Predicting IoT Network Congestion with Artificial Intelligence Techniques

Abstract

Imagine IoT networks that can foresee congestion—and act before it hits. We introduce a practical, industry-oriented approach to proactive congestion control. Using an encoder–decoder LSTM (ED-LSTM), we accurately *predict the loss ratio*—the fraction of packets lost over packets sent—so that gateways can adjust rate, priorities, or protocol knobs *before* congestion becomes visible to applications. On realistic simulations with Cooja/Contiki (6LoWPAN/IPv6, RPL, UDP/CoAP, IEEE 802.15.4), ED-LSTM consistently outperforms LSTM, GRU, RNN, CNN, and Bi-LSTM in terms of RMSE, and generalizes across topologies. This enables *proactive* QoS preservation for critical use cases such as healthcare monitoring and industrial sensing.

Summary

Predicting packet losses tens of seconds ahead lets us **act early** (rate shaping, prioritization, route/MAC adjustments), avoid queue build-ups, and **preserve QoS** without overprovisioning.

1 Why Predict the Loss Ratio?

The loss ratio is a robust, application-agnostic signal of network stress in constrained IoT environments (low bandwidth, duty-cycled radios, lossy links). When loss spikes from 0.2 to 0.6 in remote patient monitoring, the impacts are immediate: missing vitals, delayed alerts, and potential safety issues. If we can *predict* that spike, we can *prevent* it.

2 Data and Problem Setup

We emulate a 100-node grid: 98 CoAP senders, one CoAP server, and one RPL border router. Each sender transmits every 10 s (112 B request / 86 B reply). We extract a global loss ratio every 10 s over 10,000 intervals from Cooja/Contiki logs (6LoWPAN/IPv6, RPL, UDP/CoAP, IEEE 802.15.4).

We cast forecasting as supervised learning with a *sliding window*: given the last $m \in \{5, 10, 15, 20\}$ observations of the loss ratio series X_t, predict X_{t+1}. Data are normalized and split 60/40 (train/test). We repeat each configuration 10 times to report stable estimates.

3 Models

We compare: basic RNN, GRU, LSTM; advanced RNN (Bi-LSTM, Encoder–Decoder LSTM); and 1D CNN. Hyperparameters (layers, hidden units, activations) are tuned via grid search.

Key finding. **ED-LSTM** is consistently best in RMSE, with $\sim$0.08 train and < 0.10 test across window sizes, and shows strong generalization on a distinct 14-node ring scenario (RMSE $\sim$0.06 vs. LSTM $\sim$0.07).

4 Applications and Impact

- **Healthcare IoT**: ensure timely delivery of high-priority vitals (e.g., SpO_2, ECG) under rising load.
- **Industry 4.0**: smooth batch peaks, reduce false alarms and downtime.
- **Smart Cities**: protect critical sensors (fire, flood) during network events.

Benefits include lower retransmissions (energy), SLA compliance, better user experience, and avoided infrastructure costs.

5 Integration Blueprint

We propose a lightweight prediction module at the edge (gateway) or cloud:

1. **Learn** from local time series (loss, delay, queue sizes, PDR).
2. **Predict** short-term loss ratio (10–60 s horizon).
3. **Act** via proactive policies:
   - rate/window adaptation,
   - priority scheduling for critical frames,
   - RPL route adjustments or MAC rescheduling,
   - dynamic CoAP parameters (retransmissions, timeouts).

The complete architecture of this workflow is illustrated in Figure 1, which shows how data are collected, preprocessed with a sliding window, fed into the ED–LSTM predictor, and translated into proactive control actions with feedback for continual learning.

Compatibility: CoAP/UDP, RPL/6LoWPAN, IEEE 802.15.4. Progressive adoption without protocol overhaul.

6 Methods at a Glance

Preprocessing: normalization, train/test split, sliding window.
Training: Keras; RMSProp/Adam; 1–2 layers, 20–128 units.
Evaluation: RMSE, MAE, MSE with 95% CIs over 10 runs.
Insight: MAE changes little across models (less sensitive to large errors); RMSE highlights ED-LSTM superiority.

