Computer Networks Major Task

Phase_1

Project Proposal: LiteTelemetry Protocol

Submitted by (Team No.13):

Abdelrhman Mohammed Mahmoud	23P0370
Abdelrahman Mohamed Bahig	23P0390
Ahmed Mostafa Gomaa Atia	23p0375
Nagy Ahmed Nagy	23p0365
Abdallah Omar Moustafa Taman	22P0129
Kareem Younis Ahmed Foad Yehya	22P0136

1. Assigned Scenario

The assigned scenario focuses on the design and implementation of a **lightweight, application-layer communication protocol** for **IoT telemetry transmission** over **UDP**.
In this project, a network of IoT sensor devices periodically sends small telemetry readings — such as temperature, humidity, or voltage — to a **central collector**. The system must support real-time, loss-tolerant communication with minimal processing and memory overhead.
The main objectives of this scenario include:

- Building a simple **custom protocol** that defines message structure and flow.
- Supporting **real-time data transfer** without the cost of TCP handshakes.
- Detecting packet **loss, duplication, and delay** efficiently.
- Creating a working **Python prototype** with a client–server setup.

Real-world examples of this scenario include:

- Smart agriculture systems monitoring field conditions.
- Industrial sensors sending periodic status updates.
- Smart home devices reporting telemetry data to a central hub.
- Environmental stations streaming sensor readings to a monitoring platform.

2. Motivation

The **Internet of Things (IoT)** landscape continues to expand rapidly, with millions of devices generating small, periodic telemetry data. These devices often operate in **resource-constrained environments**, where battery life, network efficiency, and simplicity are critical.
Traditional transport protocols like **TCP** introduce unnecessary overhead (handshakes, acknowledgments, retransmissions), which increase **latency and energy usage**.
In contrast, **UDP** provides a fast, connectionless foundation — but lacks mechanisms for **ordering, identification, and reliability tracking**.
To bridge this gap, we propose the **LiteTelemetry Protocol (LTP)** — a **lightweight, binary, and UDP-based protocol** tailored for IoT telemetry. It offers:

- **Low-latency, connectionless communication**
- **Compact binary encoding (~12 bytes header)**
- **Built-in sequence and timestamp fields** for detecting data loss or duplication
- **Minimal CPU and power consumption**

By combining UDP's simplicity with structured application-layer design, **LiteTelemetry** ensures efficient, real-time communication suitable for embedded and low-power devices.

3. Proposed Protocol Approach

3.1 Overview

LiteTelemetry (LTP) is an **application-layer protocol** that defines message structure and communication semantics for IoT telemetry exchange.
It uses **UDP** as its transport layer and emphasizes:

- **Compact, fixed-length message headers**
- **Loss tolerance (≈5%) without retransmission**
- **Real-time responsiveness**
- **Simple parsing for data collectors**

The protocol enables each device to report its data independently without session negotiation or connection setup.

3.2 Protocol Entities

Entity	Description
Sensor Client	Periodically captures telemetry data (e.g., temperature, voltage) and sends it to the server using UDP. Each packet includes a sequence number and timestamp.
Collector Server	Receives and parses packets, detects duplicates and missing data, logs telemetry readings to a CSV file, and provides basic diagnostics.

3.3 Communication Flow

1. **INIT (Initialization):**
   When a client first starts, it sends its first message identifying itself (DeviceID + Version).
2. **DATA Transmission:**
   The client periodically sends telemetry data packets, including a sequence number, timestamp, and measured values.
3. **HEARTBEAT:**
   If no new data is available, the client sends a heartbeat message to notify the server that it remains active.
4. **Server Logging:**
   The server records all messages in a structured CSV file and flags duplicates or missing packets for analysis.

3.4 Message Format

Field Name	**Size (bits)**	**Description**
Version	4	Protocol version (currently 1).
MsgType	4	Message Type → 0 = DATA, 1 = HEARTBEAT.
DeviceID	16	Unique identifier for each sensor device.
SeqNum	16	Incremental sequence number (used for loss/duplication detection).
Timestamp	32	UNIX epoch time of the reading.
Flags	8	Optional bits (for checksum or batching extensions).

This structure ensures a **small, fixed-size header (~12 bytes)** suitable for constrained IoT devices.

3.5 Reliability and Fault Detection

Although UDP does not guarantee packet delivery, **LiteTelemetry** achieves **logical reliability** through its header fields:

- **Sequence Number Tracking:** Detects duplicates and missing packets.
- **Timestamp Validation:** Helps reorder delayed packets.
- **Heartbeat Messages:** Monitor device availability without retransmissions.
- **Per-Device State Cache:** Maintains each device's last sequence and timestamp.

This design provides meaningful diagnostic information with minimal overhead.

3.6 Implementation Overview

The prototype is implemented in **Python** and includes three components:

1. **Server (Collector):** Listens on a UDP port, parses and validates packets, logs messages to telemetry_log.csv, and flags data anomalies.
2. **Client (Sensor):** Periodically sends telemetry readings encoded in the LiteTelemetry message format.
3. **Automated Test Script**: Runs multiple client instances and simulates different conditions such as packet loss and network jitter.