



# IoT Telemetry Protocol



**University of  
East London**

**CSE361 Computer  
Networking**

Presented by:

Ferass Ahmed Mostafa - 23P0304

Ahmed Sadek Mubarak - 23P0140

Abdulrahman Mustafa Sayed - 23P0226

Ahmed Mohamed Elsayed - 23P0035

Kareem Nasrat Abdel Sattar - 1990002

Ahmed Hesham Mohamed - 23p0167

Presented to Eng. Rafik Tamer

Thursday, 25 December 2025

# Table of Contents

Table of Figures .....	1
<b>1. Introduction .....</b>	<b>1</b>
<b>2. Background and Motivation .....</b>	<b>1</b>
<b>3. Methodology .....</b>	<b>1</b>
<b>3.1 System Architecture .....</b>	<b>1</b>
<b>3.2 Protocol Design .....</b>	<b>2</b>
<b>3.3 Baseline Implementation.....</b>	<b>2</b>
<b>3.4 Experimental Setup .....</b>	<b>2</b>
<b>4. Implementation Details .....</b>	<b>2</b>
<b>4.1 Software Tools .....</b>	<b>2</b>
<b>4.2 Execution Workflow .....</b>	<b>3</b>
<b>5. Results and Analysis .....</b>	<b>3</b>
<b>5.1 Baseline Performance Analysis .....</b>	<b>3</b>
<b>5.2 Impact of Network Jitter on Stability and Processing Time.....</b>	<b>4</b>
<b>5.3 Impact of Packet Loss on Stability and Processing Time.....</b>	<b>5</b>
<b>6. Comparison with Baseline.....</b>	<b>5</b>
<b>7. Reproducibility.....</b>	<b>6</b>
<b>8. Limitations .....</b>	<b>6</b>
<b>9. Conclusion.....</b>	<b>6</b>
<b>References .....</b>	<b>7</b>

## Table of Figures

Figure 1: Baseline (1-second intervals) – Processing Time.....	3
Figure 2: Baseline (1-second intervals) – Network Stability.....	3
Figure 3: Jitter (1-second intervals) – Network Stability .....	4
Figure 4: Jitter (1-second intervals) – Processing Time .....	4
Figure 5: Packet Loss (1-second intervals) – Network Stability.....	5
Figure 6: Packet Loss (1-second intervals) – Processing Time .....	5

# 1. Introduction

The rapid growth of Internet of Things (IoT) systems has increased the demand for lightweight, efficient, and reliable communication mechanisms. Many IoT devices run under strict constraints related to bandwidth, processing power, and energy consumption. Although existing application-layer protocols such as MQTT and CoAP are widely adopted, their general-purpose design can introduce unnecessary overhead for simple telemetry applications.

This project presents the design, implementation, and experimental evaluation of a custom IoT telemetry protocol improved for periodic sensor data transmission. The protocol is evaluated under controlled network conditions and compared against a baseline UDP-based implementation. Performance is assessed in terms of network stability and processing time under baseline conditions, network jitter, and packet loss.

The main goals of this project are:

- To design a lightweight telemetry protocol suitable for constrained IoT environments.
  - To experimentally evaluate system behavior under different network impairments.
  - To compare the proposed protocol against baseline implementation.
  - To analyze performance trade-offs and name protocol limitations.
- 

# 2. Background and Motivation

IoT telemetry systems typically involve the periodic transmission of small data packets from distributed devices to a central receiver. In such scenarios, minimizing protocol overhead and ensuring predictable behavior are essential. Network impairments such as jitter and packet loss can significantly degrade system performance, particularly for real-time monitoring applications.

The motivation behind this project is to investigate whether a simplified, custom-designed protocol can support stable performance under adverse network conditions while staying lightweight and easy to implement. By focusing specifically on telemetry use cases, the protocol avoids unnecessary complexity present in general-purpose solutions.

---

# 3. Methodology

## 3.1 System Architecture

The system consists of two primary components:

- **Sender (Client):** Periodically generates telemetry data and transmits packets at fixed one-second intervals.