Figure 2 illustrates how ED–LSTM predictions closely follow observed loss ratios while providing short-horizon anticipation and smoothing of congestion spikes.

7 Limitations & Next Steps

Simulation is high-fidelity but controlled; next, pilot deployments (heterogeneous radios, real interference). Enrich features (delay, duty-cycle, queue) and close the loop with an online proactive controller to quantify end-to-end gains (goodput, latency, energy).

Questions and Answers

Why not simple loss thresholds? They react late and ignore dynamics; prediction enables *advance* action.

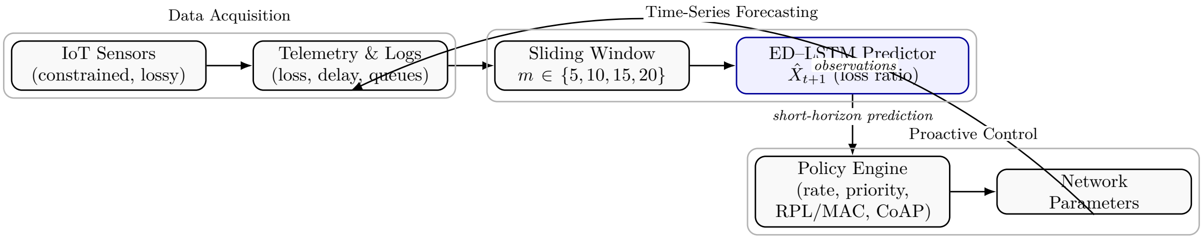

Figure 1: End-to-end pipeline: data → sliding window → ED–LSTM prediction → proactive actions. Feedback closes the loop for continual learning.

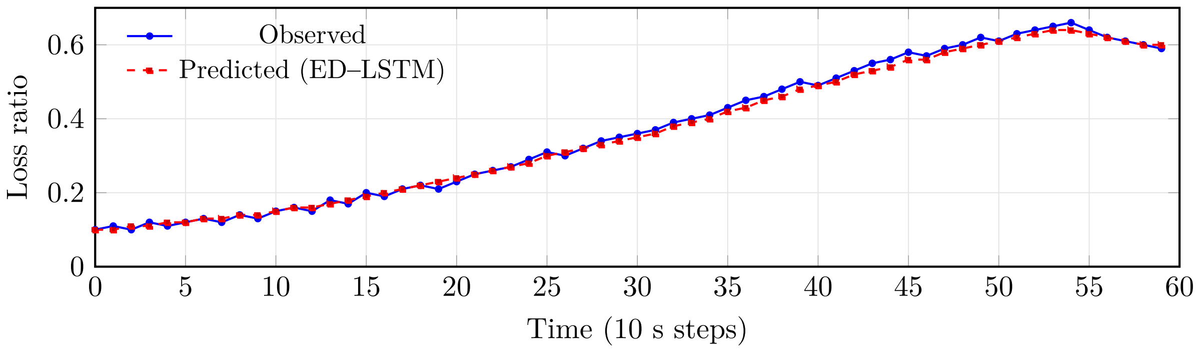

Figure 2: Illustrative observed vs. predicted loss ratio (toy example) showing short-horizon anticipation and smoothing.

Why ED-LSTM? Encoder–decoder captures temporal dependencies and handles variable-length patterns better.
Where does it run? Preferably on gateways (edge) for immediacy; cloud if local resources are tight.

Glossary

- **Loss ratio**: proportion of packets lost over packets transmitted.
- **Sliding window**: method to turn a time series into supervised examples using the m last points.
- **ED-LSTM**: encoder–decoder long short-term memory, a recurrent neural network architecture that models variable-length sequences.
- **RPL**: IPv6 Routing Protocol for Low-power and Lossy Networks.
- **CoAP**: Constrained Application Protocol, a lightweight RESTful protocol over UDP.

Reference

Hanane Benadji et al., Predictive Modeling of Loss Ratio for Congestion Control in IoT Networks Using Deep Learning, presented at IEEE GLOBECOM, 2023. DOI: 10.1109/GLOBECOM54140.2023.10437769