Journal of Architectural and Planning Research
39:2 (Winter, 2025) 16

DESIGN AND IMPLEMENTATION OF A BIG DATA PLATFORM IOT DATA ACCESS ARCHITECTURE

Hongfei Xiao
Wenwen Li
Shiqi Tang

To address the practical challenges of industrial equipment data collection on a unified big data platform, this paper proposes an IoT data-access architecture organized into three tiers: equipment and facilities, a network access center, and application services. The network access center provides device identity management, protocol adaptation, and a unified access interface, while downstream services support ingestion of heterogeneous industrial data, including time-series monitoring streams, object data (e.g., files, images, video, and logs), and structured relational records. Based on an industrial big data platform project, we implement the corresponding network architecture and connectivity modules and validate an NB-IoT low-power access scenario using an intelligent gateway and an NB module. Experimental evaluation focuses on the TCP middleware server that handles device connectivity. Under up to 100,000 concurrent connections, the measured server-side response time remains below 100 ms, and throughput scales with thread parallelism until CPU or bandwidth saturation. These results indicate that the proposed architecture and implementation provide a reliable and reusable integration path for heterogeneous industrial devices and data sources.
Keywords: big data platform, IoT connection, Data access architecture, NB-IoT

https://doi.org/10.66033/japr2025-202

INTRODUCTION

The daily operation and maintenance of large equipment fleets aims to reduce failure rates and improve asset utilization. In practice, equipment is geographically dispersed and heterogeneous. Without continuous, trustworthy data and a consistent measurement framework, it is difficult to quantify production losses, identify the key factors that reduce efficiency, and objectively evaluate and motivate maintenance teams.

Recent research on big-data architectures and IoT analytics emphasizes that scalable storage and computation must be coupled with reliable data acquisition and standardized access mechanisms [Wang et al. 2020, Pääkkönen & Pakkala, 2015]. For industrial scenarios, the required capability is not only storage and computation, but also robust data interconnection across operation monitoring, management, maintenance, overhaul, and other workflows, while shielding device heterogeneity and protocol diversity [Cai et al. 2016, Verma et al. 2017]. Cloud-based IoT platforms and smart-city systems provide useful reference models [Ray, 2016, Khan et al. 2015], yet many deployments still face practical challenges in device onboarding, protocol adaptation, and high-concurrency connectivity. Surveys on heterogeneous IoT and streaming analytics further highlight the importance of unified access and information modeling for sustainable growth of IoT ecosystems [Qui et al. 2018, Mohammadi et al. 2018].

This paper addresses these challenges by designing and implementing an IoT data-access architecture for a big data platform used in industrial data collection. The main contributions are: (i) a three-tier connectivity structure that separates equipment and facility endpoints, a network access center, and application services; (ii) an access management platform that provides device identity, protocol adaptation, and unified data interfaces to downstream applications; (iii) an implementation of an NB-IoT low-power access scenario using an intelligent gateway and an NB module; and (iv) an experimental evaluation of the TCP middleware server that supports high-concurrency device connections, including response-time, throughput, and data-processing ratio measurements. The proposed architecture provides a practical foundation for consolidating heterogeneous equipment data into a big-data environment and supporting subsequent analytics and intelligent maintenance [Mahdavinejad et al. 2018, Iqbal, Zerguine & Khan 2021].

The paper is organized as follows: §2 reviews related work and the underlying concepts. §3 describes the proposed architecture and implementation. §4 reports the experimental validation and analyzes the results. §5 concludes the paper and discusses limitations and future work.

RELATED WORK

IoT deployments couple sensing, communication, and application services to enable data-driven decision making across distributed assets. At the network and access layers, cross-layer communication design remains a core topic; for example, wireless sensing systems operating in constrained spectrum environments motivate careful coordination across PHY/MAC/routing to achieve reliable and efficient uplink delivery [Iqbal et al. 2020]. Alongside communication advances, a range of integration mechanisms has been explored, including blockchain-based trust and traceability proposals. The subsequent retraction of some published claims underscores that architectural choices must be supported by reproducible experiments and clearly defined assumptions [Tanwar et al. 2022].

To manage the scale and locality of IoT-generated data, edge or regional computing has been proposed to reduce data movement and mitigate the impact of large-scale industrial data streams [Badshah et al. 2022]. Domain applications further demonstrate that IoT data pipelines increasingly incorporate machine learning, including deep-learning-enabled monitoring and control in energy systems [Iwendi & Wang, 2022]. These trends imply that a practical IoT platform must not only connect devices but also provide stable, standardized

data interfaces to support analytics workloads.

Figure 1 illustrates the conventional three-layer view of IoT (perception, network, and application). Modern industrial IoT also includes multimodal data such as images and video, for which data quality and preprocessing can become a bottleneck; representative enhancement methods for challenging sensing environments highlight the need to treat object data as a first-class data type in IoT platforms [Zhou et al. 2023]. From the viewpoint of large-scale sensor networking, clustering and load balancing are important to extend network lifetime and stabilize throughput, especially in energy-harvesting and multi-domain scenarios [Lou, Zhang & Bai 2023, Ali et al. 2023].