- **Receiver (Server):** Receives telemetry packets, records arrival times, and measures processing time.

Communication is performed over UDP to minimize transport-layer overhead. Any sequencing or reliability-related logic is managed at the application layer.

---

## 3.2 Protocol Design

The proposed telemetry protocol uses a compact packet structure consisting of:

- A sequence number for basic ordering and loss detection
- A timestamp showing packet generation time.
- A payload holding telemetry data.

This minimal design reduces packet size and simplifies packet parsing at the receiver, making it suitable for constrained IoT devices.

---

## 3.3 Baseline Implementation

A baseline implementation using plain UDP transmission without more protocol-level mechanisms was developed for comparison. The baseline serves as a control scenario and allows direct evaluation of the impact of the proposed protocol design choices.

---

## 3.4 Experimental Setup

Experiments were conducted using a fixed transmission interval of one second. Network conditions were controlled to evaluate three scenarios:

- Baseline (no impairments)
- Network jitter
- Packet loss

For each scenario, network stability and processing time were recorded. Packet captures (PCAP files) and CSV logs were generated to support analysis and reproducibility.

---

# 4. Implementation Details

## 4.1 Software Tools

The project was implemented in Python using socket programming for UDP communication. Wireshark was used to capture network traffic, while logged CSV files were processed to generate performance plots. All scripts and configuration files are included in the project repository.

## 4.2 Execution Workflow

1. The sender transmits telemetry packets at one-second intervals.
2. Network impairments are introduced depending on the experiment scenario.
3. The receiver logs packet arrival times and processing duration.
4. Logged data is analyzed to generate performance plots.

