Project Proposal — IoTStream v1 (Phase 1)

Course: Computer Networking/ CSE361

Project: IoT Telemetry Protocol

Phase: 1 – Core Protocol Design and Prototype

1. Assigned Scenario

This proposal describes the implementation of Project 1: IoT Telemetry Protocol (Sensor Reporting) using a custom UDP-based protocol named IoTStream v1.
IoTStream v1 enables small, resource-constrained sensors to periodically send telemetry data (temperature, humidity, voltage …) to a central collector server in an efficient, loss-tolerant way.

2. Motivation

Traditional application protocols such as HTTP or MQTT are too heavy for small IoT devices that have:

- very limited memory and CPU resources,
- low or unreliable network bandwidth, and
- strict power constraints.

IoTStream v1 was designed to:

- Operate over UDP to remove handshake and retransmission overhead,
- Use a compact 12-byte binary header,
- Tolerate up to ~5 % random packet loss,
- Support configurable reporting intervals (1 s, 5 s, 30 s), and
- Remain simple enough for constrained devices.

The result is a lightweight telemetry channel that maintains data continuity through timestamps and sequence numbers rather than TCP-style reliability.

Proposed Protocol Approach

Transport Layer

Property	Value
Protocol	UDP
Port	5005
Direction	Sensor (Client) → Collector (Server)
Connection	Connectionless (no session setup)
Retransmission	None – loss tolerant

Entities

- IoT Sensor (Client) – builds and sends telemetry packets.
- Collector (Server) – listens on UDP port 5005, decodes headers, and logs data.

Message Types

Type	Code	Description
INIT	0	Sent once on startup to identify the device.
DATA	1	Sent periodically (1 Hz) carrying five float readings.
HEARTBEAT	2	(Reserved for future phase) used when no new data available.

Binary Header Format (12 bytes)

Field	Size (bytes)	Description
Version	1	Protocol version
MsgType	1	0 = INIT, 1 = DATA, 2 = HEARTBEAT
Device ID	2	Unique sensor identifier
Sequence Number	2	Incremented per packet
Timestamp	4	UNIX time (seconds)
Batching Flag	1	0 = single reading, 1 = batched
Checksum	1	8-bit header checksum placeholder

Total: 12 bytes   Python format: '!BBHHIBB'

Payload Format (DATA only)

Each DATA packet contains five readings (floats):

Field	Type	Size
Reading 1–5	float × 5	20 bytes total

DATA packet size: 12 (header) + 20 (payload) = 32 bytes.

Finite State Flow (Simplified)

Sensor: START

↓

Send INIT → Collector receives & logs

↓

Periodic 1 s timer

↓

Send DATA[n] → Collector parses, checks sequence, logs

↳ repeat

Prototype Implementation

Component	Description
client.py	Sends one INIT + 60 DATA packets (1 Hz).
server.py	Receives UDP packets, unpacks header & payload, prints decoded fields.
script.py	Automates baseline run (60 s) and captures baseline_trace.pcap.

- **Language:** Python 3
- **Libraries:** socket, struct, subprocess, tshark (optional)

---

4. Expected Outcomes

- Successful UDP communication between client and server.
- Correct decoding of header (!BBHHIBB) and float payload.
- Baseline (no-loss) test achieves ≥ 99 % packet delivery.
- Logs and .pcap trace generated as proof of functionality.

---

5. References

1. RFC 768 – User Datagram Protocol (UDP)
2. IoTStream v1 Mini-RFC (Team Design Document)
3. Python Standard Library Docs – socket, struct, subprocess
4. Course Specification – IoT Telemetry Protocol (Phase 1)