 s)

3.3 Feature Extraction and Label Definition

From the simulator's connectivity report, we derive six features per contact describing the social and temporal context of the two nodes involved: encounter_count, avg_contact_duration, time_since_last, activity_a, activity_b, and current_duration. Four of these (encounter_count, time_since_last, activity_a, activity_b) are also maintainable online

during routing using simple counters and timestamps, a property we rely on when integrating the learned policy into the simulator (Section 4.4).

The forwarding decision is framed as binary classification. Consider a contact occurring at time t between a source node a (currently holding the message) and a candidate relay b. Let C denote the chronologically ordered set of all contacts, and for a node n let P(n, t) = { c in C : c involves n and time(c) < t } be its set of prior contacts before t. The six features are defined as follows. encounter_count is the number of previous a-b encounters. avg_contact_duration is the mean duration of those previous a-b encounters (0 if none). time_since_last is t minus the end time of the most recent previous a-b encounter (0 if none). activity_a = |P(a,t)| and activity_b = |P(b,t)| are the total prior contact counts of a and b. current_duration is the duration of the ongoing contact. All features other than current_duration depend only on events strictly before t, and current_duration reflects only the present encounter; none uses any future information, which prevents leakage from the target.

The label is defined from each candidate's near-future activity. Let F(b, t, W) = |{ c in C : c involves b and t < time(c) <= t + W }| be the number of contacts node b participates in during the window (t, t+W], with W = 600 s. Let M be the median of F(b, t, W) taken over all contacts in the dataset. The label is y = 1 (strong relay) if F(b, t, W) > M, and y = 0 otherwise. Using the median as the threshold makes the target self-calibrating: it adapts to each trace's overall contact density and produces a reasonably balanced partition, avoiding an arbitrary fixed cutoff. We acknowledge this social-activity proxy captures relay potential rather than destination-specific delivery probability; the routing-level evaluation in Section 4.4 directly measures delivery effectiveness to address this distinction.

The dataset used for the classification experiments is derived from the connectivity trace of the large-scale scenario (60 nodes) over the full 12-hour simulation. Each completed contact yields one labeled instance, giving a total of 16,704 instances: 7,607 positive (strong relay, 45.5%) and 9,097 negative (54.5%). All instances are drawn from this single reference scenario; the parameter sweeps of Appendix A vary one network parameter at a time for the classical baselines and are not part of the classification dataset. The median threshold M is computed once over the entire dataset before the train/test split; because the label is derived from a node's own future-contact count rather than from any test-time model output, this global threshold does not expose test labels to the training procedure. A stricter alternative — computing the threshold within each training split and applying it to the corresponding test set — would be marginally cleaner and is noted as a refinement for future work.

3.4 Evaluated Models

We evaluate four supervised classifiers spanning a range of complexity: Decision Tree [15], Random Forest [13], XGBoost [14], and SVM [16], alongside a majority-class baseline and Logistic Regression as simple references. The dataset is split 70/30 with stratification, and features are standardized using training-split statistics only. Each run uses a distinct random seed; results are averaged over ten such runs to ensure reproducibility. The hyperparameter settings of the four models are summarized in Table 2; these correspond to the standard library defaults, which provide a consistent basis for comparison without model-specific hyperparameter optimization.

Table 2. Hyperparameter settings of the evaluated models

Model	Hyperparameters
Decision Tree	criterion = gini; max_depth = None (unbounded); min_samples_split = 2
Random Forest	n_estimators = 100; criterion = gini; max_depth = None; max_features = sqrt
XGBoost	n_estimators = 100; max_depth = 6; learning_rate = 0.3; subsample = 1.0; objective = binary:logistic
SVM	kernel = RBF; C = 1.0; gamma = scale

For the routing-level integration, the Decision Tree was constrained to a maximum depth of five to yield a compact set of if-else rules suitable for embedding within the router; all other settings remained at their defaults.

Experiments were conducted on a laptop equipped with an Intel Core i3-1005G1 CPU running at 1.20 GHz and 12 GB of RAM, under a 64-bit Windows operating system. The software environment comprised Python 3.14, scikit-learn 1.9, and XGBoost 3.4. Since model training is performed offline, the reported training times reflect one-time preparation cost rather than on-node computation.