Despite the above progress, existing work often treats device connectivity, protocol adaptation, and heterogeneous data ingestion as separate problems. In practice, industrial deployments require an integrated access architecture that unifies device onboarding, protocol translation, and high-concurrency connectivity to a big data platform. The remainder of this paper targets this gap by presenting an implementable data-access architecture and validating the performance of its connectivity middleware.

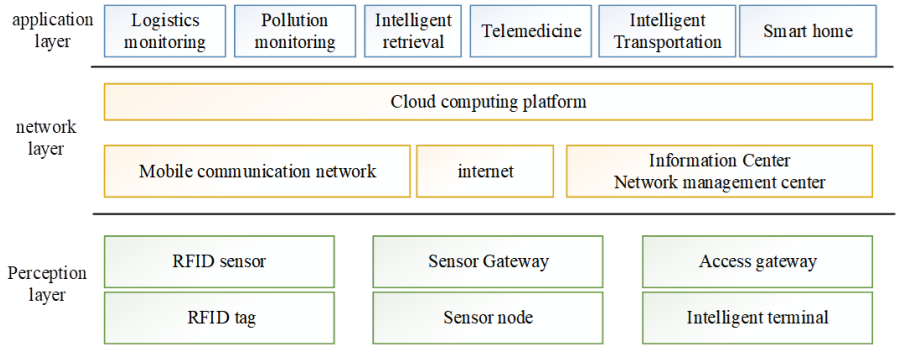

Figure 1: Structure of IoT

While the vision of "interconnecting everything" is widely recognized, industrial rollouts are often slowed by (i) the complexity of edge integration and multi-protocol compatibility, (ii) fragmentation of platform interfaces and device management mechanisms, and (iii) the need for coordinated hardware/software engineering across sensors, gateways, and cloud services. In many ecosystems, platform providers define incompatible device models and access APIs, which hinders resource sharing and increases the cost of integration and operation.

Accordingly, a reusable access platform that offers standardized device onboarding, unified protocol adaptation, and consistent data interfaces can substantially shorten R&D cycles for vertical applications. Rather than replacing existing cloud platforms, the goal is to provide a network access center that integrates heterogeneous equipment into a common big-data ingestion pipeline and exposes secure, well-defined interfaces to upstream application services.

METHODS

Construction structure

This section presents the implementation framework of the proposed IoT data-access architecture, drawing on IoT system theory and the requirements of industrial data collection projects. The architecture follows a layered design that separates terminal sensing and actuation, network access and protocol adaptation, and application-oriented services. The key elements are summarized in Figure 2 and include terminal devices

(sensors and controllers), access gateways, the network access center, and application services that consume unified data interfaces.

In this study, the device management function in the application layer is implemented as an IoT connection management platform, whose overall architecture is shown in Figure 3. Large IoT ecosystems provided by cloud vendors and telecom operators offer valuable capabilities, yet industrial deployments still require an engineering layer that can integrate heterogeneous on-site equipment, unify protocols, and provide consistent device information models across projects and vendors. Therefore, the proposed design emphasizes modular components (protocol adaptation, device modeling, access control, and ingestion services) that can be deployed independently and evolved without changing device-side hardware.

