Design of Domestic Chip Scheduling Architecture for Smart Grid Based on Edge Collaboration

Feng Chen *, Hongjing Liang , Shangxi Li , Lan Yue , Pingguo Xu

Posted Date: 28 May 2025

doi: 10.20944/preprints202505.2141.v1

Keywords: smart grid; edge computing; domestic chip; cooperative scheduling

 doi:10.20944/preprints202505.2141.v1

Article

Design of Domestic Chip Scheduling Architecture for Smart Grid Based on Edge Collaboration

Feng Chen [1,*] Hongjing Liang [1], Shangxi Li [2], Lan Yue [3] and Pingguo Xu [3]

[1] Shenzhen Haofengquan Electronics Co., LTD, Shenzhen, China
[2] Shaofengquan Electronics Co., LTD, Shenzhen, China
[3] Shenzhen Fucore Electronic Technology Co., LTD, Shenzhen, China
* Correspondence: cf1981518@163.com

Abstract: In order to improve the dispatch autonomy and localization control level of smart grid in complex operation environment, a domestic chip dispatch architecture based on edge collaboration is constructed, taking RISC-V architecture SoC and FPGA collaboration unit as the core, integrating the lightweight protocol stack, distributed collaboration mechanism, and edge-cloud scheduling strategy, and researching the optimization of multi-node scheduling and link adaptive mechanism. The system design covers key modules such as hardware architecture, communication protocols, state synchronization and scheduling security control, and has been validated in grid multi-source sensing and load prediction scenarios. It is analyzed that the architecture maintains the task response delay below 480ms and the scheduling stability fluctuation within ±60ms under high load conditions, and has good elasticity and robustness.

Keywords: smart grid; edge computing; domestic chip; cooperative scheduling

1. Introduction

As power systems advance toward higher perception, control, and autonomy, traditional centralized scheduling can no longer meet the demands of distributed power access, frequent grid fluctuations, and dynamic edge load adjustments. In response, building an edge-cooperative scheduling system with domestic autonomous control has become essential for smart grid evolution. Domestic chips, with advantages in delay control, heterogeneous computing, and protocol compatibility, offer a solid foundation for enhancing real-time response and resource elasticity. Studying edge-coordinated domestic chip architectures holds significant engineering and technical value for achieving reliable and autonomous power system operation.

2. Design of Edge Computing Nodes Based on Domestic Chips

2.1. Edge Computing Node Hardware Architecture Design

To meet the dual demands of high performance and low power consumption in distributed load scheduling, the edge computing node adopts a domestic RISC-V architecture chip as the core, integrated with an FPGA co-processor and reliable communication module. The overall design follows modular and low-latency principles [1]. The master chip handles data scheduling and lightweight computing, while the FPGA focuses on signal preprocessing and protocol conversion, supporting adaptive clock domain management to enhance parallelism and bus efficiency. Communication interfaces include dual-mode Ethernet, 5G, and LoRa, ensuring access stability across diverse network topologies. Figure 1 illustrates the interconnection among the processing core, data buffer, encryption module, sensing interface, and energy management unit, highlighting the integrated response and hardware scheduling capabilities at the chip level.

 doi:10.20944/preprints202505.2141.v1

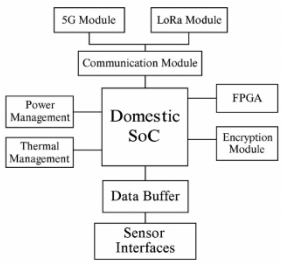

Figure 1. Block diagram of hardware structure of edge node domestic chip.

2.2. Edge Node Software System Design

Based on the domestic SoC platform, the edge node software adopts a lightweight modular architecture, compatible with domestic OS kernels such as OpenEuler or embedded RT-Thread, and integrates task scheduling, communication stacks, security modules, and sensing service management [2]. Following a microkernel model, the system uses middleware to decouple hardware and upper-layer services, enabling resource sharing and unified scheduling control. The scheduling framework combines event-triggered and cyclic strategies to support real-time responsiveness and low-power operation. Communication interfaces support MQTT, CoAP, and Modbus, accommodating smart grid multi-source data. Table 1 outlines the structure, core functions, and resource scheduling of each software subsystem, demonstrating efficient collaboration and security under chip resource constraints.