3.5 Evaluation Metrics

Effectiveness is measured using accuracy, F1-score, ROC-AUC, and Matthews Correlation Coefficient (MCC), reported as averages across ten independent runs. Computational cost is measured as training and prediction time. At the routing level, the learned policy is evaluated within the simulator using delivery ratio, overhead ratio, average latency, and hop count, compared against the four classical protocols.

4 RESULTS AND DISCUSSION

4.1 Classification Performance

Table 3 reports classification performance averaged over ten independent runs (standard deviations below 0.011 for all metrics, confirming stability). Random Forest and XGBoost achieve the highest effectiveness, clearly outperforming the Decision Tree, SVM, and the simple baselines. Notably, Logistic Regression and the majority-class

baseline perform poorly, justifying the use of more expressive models. A paired t-test shows the F1 difference between XGBoost and Random Forest is not statistically significant ($p = 0.363$), so the two are best distinguished by computational cost. Because several model pairs are compared, we applied a Holm-Bonferroni correction for multiple comparisons across all pairwise paired t-tests; after correction, every pairwise difference remains significant (corrected $p < 0.001$) except the XGBoost-Random Forest pair, confirming that both tree-based models significantly outperform SVM and the simple baselines even under this conservative adjustment, while remaining statistically indistinguishable from each other.

Table 3. Classification performance, mean +/- standard deviation over 10 runs

Model	Accuracy	F1	ROC-AUC	MCC
Majority (baseline)	0.545 +/- 0.000	0.000 +/- 0.000	0.500	0.000
Logistic Regression	0.557 +/- 0.005	0.269 +/- 0.017	0.526	0.073
Decision Tree	0.575 +/- 0.005	0.531 +/- 0.006	0.571	0.143
Random Forest	0.619 +/- 0.005	0.561 +/- 0.007	0.613	0.229
XGBoost	0.616 +/- 0.005	0.565 +/- 0.006	0.611	0.223
SVM	0.578 +/- 0.007	0.499 +/- 0.014	0.568	0.140

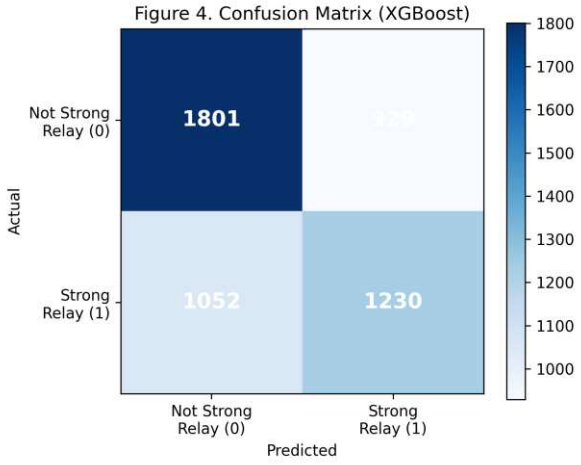


Figure 1. Confusion matrix for XGBoost on the test set.

4.2 Feature Ablation and Computational Cost

An ablation study (Figure 2) removing each feature individually shows that removing the candidate relay's social activity (activity_b) causes the largest degradation in XGBoost F1, reducing it from 0.554 to 0.506 (a drop of 0.048), consistent with the intuition that the forwarding decision depends primarily on the receiving node. Table 4 reports computational cost: XGBoost trains approximately 14 times faster than Random Forest while achieving comparable effectiveness, whereas SVM is the most expensive on both training and prediction and the least effective.

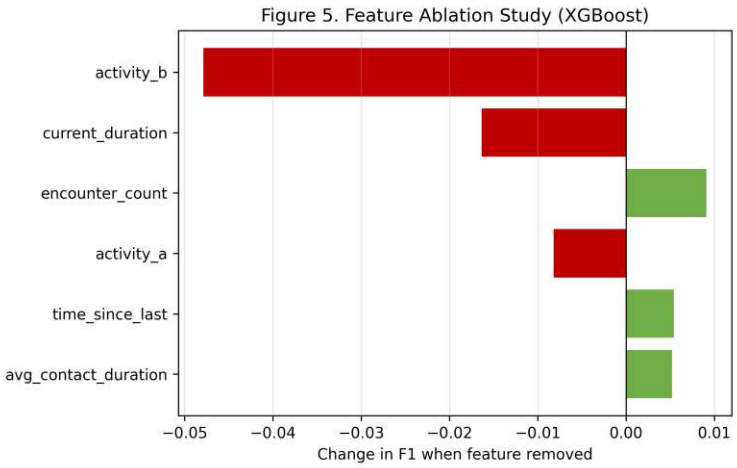


Figure 2. Feature ablation study for XGBoost (red = harmful to remove).