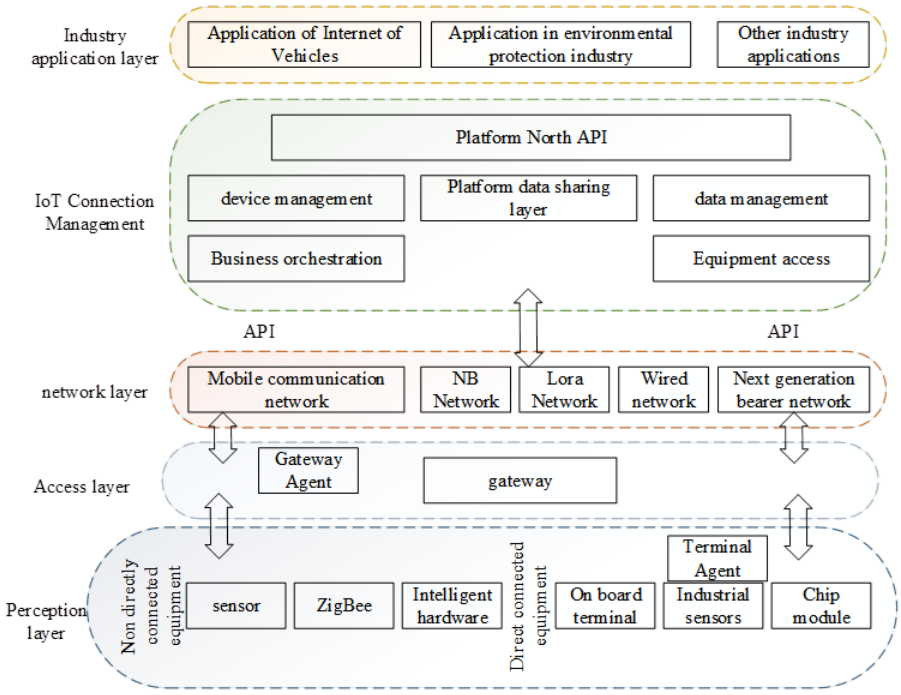

Figure 2: Architecture Design of IoT Based on Device Access

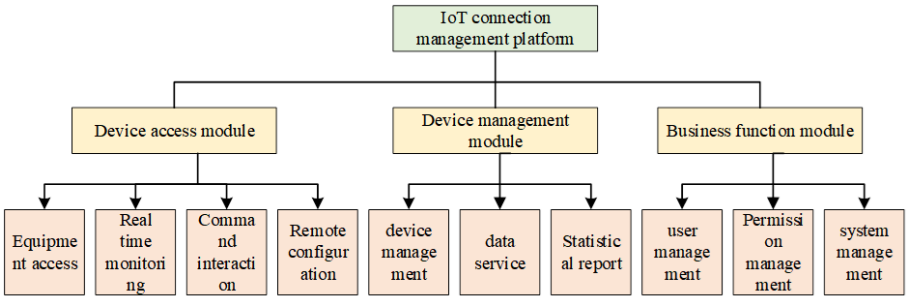

Figure 3: Overall architecture of application layer IoT connection management platform construction

IoT equipment access management platform

As shown in Figure 4, the IoT equipment access management platform is designed to provide unified access and centralized management for heterogeneous terminal devices. From an engineering perspective, the

platform must address two coupled problems: (i) enabling reliable network access and protocol adaptation for diverse terminal devices, and (ii) providing device management, visualization, and standardized data interfaces for upstream applications.

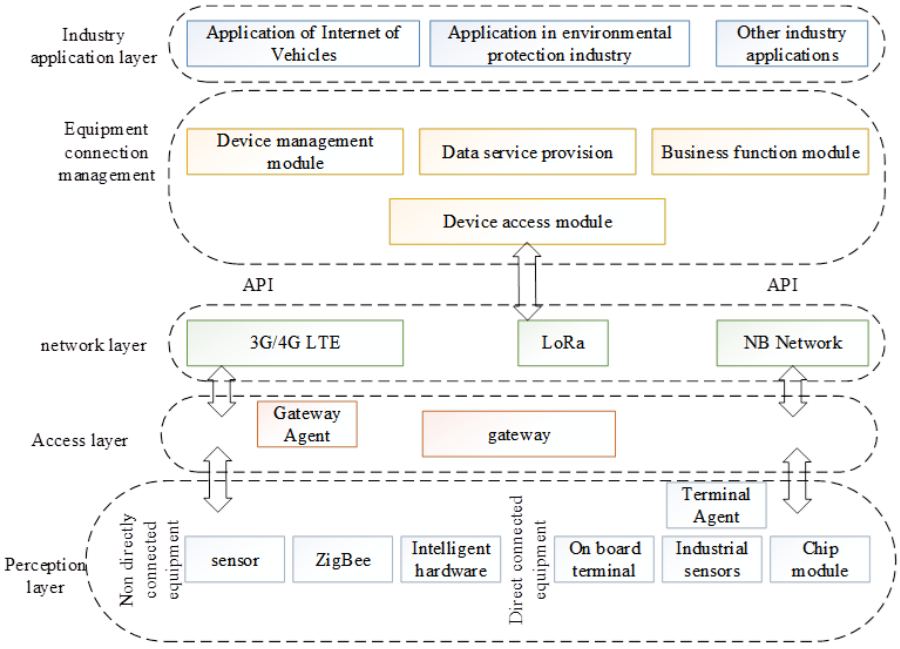

Figure 4: Overall architecture of IoT equipment access management platform

The platform is organized into four logical layers:

(1) Perception layer

Directly connected devices (often IP-capable terminals) are equipment endpoints with sensing/actuation capabilities that can generate data and interact with the external environment. Typical examples include vehicle terminals, industrial controllers, and intelligent meters. Indirectly connected devices are resource-constrained endpoints (e.g., simple sensors, ZigBee devices, and low-power hardware) that require a gateway for data reporting and command delivery.

(2) Access layer

The access layer provides network connectivity for perception-layer devices and unifies heterogeneous protocols and payload formats through an agent gateway. Resource information can be collected and reported through a gateway agent (for indirectly connected devices) or a terminal agent (for IP-capable devices). To support large-scale deployments and avoid hot spots, the access layer assigns devices to logical partitions using deterministic rules (e.g., hashing on device identifiers), enabling horizontal expansion and stable load distribution.

(3) Network layer

The network layer supports remote information exchange between terminal equipment and the IoT connection management platform. Transmission modes are selected according to application scenarios and deployment constraints. In this work, wireless access is emphasized, including 3G/4G/LTE and low-power wide-area communication (NB-IoT) for resource-constrained devices.

(4) Industry application layer

After collection and preprocessing, the platform exposes data to industry applications for analysis and service development. By combining big data pipelines with AI-related technologies, the application layer can support intelligent monitoring, diagnosis, and decision-making for industrial operation and maintenance.

NB IoT equipment access scenario

This section designs and implements terminal access in an NB-IoT low-power scenario. NB-IoT is suitable for low-rate, small-payload telemetry and control because of its wide coverage, low power consumption, and low device cost. In practice, NB-IoT can be used to report monitoring data periodically and to deliver control commands with bounded latency, while higher-bandwidth links (e.g., 4G/LTE) can be used for large object data when required. Figure 5 summarizes the relationship between terminals, gateways, the NB-IoT network, and the platform.