Table 1. Functional module division of edge node software.

module (in software)	core functionality	Chip Resource Call
Task Scheduler	Real-time task scheduling, priority management	CPU core scheduler, interrupt controller
Protocol Stack	Support MQTT/CoAP/Modbus protocol parsing and sending/receiving	Network Interface Driver, Cache Control Module
Security Service	Local authentication, access control, security log management	Encryption Module, Storage Controller
Sensor Manager	Sensory task management, data sampling rate adjustment	GPIO/I2C/SPI interface, interrupt line management
System Monitor	Node status monitoring, fault detection, energy consumption estimation	Power management module, clock source, current monitoring unit

3. Edge Co-Scheduling Architecture Design

3.1. Distributed Edge Collaboration Framework

To address the dynamic complexity of smart grids, a distributed edge collaboration framework is built on multi-node autonomous interconnection, enabling task distribution, resource scheduling, and adaptive state feedback through multi-level coordination between regional edge nodes and higher-level master nodes. Each node runs a lightweight agent for local state awareness, task execution, and collaboration requests, while the master node maintains a global scheduling table and resource matrix for optimal mapping and task migration. The scheduling objective follows a dual-optimization model to maximize resource utilization and minimize execution delay [3]:

$$min(\sum_{i=1}^{N} \alpha_i T_i + \sum_{j=1}^{M} \beta_j U_j^{-1})$$

where Ti denotes the actual execution time of the ith task, Uj is the current resource utilization rate of the jth node, and αi and βj are the scheduling weights. Figure 2 illustrates the control flow and

 doi:10.20944/preprints202505.2141.v1

closed-loop feedback as tasks are dispatched from the source to distributed nodes, emphasizing the benefits of multi-node collaboration in latency control and task accuracy.

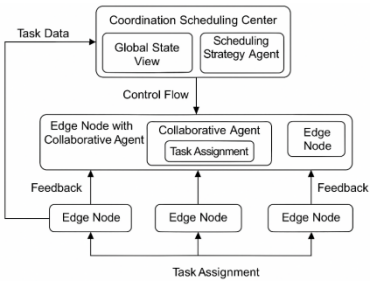

Figure 2. Architecture diagram of distributed edge co-scheduling framework.

3.2. Edge Node Communication Protocol Design

In order to realize efficient and cooperative communication among distributed edge nodes, the system adopts a multi-protocol fusion design to construct a multi-layer communication stack structure composed of MQTT, CoAP and autonomous lightweight TDCP (Time-aware Distributed Coordination Protocol). The protocol layered structure is shown in Figure 3, the bottom layer is based on LoRa and Ethernet to build physical connections, the middle layer is managed by the protocol scheduling engine for time window mapping and retransmission policy, and the high layer supports command control and state feedback.The node connection delay model based on latency weight is used in the TDCP protocol, and its expression is [4]:

$$D_{ij} = T_{tr} + T_{proc} + \frac{L_{ij}}{B_{ij}} + \lambda \cdot \sigma_{ij}$$

where Dij is the communication delay from node i to j, Ttr is the transmission initialization delay, Lij is the load size, Bij is the channel bandwidth, σij is the link volatility, and λ is the regulation factor.

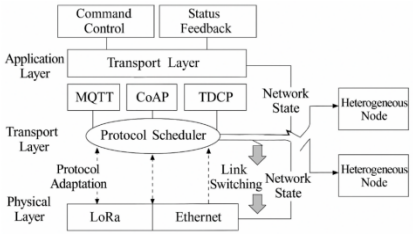

Figure 3. Diagram of multilayer edge communication protocol structure and link switching mechanism.

3.3. Side Cloud Synergy Mechanism Design

To integrate the cloud's global scheduling with the edge's real-time response, the system adopts an edge-cloud synergy mechanism based on multi-level state awareness and task offloading. Edge nodes periodically upload local resource status, communication load, and task execution rates, while the cloud control center maintains a global state graph to dynamically assign tasks, models, and policies. The edge-cloud data interaction latency is modeled as follows [5]:

$$W_{ij} = \gamma \cdot \frac{C_i}{B_{ij}} + \delta \cdot L_{ij}$$

where Wij is the offload weight from edge node i to cloud server j, Ci is the node computational complexity, Bij is the edge-cloud link bandwidth, Lij is the packet transmission delay, and γ and δ are the policy coefficients. A hybrid synchronization mechanism combining lightweight RPC and event triggering ensures high-priority tasks and large models are offloaded without blocking edge control flow. Table 2 details the modules and responsibilities of the edge-cloud collaboration,

 doi:10.20944/preprints202505.2141.v1

including data sync, model offload, policy dispatch, and state feedback, enhancing structural coherence and scheduling flexibility.