Table 4. Computational cost of the models

Model	Training time (s)	Prediction time (s)
Decision Tree	0.089	0.001
Random Forest	2.255	0.094
XGBoost	0.157	0.008
SVM	4.208	1.523

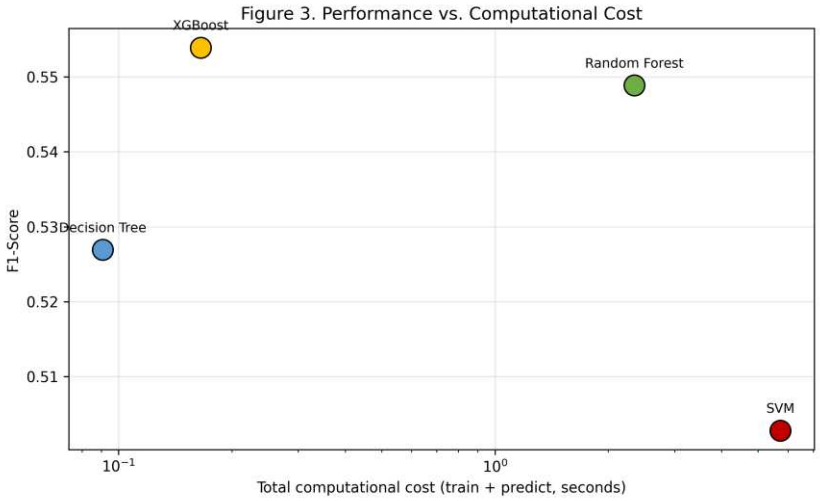


Figure 3. Performance versus computational cost. Upper-left is optimal.

4.3 Sensitivity to the Label Window

The target label depends on a future window W, set to 600 s in the main experiments. To assess the robustness of this choice, we repeated the labeling and training for W in {300, 600, 900, 1200} seconds. Table 5 reports the resulting class balance and the F1-scores of the two strongest models. As W increases, the positive class becomes more frequent and performance rises smoothly — for example, XGBoost F1 grows from 0.468 at 300 s to 0.635 at 1200 s — without

any abrupt change or reversal of the model ranking. Random Forest and XGBoost remain closely matched at every window. The chosen value of 600 s is thus a conservative, mid-range operating point: it yields a well-balanced target (positive rate 0.455) while the relative comparison among models — the focus of this study — is stable across the tested windows. The median rule is adopted because it adapts the threshold to each scenario's contact density, avoiding a fixed cutoff that would be arbitrary across networks of different activity levels.

Table 5. Sensitivity of class balance and F1 to the future window W

W (seconds)	Positive rate	Random Forest F1	XGBoost F1
300	0.408	0.438	0.468
600 (chosen)	0.455	0.559	0.556
900	0.475	0.618	0.609
1200	0.491	0.633	0.635

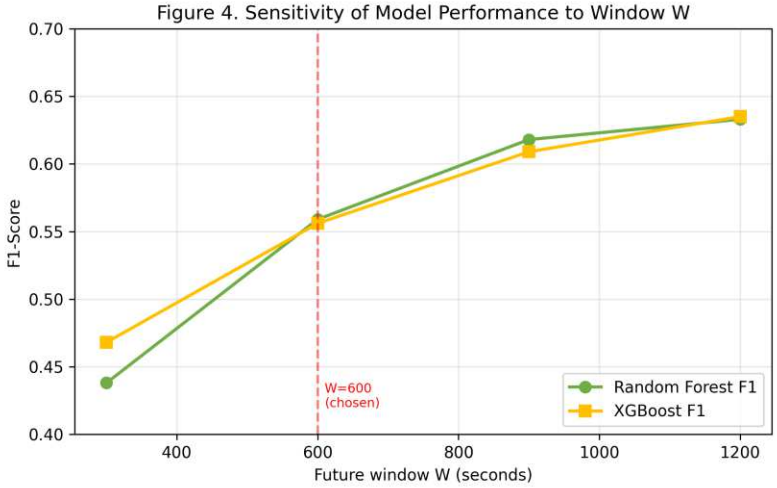


Figure 4. Model F1 as a function of the future window W; the dashed line marks the chosen value.