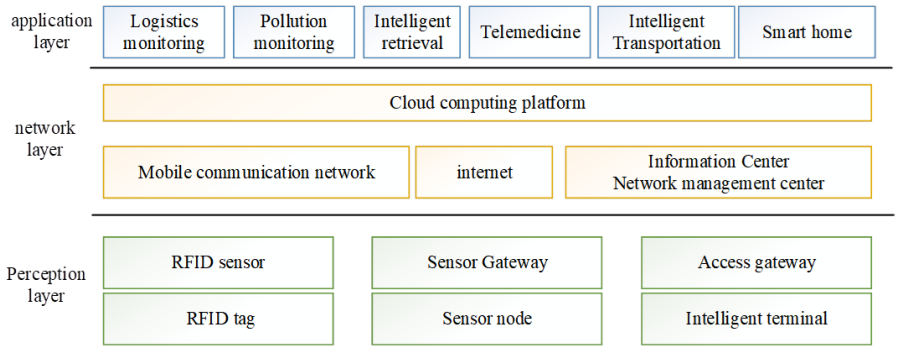

Figure 5: NB IoT Vertical Industry System Architecture

In this paper, an intelligent gateway based on a Zynq-7000 platform, a BC95 NB communication module, and external IoT sensor equipment are used to complete terminal access in the low-power NB-IoT scenario. The communication module integrates the USIM card and RF antenna provided by the telecom operator and exposes serial interfaces for terminal control and data transfer.

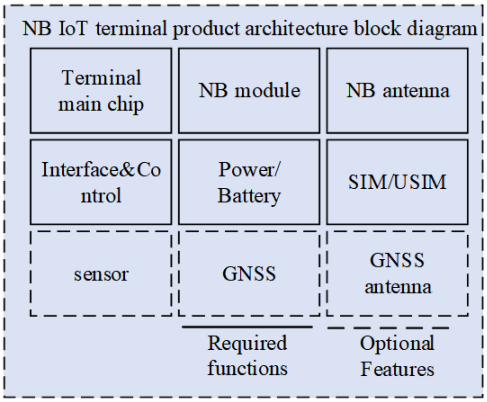

Figure 6: Product Architecture of NB IoT Module

Figure 6 shows the product architecture of the NB-IoT module. Depending on the intended use, optional functional blocks (indicated by dotted lines) can be configured. In general, NB modules, SIM/USIM, and NB antennas must be integrated into existing hardware to enable NB-IoT networking. The power supply design must satisfy ripple noise, dynamic response, and loop stability requirements; the clock source must

meet frequency offset and phase noise requirements; and the RF front-end must satisfy transmission power, EVM, receiving sensitivity, isolation, and noise-figure requirements to ensure stable network access.

After integrated commissioning of the equipment and gateway, the gateway is connected to the NB communication module. The BC95 module is controlled through standard AT commands to complete device registration and activation, establish connectivity with the platform, and submit sensor data through the gateway. The platform can then manage the device lifecycle (registration, configuration, monitoring, and command delivery) and visualize device and data states. In our implementation, the platform provides an open API that allows collected data to be consumed by external device-management and analytics systems. Figure 7 illustrates the end-to-end access and integration process for the NB-IoT scenario.

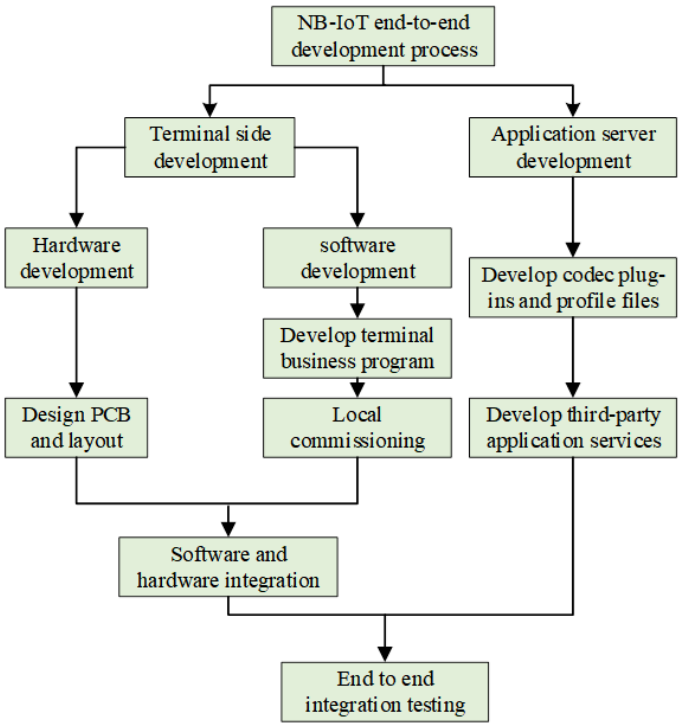

Figure 7: NB IoT Development Access Process

At present, the terminal controls the communication module through AT commands. Therefore, in addition to implementing application-specific business logic, the terminal needs a lightweight scheduler to manage serial communication, module state transitions, and message retransmission. A typical procedure is as follows: the module connects to the terminal, the serial port is configured, and the network access parameters (e.g., platform address) are provisioned. Signal detection is performed after enabling the module. In our tests, a signal strength above the recommended threshold indicates that the device can communicate reliably. The AT command AT+CGATT=1 is used to attach to the network; a returned OK indicates successful activation. Before bidirectional data exchange, the uplink and downlink notification functions are configured to avoid missing data reception or transmission.