## 5. Results and Analysis

### 5.1 Baseline Performance Analysis

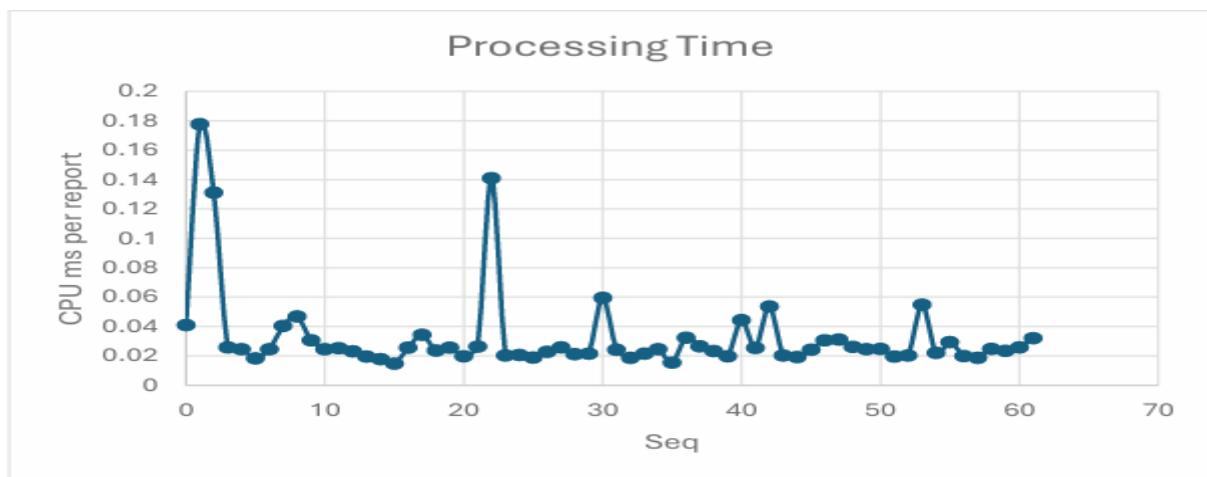


Figure 1: Baseline (1-second intervals) – Processing Time

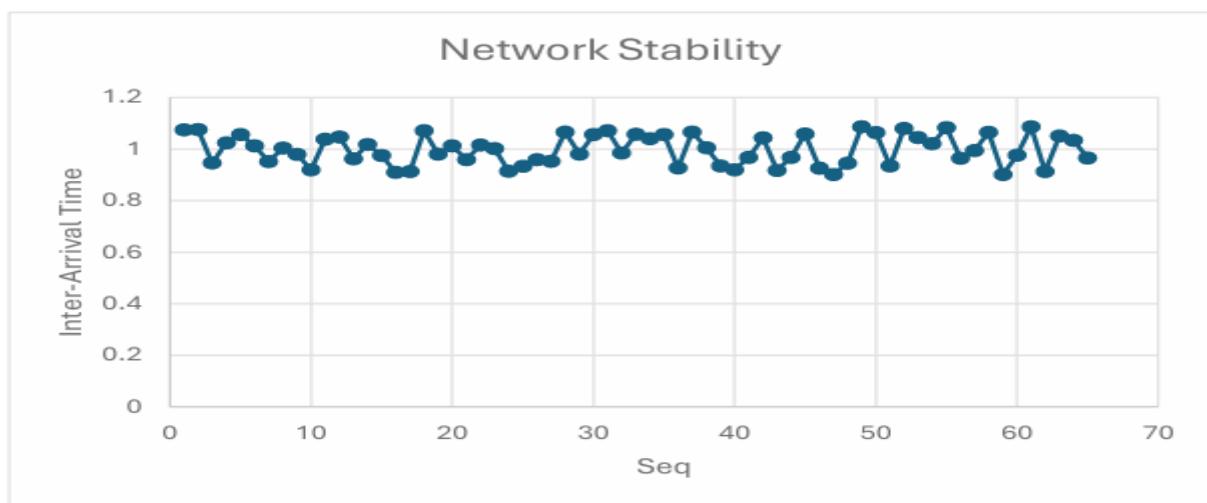


Figure 2: Baseline (1-second intervals) – Network Stability

This experiment proves a reference for system behavior under stable network conditions with no artificial impairments. Figures 1 and 2 illustrate consistent packet arrival patterns and relatively stable processing time. The observed behavior shows predictable system performance and minimal overhead.

under normal operating conditions. This baseline serves as a control scenario against which the impact of network jitter and packet loss can be directly compared.