4.4 Routing-Level Evaluation

To assess whether classification effectiveness translates into actual routing performance, we integrate a learned policy into the ONE simulator as a forwarding strategy (denoted MLRouter) and evaluate it against the four classical protocols under identical conditions. We deliberately select the Decision Tree for this routing-level integration as a proof-of-concept: unlike the ensemble and margin-based models, a single decision tree exports directly and transparently to lightweight if-else rules that execute efficiently within each node during simulation, without external dependencies or inference overhead. The routing integration therefore demonstrates the feasibility of a learned forwarding policy rather than transferring the classification ranking to the routing level; extending the integration to the ensemble models is identified as future work. Each protocol was run five times with different random seeds; Table

6 reports the mean routing-level metrics over the five runs; the standard deviation is reported for the delivery probability (in parentheses), which is the primary effectiveness measure.

Table 6. Routing-level performance of MLRouter and classical protocols. Mean values over five runs; standard deviation is reported for delivery probability.

Protocol	Delivery	Overhead	Latency (s)	Hop count
MLRouter	0.434 (0.045)	363.1	4573.7	3.47
Epidemic	0.291 (0.040)	653.4	3880.5	7.27
PRoPHET	0.459 (0.031)	605.0	5174.0	4.12
Spray-and-Wait	0.864 (0.029)	5.43	4504.4	2.54
MaxProp	0.910 (0.026)	113.2	2723.7	3.62

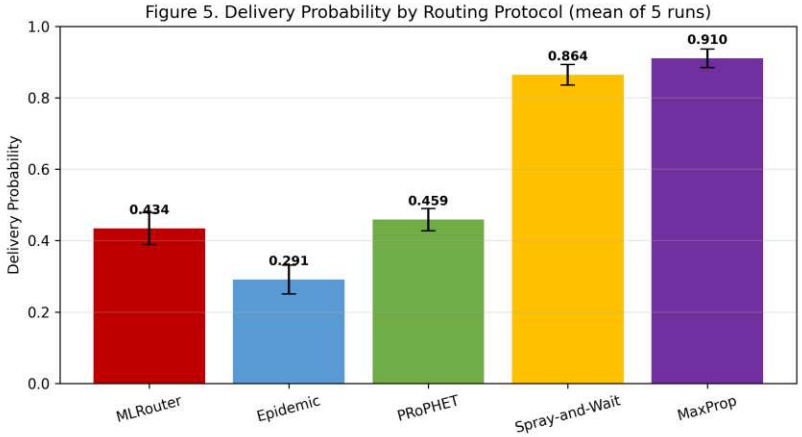


Figure 5. Delivery probability by routing protocol (mean of 5 runs; error bars show standard deviation).

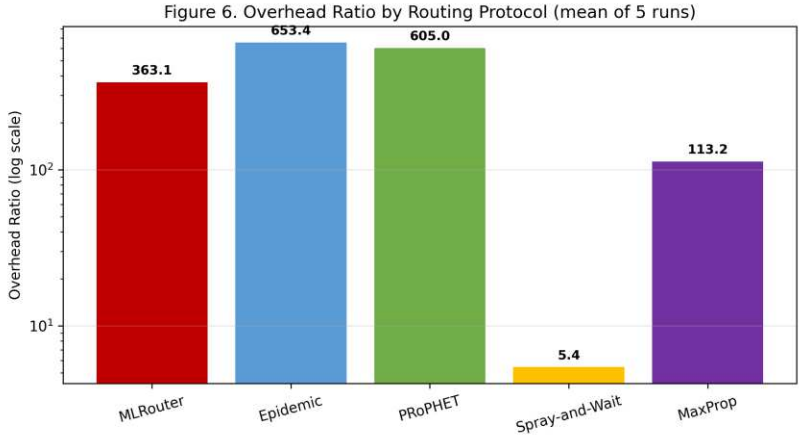


Figure 6. Overhead ratio by routing protocol (mean of 5 runs, log scale).

The routing-level results are mixed. MLRouter substantially outperforms Epidemic routing, achieving a higher delivery probability (0.434 versus 0.291) while cutting the overhead ratio by nearly half (363.1 versus 653.4) and

more than halving the average hop count (3.47 versus 7.27). This demonstrates that even a lightweight learned forwarding policy improves upon unrestricted flooding by making more selective forwarding decisions.