The device can send telemetry to the platform through the NB module and can also receive downlink commands. The platform receives, stores, and exposes the data in near real time through open APIs, which can be integrated into downstream device-management platforms as required.

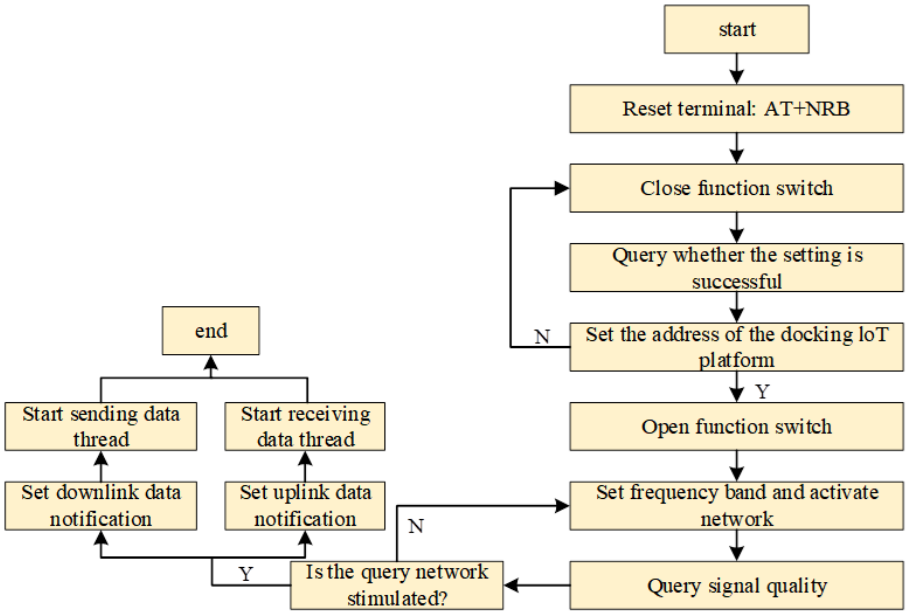

Figure 8: Flow Chart of NB Module Communication Module

Heterogeneous data access

The data ingested by the big data platform can be divided into three categories: equipment real-time data (time-series telemetry such as operating measurements), object data (e.g., fault recorder files, vibration data files, images, video, and log archives), and relational data (e.g., production management records). Different front-end services are required for different data types, such as time-series ingestion services, object storage front-ends, structured data ETL services, and API-based integration services.

Time-series telemetry typically exhibits strong temporal continuity and large throughput, with burstiness and latency sensitivity during peak periods. Therefore, single-node ingestion services must provide high write throughput while ensuring idempotent processing to avoid loss or duplication. To improve reliability, the design supports continuous ingestion under single-node failure through buffering and retry mechanisms at the gateway and at the access service. As shown in Figure 9, the platform provides three access modes for time-series data: real-time access, batch access, and timed batch access. Real-time access targets continuously generated monitoring data that must be persisted for later analysis and diagnosis.

Object data access includes object registration, integrity verification, and object upload. Registration creates object metadata (e.g., file name, type, time range, and purpose) to support indexing and retrieval. Integrity verification uses MD5 checks to ensure that the received content matches the transmitted content. Upload operations rely on object storage and distributed file system interfaces. The IoT connection ETL tool provides three upload services for different scenarios: upload through the management console (simple interactive uploads), upload through REST APIs (web-friendly medium-size uploads), and upload through Java SDK programs (large-scale and automated uploads).

Communication mechanism

Figure 10 shows the communication mechanism between the IoT platform and equipment. The platform provides functions for device registration, configuration management, log monitoring, and command issuance.

Figure 9: Time sequence data access

Users operate the device-management functions through the management interface. The platform middleware starts a Socket server that listens for device connections and processes device uplink messages.

After receiving a request, the server forwards it to the device access module, which validates message format, checks device identity and permissions, performs protocol translation if needed, and dispatches the message to the corresponding ingestion service (time-series, object, or structured). For downlink control, the platform generates commands, routes them to the correct device connection based on the device identifier and partition mapping, and sends acknowledgments to ensure reliable delivery.

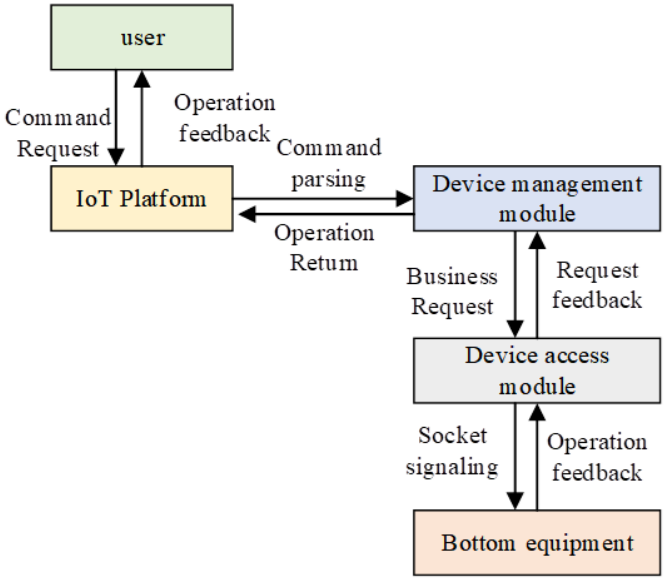

Figure 10: Communication Mechanism between IoT Equipment and Platform

EXPERIMENTS

The client transmits a large number of requests to the server in a client–server architecture. For an IoT access platform, concurrent access capacity and response time are critical to availability and stability. In this section, we evaluate the implemented TCP middleware server and the TCP data receiving module in terms of response time, throughput, and effective processing ratio under increasing concurrency.