Table 2. Functional module division of Bianyun Collaboration Mechanism.

Module name	Description of the main functions	Communication Parameter Configuration
Cloud Dispatcher	Cloud command generation, resource mapping scheduling	API trigger frequency: 500ms
Edge Reporter	Edge resource uploading, node execution status packaging and sending	Data packing interval: 200ms
Model Synchronizer	AI Model Synchronization and Edge Offloading Policy Pushing	Supported compression rate: 85%
State Monitor	System status map update and event monitoring processing	State refresh cycle: 1s

4. Smart Grid Scheduling Application Design and Implementation

4.1. Grid Monitoring Data Acquisition and Processing

To enable real-time sensing of grid operation, the monitoring data acquisition layer adopts a layered, multi-source fusion architecture, covering substation terminals, feeder switch units, and distributed sensors [6]. Data is batch cached via multi-channel links with unified clock synchronization, and physical interfaces support IEC 61850, DNP3, and Modbus. Redundant data is suppressed through predefined sampling rate control. Acquisition task scheduling follows a weighted load balancing model [7]:

$$L_{total} = \sum_{i=1}^{n} \left(\frac{f_i \cdot \lambda_i}{P_i} \right)$$

where fi is the sampling frequency of the ith channel, λi is the data variability rate of the channel, Pi is the current allocatable processing capacity of the edge node, and Ltotal is the total instantaneous load estimate of the system. Processing includes filtering, anomaly detection, compression, format conversion, and event classification via edge-side stream computing engines (e.g., Flink or OpenEdge). Table 3 summarizes collection targets, protocol types, sampling rates, and preprocessing methods, demonstrating compatibility across heterogeneous sensors and consistency in processing logic.

Table 3. Grid monitoring data acquisition channel configuration table.

target collection	protocol standard	Sampling frequency (Hz)	Data pre-processing methods
Transmission section current	IEC 61850	50	Wavelet filtering + timing normalization
Distribution switch status	DNP3	1	State Trigger Event Detection
Ring Cabinet Temperature	Modbus	0.2	Sliding window mean + outlier removal
Power quality parameters	IEC 61850	25	Fast Fourier Transform (FFT) Processing

4.2. Load Prediction and Scheduling Optimization

In the load prediction stage, the smart grid scheduling system adopts a cooperative fusion mechanism combining edge-side data aggregation and cloud-based time series modeling. An LSTM variant models electricity consumption behavior, incorporating holiday and climate variables for feature crossover. The output is screened by edge nodes and sent to the cloud scheduler, which

 doi:10.20944/preprints202505.2141.v1

generates the task allocation graph via a multi-objective optimization algorithm. The scheduling objective considers load balancing, energy consumption, and task delay, defined as follows [8]:

$$min\left[\sum_{j=1}^{m}(\omega_1 \cdot P_j + \omega_2 \cdot D_j + \omega_3 \cdot V_j)\right]$$

Among them, Pj is the energy consumption of node j, Dj is its scheduling delay, Vj is the load fluctuation index, and ω1, ω2, and ω3 are weight coefficients. Scheduling uses hybrid heuristics to generate task graphs and match local migration strategies, ensuring stability while adapting to node capacity and link dynamics. Table 4 shows the heatmap of scheduling mode switching thresholds based on prediction granularity.

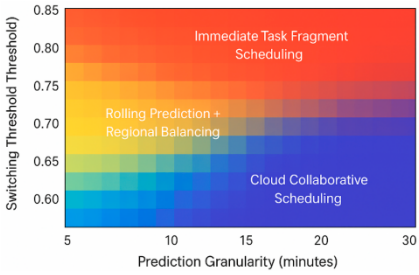

Figure 4. Thermal map of edge scheduling mode switching threshold based on prediction granularity.

4.3. Grid Security Protection

The edge scheduling system adopts a three-level security mechanism covering communication links, task scheduling, and operational data to ensure system-wide protection. To enhance communication security, nodes use a lightweight encrypted handshake based on time-window authentication, combining key rotation and node identity binding to generate dynamic symmetric key streams [9]. Scheduling security relies on a trusted task mapping mechanism, enforcing hash signature and identity verification before task distribution. The integrity check is defined as follows [10]:

$$S_i = H(T_i \| K_s \| N_t)$$

Among them, Si represents the signature digest of the i-th task, Ti is the task data body, Ks is the current session key, Nt is the timestamp seed, and H (·) is the hash function. For tamper prevention, the system employs a chain hash structure on key indicators (e.g., frequency, voltage, current) and integrates behavior recognition models to detect risks via drift rates of anomalies. Figure 5 illustrates the indicator drift detection under normal and abnormal conditions. All mechanisms are unified in a security policy mapping at the master node, enabling real-time traceability, rapid isolation, and tiered responses for distributed grid protection.