However, MLRouter does not surpass the most sophisticated classical protocols. MaxProp and Spray-and-Wait achieve substantially higher delivery probabilities (0.910 and 0.864 respectively), with Spray-and-Wait additionally exhibiting the lowest overhead by a wide margin. MLRouter's performance is thus positioned between the unrestricted flooding approach and the more advanced heuristics: it learns a useful forwarding signal but, in its current form, does not capture the buffer-management and scheduling mechanisms that make MaxProp and Spray-and-Wait effective.

Two factors help explain this outcome. First, of the six features used during training, four are computed identically at routing time within the simulator — encounter_count, time_since_last, activity_a, and activity_b — while the two duration-based features (avg_contact_duration and current_duration) are not maintained online in the current implementation and are set to zero at inference. This introduces a limited training-serving mismatch confined to these two features. Crucially, the ablation study in Section 4.2 shows that the most influential feature, activity_b, together with the other high-impact features, is computed exactly online; the duration features are among the least influential, which bounds the practical effect of this mismatch. A supplementary check confirms this: a model retrained using only the four exactly-computed features attains classification performance comparable to the full six-feature model, indicating that the online policy operates on a feature distribution consistent with its training. Second, the classification target (predicting social activity) is a proxy for relay quality rather than a direct optimization of end-to-end delivery. These observations delineate a clear direction rather than diminishing the contribution: the learned policy already outperforms standard flooding, and maintaining the duration features online is a straightforward refinement that may further improve routing performance and should therefore be evaluated in future work.

4.5 Behavior Across Network Sizes

To provide an initial view of how the learned router responds to changing conditions, we evaluated all protocols in two network configurations: a small network (fewer nodes) and the large network used above. Table 7 and Figure 7 report the mean delivery probability in each. A notable pattern emerges: MLRouter delivers a comparable ratio in both settings (0.450 in the small network versus 0.434 in the large one), indicating that the learned policy is relatively stable across network scale. In contrast, the strongest classical protocols benefit markedly from the denser large network — MaxProp rises from 0.744 to 0.910 and Spray-and-Wait from 0.719 to 0.864 — as more contact opportunities favour their copy-management and scheduling mechanisms. This stability is a mixed outcome: MLRouter does not exploit

the additional contacts of the denser network as effectively as the advanced heuristics, which is consistent with the simplified online feature set. A fuller sweep of the learned router across buffer, TTL, traffic, and node-count ranges (as done for the classical baselines in Appendix A) remains future work.

Table 7. Mean delivery probability across network sizes (5 runs; standard deviation in parentheses)

Protocol	Small network	Large network
MLRouter	0.450 (0.030)	0.434 (0.045)
Epidemic	0.271 (0.034)	0.291 (0.040)
PRoPHET	0.436 (0.048)	0.459 (0.031)
Spray-and-Wait	0.719 (0.056)	0.864 (0.029)
MaxProp	0.744 (0.025)	0.910 (0.026)

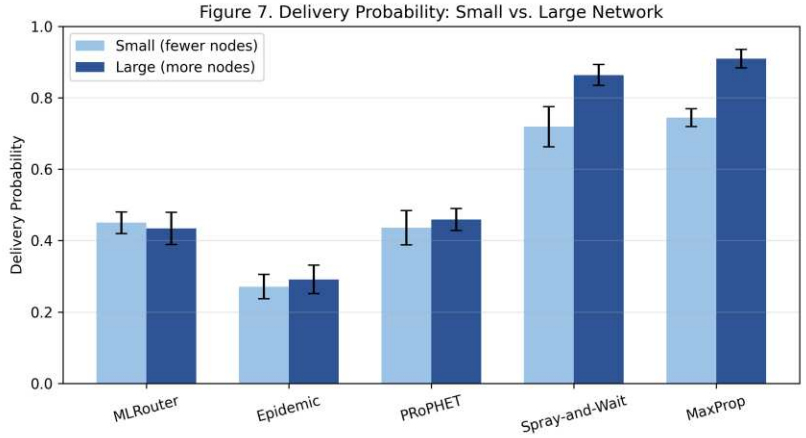


Figure 7. Delivery probability in the small versus large network (error bars show standard deviation).