RT denotes response time, defined here as the server-side elapsed time between receiving a TCP request and completing the corresponding response write operation. A response time within 100 ms is commonly

considered acceptable for interactive device management and near-real-time monitoring. A concurrency scenario denotes the number of simultaneous connections or requests that the system is required to sustain. Concurrency testing is used to evaluate carrying capacity and delay behavior under load.

In the test setup, a client program runs in a JDK 1.8 environment and uses Netty (4.x) to establish TCP client connections. Java's ScheduledExecutorService is used to generate connections and requests according to the target concurrency. Unless otherwise stated, each scenario is executed for 30 s to reduce the impact of transient fluctuations, and results are reported as the average of repeated runs under the same configuration. After the middleware server is started, it receives TCP requests from the client and records connection events. For confidentiality, client addresses in the logs are masked, as shown in Table 1.

Table 1: Connection Information of IoT Access Platform Server

boot. server. TcpServer handler	Connected client address: XXX
boot. server. TcpServer handler	Connected client address: XXX
boot. server. TcpServer handler	Connected client address: XXX
boot. server. TcpServer handler	Connected client address: XXX

System evaluation

The performance of the server is verified through concurrency tests in a middleware server simulation environment. Table 2 reports concurrent response time under different concurrency levels, where C1, C2, and C3 denote different thread-pool configurations (increasing the configured number of worker threads and cores). The results show that response time remains below 100 ms for up to 100,000 concurrent connections across all configurations. When concurrency increases to 1,000,000, response time rises to approximately 300 ms, indicating that system resources (CPU scheduling, network stack, and memory) approach saturation. In production, this behavior motivates horizontal scaling, in which multiple access nodes share the load via deterministic device partitioning and external load balancing (e.g., Nginx or L4 load balancers) so that each node operates within its stable concurrency range.

Table 2: Concurrency Scenario Response Time

[UNK][UNK];	100	1000	2000	5000	10000	100000	1000000
C1(ms/1)	9	8	11	17	16	49	325
C2(ms/10)	10	13	13	18	27	58	304
C3(ms/20)	14	17	22	29	33	68	312

Table 3 reports throughput in transactions per second (TPS) under different thread counts, with a fixed message size of 2048 bytes. Throughput increases substantially as the number of worker threads increases, rising from 361 TPS with a single thread to over 3400 TPS at 100 threads, and reaching 4202 TPS at 500 threads. The gain diminishes at high thread counts because CPU context switching and network I/O become dominant bottlenecks. Overall, the results indicate that the middleware server can meet the concurrent access requirements of lightweight IoT devices under typical industrial workloads.

Table 3: TPS Test Results under Load Data

Serial No	Message (bytes)	Number of threads	TPS
1	2048	1	361
2	2048	9	1601
3	2048	33	2101
4	2048	100	3413
5	2048	500	4202

There are 500 million pieces of test data collected by terminals of three types of equipment. The access performance results are in Figure 11 and Figure 12. The horizontal axis in the figure indicates the number of threads, the vertical axis in Figure 13 indicates the throughput (times/second), and the vertical axis in Figure 14 indicates the delay time (milliseconds).

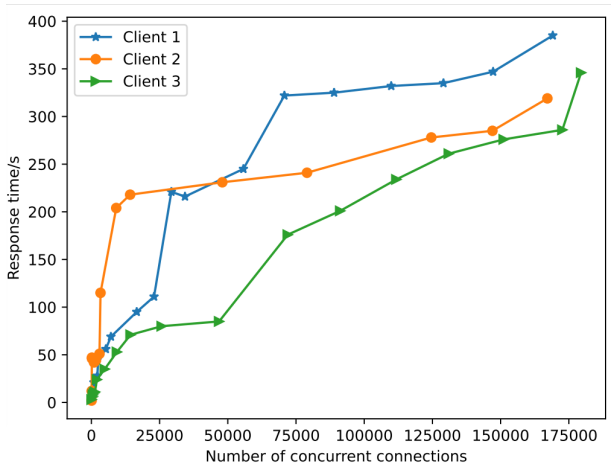

Figure 11: Concurrent Access Response Test Results

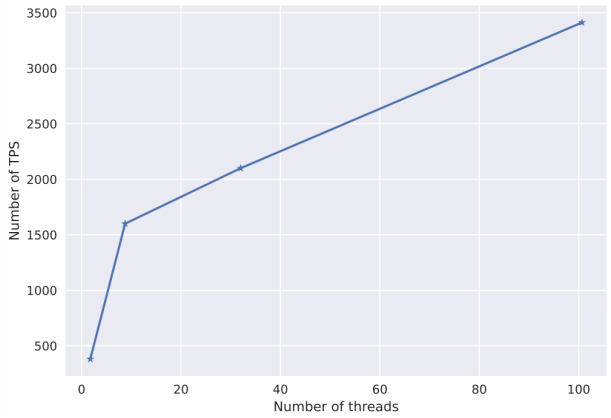

Figure 12: TPS Testing

Figure 13 shows that throughput increases with the number of threads and then approaches saturation. When the thread count rises to around 1000, throughput changes only slightly, indicating that bandwidth and CPU resources become the limiting factors. This saturation behavior is expected in high-concurrency network servers and suggests that further throughput improvements require either horizontal scaling (more server instances) or optimization of the I/O path (e.g., batching, zero-copy, and reduced per-message overhead).

