CSE361: Computer Networking

RFC

Submitted by:

Belal Anas 23P0193

Carol Eid 23P0333

Maryam Mohamed 23P0056

Jana Mohamed 23P0050

Toka Elsayed 23P0044

Mariam Tarek 23P0214

Submitted to:

Dr. Karim Emara

TA. Noha Wahdan

Table of Contents

1. Introduction

The **Internet of Things (IoT)** requires lightweight and efficient communication protocols to facilitate data exchange between sensing devices and central collection units. The **Tiny Telemetry Protocol (TTP)** is an application-layer protocol developed specifically for temperature sensing systems operating within local area networks. Built on top of the **User Datagram Protocol (UDP)**, TTP provides a simple and efficient mechanism for transmitting periodic temperature readings from sensor nodes to a central collector for monitoring and analysis.

TTP is designed to operate reliably in environments where devices have limited computational power, memory, and energy resources. It utilizes UDP to minimize communication overhead and latency, enabling real-time data transfer across the network. Although UDP does not provide built-in reliability, TTP incorporates lightweight validation features such as sequence numbering and checksums to ensure data integrity and support accurate performance assessment.

The protocol is developed with the following design objectives:

- **Low power and bandwidth usage** to support constrained temperature sensors.
- **Simple message structure** to ensure ease of implementation and low processing overhead.
- **Efficient operation within local networks** to achieve consistent and low-latency communication.
- **Reliability mechanisms**, including checksums and sequence tracking, to identify packet corruption, duplication, or loss.
- **Scalability**, allowing multiple temperature sensors to transmit concurrently to a single collector node.

2. Protocol Architecture

The **Tiny Telemetry Protocol (TTP)** system is designed to enable reliable, low-latency data transmission between distributed sensor nodes and a central collector over a **UDP-based local network**. It focuses on efficiency, simplicity, and fault detection within LAN environments.

Entities

- **Sensor Node (Client):**
  Acts as a lightweight telemetry device that periodically gathers environmental data such as temperature, humidity, or other measurable parameters. Each sensor node encapsulates its readings into structured UDP packets that include sequence numbers, timestamps, and checksums for validation before sending them to the collector.
- **Collector Node (Server):**
  Functions as the central receiver that continuously listens for incoming UDP packets from multiple sensor nodes. Upon receiving data, it validates the packet structure and checksum integrity, detects packet loss, duplicates, or sequence gaps, and records all valid readings in a structured CSV file for later processing and performance evaluation.

Network Setup

- Communication between the entities is established over **UDP sockets** within the same **Local Area Network (LAN)**.
- The **collector node** operates on a predefined listening port (default: **9999**) to receive telemetry data.
- The **sensor nodes** are configured with the collector's IP address and send readings at fixed intervals.
- The system design emphasizes **low overhead and real-time data delivery**, while checksum validation ensures data integrity despite UDP's connectionless nature.
- This architecture allows easy scaling, where multiple sensor nodes can transmit simultaneously to the same collector with minimal configuration

3. Message Formats

Each TTP message consists of a fixed-size header followed by an optional payload containing sensor readings.

Tiny Telemetry Header Structure:

Field Name	Size (Bytes)	Description
SeqNum	2	Sequence number incremented with each message to detect duplicates or gaps.
DeviceID	1	Unique identifier for each IoT sensor node.
MsgType	1	Specifies the message type (1=INIT, 2=DATA, 3=HEARTBEAT).
Timestamp	4	Unix timestamp (seconds) when the message was generated.
BatchFlag	1	Set to 1 if multiple readings are included; otherwise 0.
Checksum	2	Modulo-65536 sum of all bytes with the checksum field zeroed.
Version	1	Protocol version for compatibility control.

Total Header Size: 12 bytes.

Encoding:

- All header fields are encoded in binary format using network byte order (big-endian).
- The payload is encoded as a UTF-8 string (e.g., “25.7”) or as a binary float value.
- The compact design reduces bandwidth usage and simplifies parsing on constrained IoT devices