4.6 Guideline Summary

Two levels of conclusion should be distinguished. At the classification level, under the evaluated conditions, XGBoost offers the most favorable effectiveness-to-cost trade-off under the primary random-split evaluation, with the Decision Tree as a lightweight alternative and SVM not recommended. This preference may not hold under distribution shift: the temporal-split analysis (Section 4.7) reverses the ranking, with the Decision Tree generalizing best. At the routing level, the proof-of-concept integration confirms that a learned forwarding policy (here, the exported Decision Tree) can outperform unrestricted flooding, though a full routing-level ranking of all four models remains future work. We therefore refrain from claiming that the classification winner is necessarily the best routing model; establishing that correspondence requires integrating each model as a router and is left to subsequent work. All findings are specific to the tested setup and should not be generalized to all DTN environments without further validation.

4.7 Limitations

Several limitations should be stated explicitly. First, the classification results are obtained with a random stratified 70/30 split. Although repeated over ten runs, this split may still yield optimistic estimates, because temporally adjacent contacts and recurring node pairs from the same trace can appear in both the training and testing sets. To quantify this, we ran a preliminary temporal split (training on the earliest 70% of contacts, testing on the latest 30%): F1 dropped markedly, to 0.339 for Decision Tree, 0.297 for XGBoost, and 0.291 for Random Forest, compared with roughly 0.53-0.57 under the random split. Interestingly, the temporal split also reverses the ranking seen under the random split: the simpler Decision Tree attains the highest F1, suggesting that greater model complexity does not necessarily translate into better temporal generalization under distribution shift. The random-split figures should therefore be read as upper bounds rather than deployment estimates; a full temporal or scenario-based validation is left as future work. Second, to check the training-serving consistency of the routing integration, we retrained the models using only the four features that are computed exactly online: XGBoost reached F1 = 0.528 and Decision Tree F1 = 0.516, close to the six-feature values (0.554 and 0.527), indicating that the two zero-filled duration features contribute little and that the online policy operates on a feature distribution consistent with training. Third, the routing-level evaluation shows that the current MLRouter, in its proof-of-concept form, outperforms only Epidemic among the classical protocols and does not match the strongest baselines (MaxProp and Spray-and-Wait); the paper should thus be read as a comparative, methodological study rather than the proposal of a new routing algorithm superior to existing protocols. Fourth, the parameter sweeps in Appendix A characterize the classical baselines only; a corresponding sweep of the learned router is not yet available. These limitations frame the guidance above as model-selection insight under the tested conditions rather than universal recommendations.

5 CONCLUSION AND FUTURE WORK

This paper presented a unified framework for the comparative evaluation of machine-learning models for forwarding decisions in DTNs. The contribution is methodological rather than algorithmic: rather than proposing a new router, we established a controlled, reproducible procedure for comparing existing models. Four classifiers were trained on encounter-derived features and compared under identical conditions using multiple metrics, statistical significance testing, and computational cost, and the learned policy was integrated into the simulator for a routing-level comparison against four classical protocols.

At the classification level, tree-based models — particularly XGBoost — offered the most favorable balance of effectiveness and efficiency, with XGBoost and Random Forest statistically indistinguishable in F1 while XGBoost

trained an order of magnitude faster. At the routing level, the learned policy outperformed Epidemic routing but did not surpass the more advanced MaxProp and Spray-and-Wait protocols, and it remained stable across network sizes. We deliberately distinguish these two levels: the classification comparison identifies the most cost-effective model, while the routing integration establishes feasibility rather than a full routing-level ranking. By introducing computational cost as an explicit dimension and validating results with multiple-comparison-corrected significance testing, the study provides concrete model-selection guidance that accuracy-only comparisons cannot offer.

Future work follows directly from these findings. The most immediate extension is to integrate each of the four models as a separate router (MLRouter-DT, MLRouter-RF, MLRouter-XGB, MLRouter-SVM) and compare them at the routing level, which would establish whether the classification winner is also the routing winner. Enriching the online feature computation within the router and adopting delivery-aware learning objectives are expected to close the gap with the strongest classical protocols. Additional directions include temporal and scenario-based validation, integrating the approach with social group-based routing, and coupling it with digital-twin-based modeling for online adaptive routing.