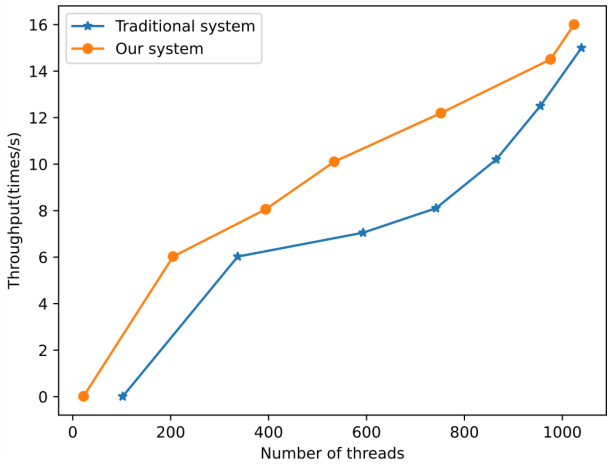

Figure 13: Performance Throughput Test Curve

Figure 14 shows the corresponding delay trend. As the number of threads increases, CPU scheduling becomes more frequent, and the CPU time allocated to each thread decreases, leading to increased access delay. In our implementation, two mechanisms mitigate delay growth under high concurrency: (i) an event-driven, non-blocking I/O model provided by Netty that reduces thread-per-connection overhead, and (ii) deterministic device partitioning and request routing (based on device identifiers) that reduces hot spots and improves cache locality. As a result, delay increases gradually and remains within an acceptable range for typical industrial monitoring and control tasks.

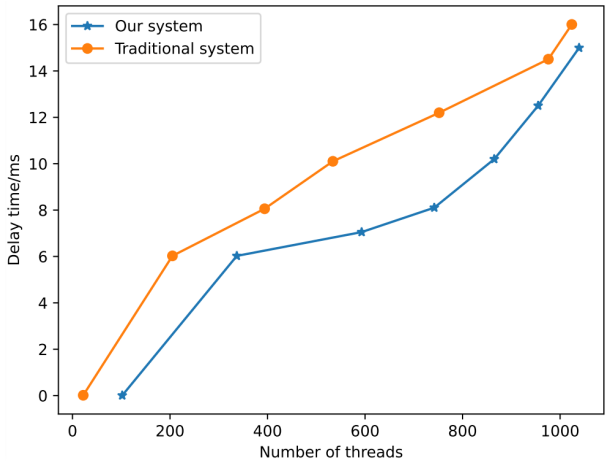

Figure 14: Performance Delay Test Curve

The TCP data receiving module is a core component of the platform and directly determines network communication performance because it is responsible for receiving and preprocessing sensor data. This section explores the performance and configuration sensitivity of the TCP server through controlled experiments. The concurrency parameter is denoted by n; we increase n by one order of magnitude when n is small and by smaller increments when n is large. Specifically, we select $n \in \{1, 10, 100, 200, 500, 600, 700, 800, 900, 1000\}$. To minimize connection establishment overhead and reduce the impact of transient server fluctuations, each test runs for 30 s. The results are reported in Table 4.

Table 4 reports the total bytes sent by clients, the total bytes returned by the server, and the effective data processing ratio P (returned bytes divided by sent bytes) measured over the fixed test window. As concurrency

increases beyond 500, P decreases gradually, and at $n = 1000$ approximately half of the transmitted bytes are processed and returned within the test window. This decrease reflects queueing and backpressure effects under heavy load rather than functional failure. In typical industrial deployments, sensor sampling periods are often on the order of seconds rather than 100 ms; increasing the sampling period by one order of magnitude reduces the arrival rate accordingly and allows higher concurrency at the same processing ratio. Moreover, in many monitoring scenarios, a bounded delay (e.g., around 100 ms) is acceptable for aggregated telemetry, which further enlarges the feasible concurrency range. Therefore, while TCP server performance declines as concurrency increases, the observed behavior is sufficient for most practical production environments when combined with appropriate sampling periods and horizontal scaling.

Table 4: Server Test Results

Test No	Concurrency n	Send data (Byte)	Server returned data (Byte)	Data processing ratio P
1	1	799	799	100.00%
2	10	8446	8446	100.00%
3	100	81298	81279	99.98%
4	200	162058	159925	98.69%
5	500	264517	246391	93.15%
6	600	251554	226369	89.99%
7	700	275305	202109	73.42%
8	800	277339	196624	70.89%
9	900	424435	219448	51.71%
10	1000	2118259	1002175	47.32%

CONCLUSION

(1) The proposed IoT connection architecture provides a unified path for connecting heterogeneous industrial equipment to a big data platform through a network access center, enabling consistent device onboarding, protocol adaptation, and access management. (2) By separating ingestion services for time-series telemetry, object data, and structured relational records and exposing open interfaces to application services, the architecture reduces coupling between device-side protocols and analytics applications. (3) Experimental results on the implemented TCP middleware demonstrate stable response time under up to 100,000 concurrent connections and scalable throughput until system resources (CPU or bandwidth) saturate, which supports practical industrial deployments. Future work will focus on distributed deployment in production and on systematic security evaluation of authentication, authorization, and encrypted transport mechanisms.