---

## 5.2 Impact of Network Jitter on Stability and Processing Time

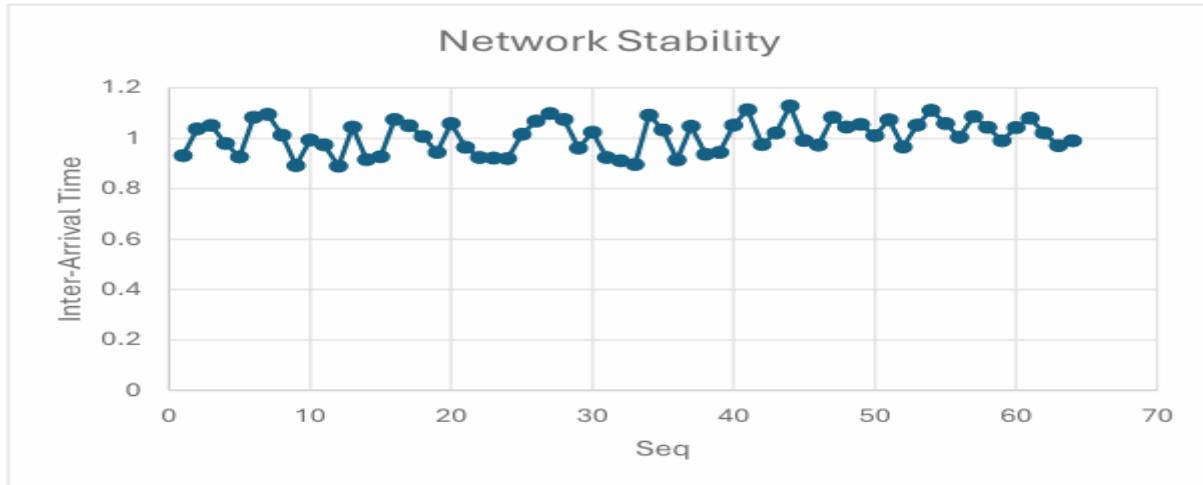


Figure 3: Jitter (1-second intervals) – Network Stability

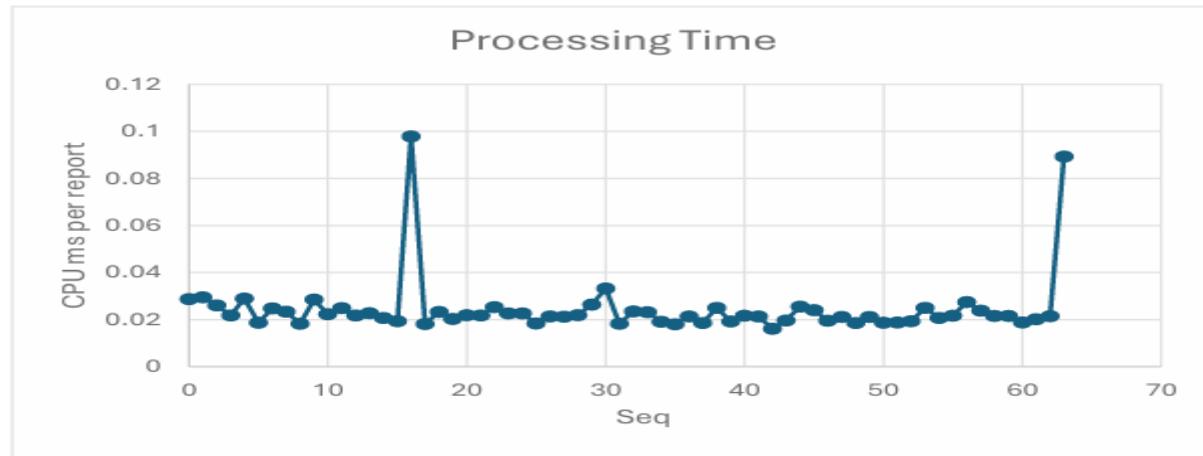


Figure 4: Jitter (1-second intervals) – Processing Time

In this experiment, network jitter was introduced while supporting a fixed one-second transmission interval. Compared to the baseline results, Figures 3 and 4 show increased variability in packet arrival times, leading to reduced network stability despite the absence of packet loss. The irregular arrival pattern causes fluctuations in processing time, as the receiver must manage uneven packet spacing. These results prove that timing variability alone can negatively affect telemetry system performance, even when all packets are successfully delivered.