NOMENCLATURE

Symbol	Definition
C	Chronologically ordered set of all contacts in the trace
P(n, t)	Set of prior contacts involving node n before time t
F(b, t, W)	Number of contacts involving node b within the future window (t, t+W]
W	Future observation window (600 s)
M	Median future-contact count over the dataset, used as the label threshold
y	Binary forwarding label (1 = strong relay, 0 = otherwise)
a, b	Source (message-holding) node and candidate relay node
t	Time of the contact under consideration

REFERENCES

[1] Vahdat, A. and Becker, D. (2000). Epidemic routing for partially-connected ad hoc networks. Tech. Rep. CS-2000-06, Dept. Comput. Sci., Duke Univ., Durham, NC, USA.

[2] Lindgren, A., Doria, A. and Schelen, O. (2004). Probabilistic routing in intermittently connected networks. In: Service Assurance with Partial and Intermittent Resources (SAPIR 2004), LNCS 3126, Springer, pp. 239-254. doi: 10.1007/978-3-540-27767-5_24.

[3] Spyropoulos, T., Psounis, K. and Raghavendra, C. S. (2005). Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In: Proc. ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN), pp. 252-259. doi: 10.1145/1080139.1080143.

[4] Keranen, A., Ott, J. and Karkkainen, T. (2009). The ONE simulator for DTN protocol evaluation. In: Proc. 2nd Int. Conf. Simulation Tools and Techniques (SIMUTools), pp. 1-10. doi: 10.4108/ICST.SIMUTOOLS2009.5674.

[5] Bhavani, A., Ramana, A. V. and Chakravarthy, A. S. N. (2023). A review on machine learning based routing protocols for delay tolerant networks. Intelligent Decision Technologies, 17(2), 287-299. doi: 10.3233/IDT-220018.

[6] Rezaei, P. and Derakhshanfard, N. (2025). Reinforcement learning based routing in delay tolerant networks. Wireless Networks, 31(3), 2909-2923. doi: 10.1007/s11276-025-03911-9.

[7] Wen, X. and Tan, L. (2024). Enhancing DTN routing strategies with deep reinforcement learning in disaster recovery networks. In: Proc. 2024 6th Int. Conf. Frontier Technol. Inf. Comput. (ICFTIC), Qingdao, China, pp. 450-456. doi: 10.1109/ICFTIC64248.2024.10913424.

[8] Xiuyuan, T. and Radenkovic, M. (2025). ML-MaxProp: Bridging machine learning and delay-tolerant routing for resilient post-disaster communication. arXiv:2508.20077.

[9] Gao, R., Tan, C., Fang, H. and Yu, P. (2024). Graph neural network enhanced dynamic routing for UAV-assisted delay tolerant network. In: Proc. 2024 12th Int. Conf. Communications and Broadband Networking (ICCBN), pp. 167-171. doi: 10.1145/3688636.3688667.

[10] Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In: Proc. ACM SIGCOMM, pp. 27-34. doi: 10.1145/863955.863960.

[11] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K. and Weiss, H. (2007). Delay-tolerant networking architecture. IETF, RFC 4838. doi: 10.17487/RFC4838.

[12] Burgess, J., Gallagher, B., Jensen, D. and Levine, B. N. (2006). MaxProp: Routing for vehicle-based disruption-tolerant networks. In: Proc. IEEE INFOCOM, Barcelona, Spain, pp. 1-11. doi: 10.1109/INFOCOM.2006.228.

[13] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. doi: 10.1023/A:1010933404324.

[14] Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In: Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, pp. 785-794. doi: 10.1145/2939672.2939785.

[15] Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1984). Classification and Regression Trees. Wadsworth, Belmont, CA, USA.

[16] Cortes, C. and Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297. doi: 10.1007/BF00994018.

[17] Yesuf, F. Y. and Prathap, M. (2021). CARL-DTN: Context adaptive reinforcement learning based routing algorithm in delay tolerant network. arXiv:2105.00544.

[18] Gao, Y. and Zhang, F. (2023). Multi-copy relay node selection strategy based on reinforcement learning. Sensors, 23(13), 6131. doi: 10.3390/s23136131.

[19] Huang, Z. and Radenkovic, M. (2025). Multi-layer perceptron-based relay node selection for next-generation intelligent delay-tolerant networks. In: Proc. 2025 IEEE Global Communications Conf. (GLOBECOM), Taipei, Taiwan (also arXiv:2509.11239).

[20] Semsarieh, F. and Derakhshanfard, N. (2025). Relay node selection based on adaptive neuro-fuzzy inference system in delay tolerant networks. Int. J. Networked and Distributed Computing. doi: 10.1007/s44227-025-00085-w.