ACKNOWLEDGMENTS

This study was financially supported by the Information security and big data application technology innovation platform(Project No.: YJP-2019-01);Internet plus Urban and Rural Creative Development Center(Project No.: YJP-2021-03).

CONSENT FOR PUBLICATION

All authors reviewed the results, approved the final version of the manuscript and agreed to publish it.

DATA AVAILABILITY

The experimental data used to support the findings of this study are available from the corresponding author upon request.

CONFLICTS OF INTEREST

The authors declared that they have no conflicts of interest regarding this work.

REFERENCES

Wang J, Yang Y, Wang T, Sherratt RS, Zhang J. Big data service architecture: a survey. Journal of Internet Technology. 2020 Mar 1;21(2):393-405.

Pääkkönen P, Pakkala D. Reference architecture and classification of technologies, products and services for big data systems. Big Data Research. 2015 Dec 1;2(4):166-86.

Cai H, Xu B, Jiang L, Vasilakos AV. IoT-based big data storage systems in cloud computing: perspectives and challenges. IEEE Internet of Things Journal. 2016 Oct 19;4(1):75-87.

Verma S, Kawamoto Y, Fadlullah ZM, Nishiyama H, Kato N. A survey on network methodologies for real-time analytics of massive IoT data and open research issues. IEEE Communications Surveys & Tutorials. 2017 Apr 14;19(3):1457-77.

Ray PP. A survey of IoT cloud platforms. Future Computing and Informatics Journal. 2016 Dec 1;1(1-2):35-46.

Khan Z, Anjum A, Soomro K, Tahir MA. Towards cloud based big data analytics for smart future cities. Journal of Cloud Computing. 2015 Feb 18;4(1):2.

Qiu T, Chen N, Li K, Atiquzzaman M, Zhao W. How can heterogeneous internet of things build our future: A survey. IEEE Communications Surveys & Tutorials. 2018 Feb 8;20(3):2011-27.

Mohammadi M, Al-Fuqaha A, Sorour S, Guizani M. Deep learning for IoT big data and streaming analytics: A survey. IEEE Communications Surveys & Tutorials. 2018 Jun 6;20(4):2923-60.

Mahdavinejad MS, Rezvan M, Barekatain M, Adibi P, Barnaghi P, Sheth AP. Machine learning for Internet of Things data analysis: A survey. Digital Communications and Networks. 2018 Aug 1;4(3):161-75.

Iqbal N, Zerguine A, Khan S. OFDMA-TDMA-based seismic data transmission over TV white space. IEEE Communications Letters. 2021 Jan 18;25(5):1720-4.

Iqbal N, Al-Dharrab SI, Muqaibel AH, Mesbah W, Stüber GL. Cross-layer design and analysis of wireless geophone networks utilizing TV white space. IEEE Access. 2020 Jun 26;8:118542-58.

Tanwar S, Gupta N, Iwendi C, Kumar K, Alenezi M. [Retracted] Next Generation IoT and Blockchain Integration. Journal of Sensors. 2022;2022(1):9077348.

Badshah A, Iwendi C, Jalal A, Hasan SS, Said G, Band SS, Chang A. Use of regional computing to minimize the social big data effects. Computers & Industrial Engineering. 2022 Sep 1;171:108433.

Iwendi C, Wang GG. Combined power generation and electricity storage device using deep learning and internet of things technologies. Energy Reports. 2022 Nov 1;8:5016-25.

Zhou J, Sun J, Zhang W, Lin Z. Multi-view underwater image enhancement method via embedded fusion mechanism.

Engineering applications of artificial intelligence. 2023 May 1;121:105946.

Luo X, Zhang C, Bai L. A fixed clustering protocol based on random relay strategy for EHWSN. Digital Communications and Networks. 2023 Feb 1;9(1):90-100.

Ali J, Jhaveri RH, Alswailim M, Roh BH. ESCALB: An effective slave controller allocation-based load balancing scheme for multi-domain SDN-enabled-IoT networks. Journal of King Saud University-Computer and Information Sciences. 2023 Jun 1;35(6):101566.

Additional information may be obtained by writing directly to Hongfei Xiao, School of Information Engineering, Chuzhou Polytechnic, Chuzhou, Anhui 239000, China; Email: hxo02ao@163.com

AUTOBIOGRAPHICAL SKETCHES

Hongfei Xiao is at School of Information Engineering, Chuzhou Polytechnic, Chuzhou, Anhui 239000, China.

Wenwen Li is at School of Media and Design, Chuzhou Polytechnic, Chuzhou, Anhui 239000, China.

Shiqi Tang is at Information Center of Ministry of Science and Technology, Beijing, 100862, China.

Manuscript revisions completed 30 November 2025.