---

### 5.3 Impact of Packet Loss on Stability and Processing Time

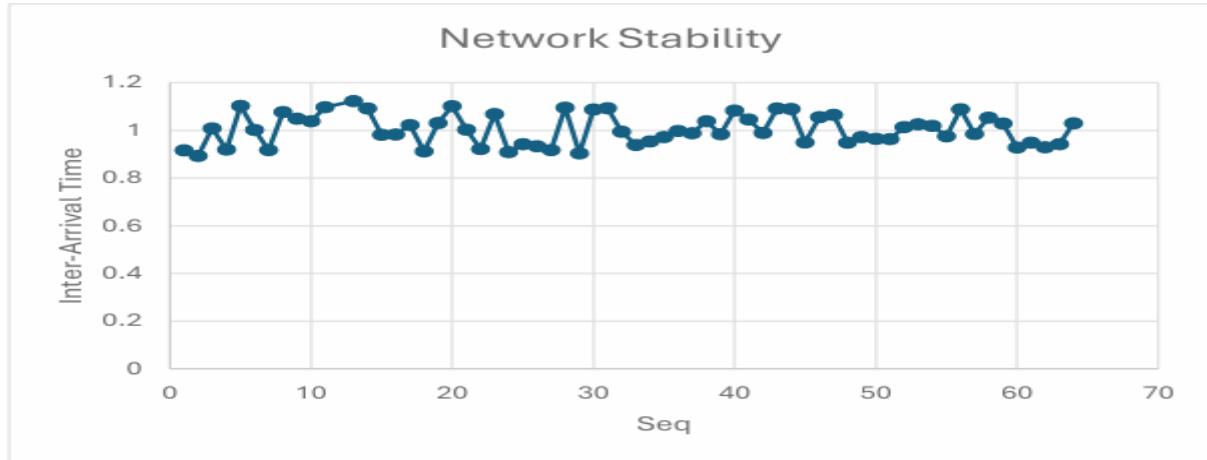


Figure 5: Packet Loss (1-second intervals) – Network Stability

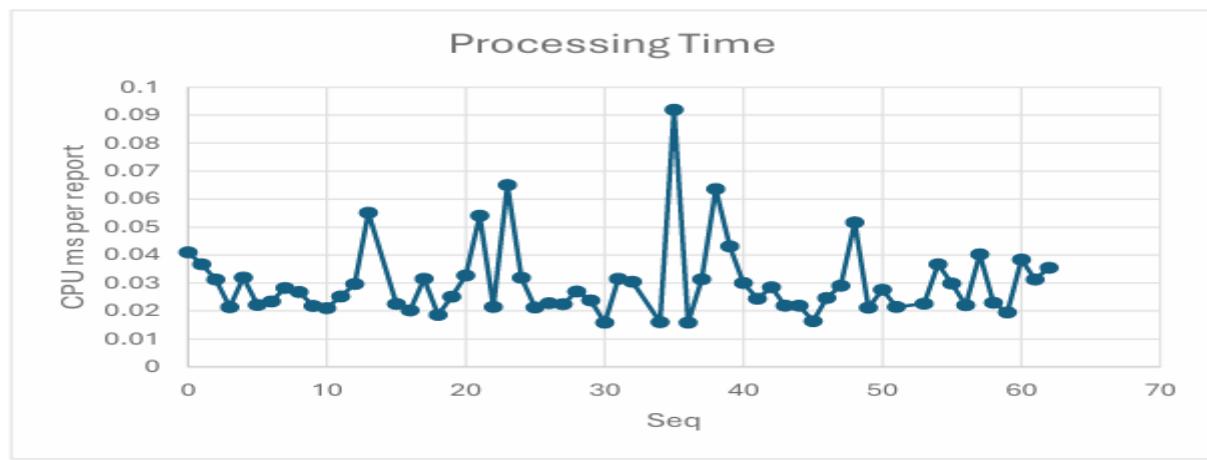


Figure 6: Packet Loss (1-second intervals) – Processing Time

This experiment evaluates system behavior under packet loss conditions. Figures 5 and 6 reveal visible disruptions in network stability caused by missing packets, resulting in gaps in the received telemetry stream. Processing time also shows greater variability compared to the baseline, reflecting the added overhead required to detect and manage missing data. Packet loss has a more severe effect on system reliability than jitter alone, emphasizing the importance of lightweight loss-detection mechanisms in IoT telemetry protocols.

## 6. Comparison with Baseline

Compared to baseline implementation, the proposed telemetry protocol proves improved robustness under adverse network conditions. While baseline performance stays stable only in ideal conditions, the proposed protocol provides better visibility into network impairments through sequencing and timing analysis. The results highlight trade-offs between protocol simplicity and reliability, with the proposed approach offering meaningful advantages for telemetry-focused applications.

## 7. Reproducibility

To reproduce the experiments:

1. Clone the project repository.
2. Install the required Python dependencies.
3. Run the sender and receiver scripts.
4. Introduce network impairments as specified.
5. Capture traffic and analyze generated logs to recreate the plots.

All experiment parameters are documented to ensure reproducibility.

---

## 8. Limitations

The experiments were conducted in a controlled environment and may not fully be real-world wireless networks. Security features such as encryption and authentication were not considered. Additionally, scalability to large numbers of IoT devices was not evaluated.

---

## 9. Conclusion

This project presented the design and evaluation of a custom IoT telemetry protocol refined for periodic data transmission. Experimental results show stable baseline performance and clear degradation under jitter and packet loss, with packet loss having the most significant impact. The findings prove that lightweight protocol design can improve telemetry system observability and robustness, while also highlighting areas for future improvement.

## References

- [1] RFC 768 – User Datagram Protocol (UDP)
- [2] Wireshark Network Protocol Analyzer Documentation