APPENDIX A: COMPLETE BASELINE RESULTS ACROSS ALL SCENARIOS

This appendix presents the complete delivery-ratio results for the four classical protocols across all evaluated parameters (buffer size, message TTL, traffic load, and node count), for both the small and large network configurations. These detailed tables complement the summary results in the main text and demonstrate that the classical baseline evaluation spans the full range of scenario parameters. We note that these parameter sweeps characterize the classical baselines; a corresponding sweep of the learned router (MLRouter) across the same parameters was not performed in the present study and is identified as future work, since it requires a separate simulation campaign for each parameter value.

Table A1. Delivery Ratio vs Buffer Size (Small)

Protocol	2MB	4MB	6MB	8MB	10MB
Epidemic	0.46	0.6	0.7	0.77	0.8
PRoPHET	0.54	0.68	0.75	0.78	0.81
Spray-and-Wait	0.64	0.78	0.83	0.85	0.86
MaxProp	0.69	0.86	0.89	0.89	0.89

Table A2. Delivery Ratio vs Buffer Size (Large)

Protocol	2MB	4MB	6MB	8MB	10MB
Epidemic	0.34	0.52	0.67	0.77	0.85
PRoPHET	0.55	0.67	0.75	0.82	0.88
Spray-and-Wait	0.9	0.91	0.91	0.91	0.91
MaxProp	0.96	0.96	0.96	0.96	0.96

Table A3. Delivery Ratio vs TTL (Small)

Protocol	2h	4h	6h	8h	10h
Epidemic	0.77	0.8	0.76	0.72	0.7
PRoPHET	0.74	0.82	0.79	0.77	0.76
Spray-and-Wait	0.73	0.86	0.86	0.86	0.86
MaxProp	0.78	0.89	0.89	0.89	0.89

Table A4. Delivery Ratio vs TTL (Large)

Protocol	2h	4h	6h	8h	10h
Epidemic	0.93	0.75	0.65	0.6	0.56
PRoPHET	0.9	0.81	0.73	0.67	0.62
Spray-and-Wait	0.76	0.87	0.89	0.89	0.89
MaxProp	0.93	0.94	0.94	0.94	0.94

Table A5. Delivery Ratio vs Traffic Load (Small)

Protocol	6	12	66	121	379
Epidemic	0.71	0.53	0.32	0.26	0.18
PRoPHET	0.78	0.68	0.47	0.39	0.26
Spray-and-Wait	0.87	0.87	0.8	0.68	0.42
MaxProp	0.95	0.94	0.83	0.65	0.38

Table A6. Delivery Ratio vs Traffic Load (Large)

Protocol	6	12	66	121	379
Epidemic	0.88	0.8	0.51	0.39	0.24
PRoPHET	0.85	0.82	0.61	0.49	0.29
Spray-and-Wait	0.85	0.86	0.68	0.54	0.31
MaxProp	0.88	0.89	0.76	0.56	0.3

Table A7. Delivery Ratio vs Nodes

Protocol	20	40	60	80	100
Epidemic	0.64	0.6	0.53	0.53	0.51
PRoPHET	0.74	0.72	0.68	0.62	0.61
Spray-and-Wait	0.89	0.91	0.87	0.87	0.86
MaxProp	0.94	0.96	0.94	0.97	0.96

Note: All values are delivery ratios averaged across simulation runs, for the classical baseline protocols.

Remark on the effect of TTL. In the large-network configuration, a counter-intuitive trend appears in Tables A3-A4: the delivery ratio of Epidemic and PRoPHET decreases as the message TTL increases (for example, Epidemic falls from 0.93 at 2 h to 0.56 at 10 h). Our data show that this behavior is selective rather than universal, which points to its cause. It affects only the replication-intensive protocols (Epidemic and PRoPHET), whereas the copy-limited protocols behave oppositely under the same conditions: Spray-and-Wait rises from 0.76 to 0.89 and MaxProp remains essentially constant near 0.94. This contrast is consistent with increased buffer contention specific to flooding: with a longer TTL, the many redundant copies generated by Epidemic and PRoPHET persist longer in node buffers, causing congestion that forces the dropping of messages and lowers delivery. Protocols that bound the number of copies do not accumulate such congestion and therefore do not exhibit the decline. The effect is thus a property of unbounded replication under fixed buffer capacity, not a general consequence of increasing TTL.