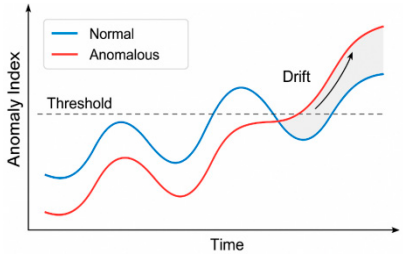

Figure 5. Abnormal Indicator Drift Detection Curve.

 doi:10.20944/preprints202505.2141.v1

5. System Validation and Performance Evaluation

5.1. Experimental Platform Construction

To validate the proposed edge-based cooperative scheduling architecture, a three-layer experimental platform is built with distributed heterogeneous node deployment, comprising edge-embedded units, a scheduling master server, and a cloud simulation environment. Hardware includes RISC-V-based domestic SoC boards with FPGA co-processors and dual-mode (5G + Ethernet) communication; the master server uses a high-performance industrial computer for task scheduling and state analysis; the cloud side runs on OpenStack to simulate task mapping and perform log analysis. Software deployment uses RT-Thread and OpenEuler across nodes, supporting MQTT, TDCP, and Modbus for protocol compatibility testing. Table 4 summarizes key configuration parameters and communication mechanisms by layer, including node type, system image, network protocol, and data sync frequency. All components connect to a unified test bus to support scheduling link initiation and state synchronization.

Table 4. Configuration table of key components of the experimental platform.

level	main component	system image	network protocol	synchronization frequency
edge node	RISC-V SoC + FPGA	RT-Thread	MQTT, Modbus	200ms
Edge Master	Industrial Controls (ARM64 Architecture)	OpenEuler	TDCP, MQTT	500ms
Cloud Simulation Environment	OpenStack Virtual Machine Cluster	Ubuntu Server	MQTT, HTTP	1s

5.2. Functional Verification

After constructing the edge cooperative scheduling platform based on the domestic RISC-V architecture, system functions are verified hierarchically, focusing on five core aspects: task distribution, edge sensing, state synchronization, link switching, and scheduling execution. Testing uses a simulated grid load scenario with dynamic task loading, where edge nodes trigger data collection, processing, and uploading to assess response consistency and scheduling integrity under multi-task and communication switching conditions. The edge node completes a closed-loop cycle—sensing, protocol parsing, and task feedback—within 200 ms, with a 98.4% success rate in mapping received state packets to execution tasks. Under link disturbances (5% 5G packet loss, LoRa bandwidth reduced by 40%), the system adapts using TDCP protocol, maintaining control command delay at 472 ms, meeting real-time requirements. Figure 6 illustrates the timing consistency between edge execution and master feedback, showing task trigger and feedback windows remain within a 500 ms fluctuation range, validating system stability in dynamic environments.

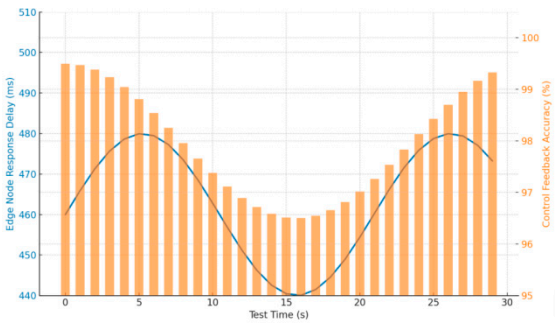

Figure 6. Consistency curve between edge node scheduling response and master control feedback state.

 doi:10.20944/preprints202505.2141.v1

5.3. Performance Evaluation

Building on functional validation, the system's performance under multi-task concurrency and link disturbances is further evaluated, focusing on scheduling latency, load balancing, and stability. A 3D visual model (Figure 7) maps task execution latency and CPU usage across load levels (30%, 60%, 90%). Results show that even with node utilization exceeding 85%, average task response time remains within 480 ms, and load distribution deviation is under 12%, indicating strong elasticity and robustness. With frequent link switching ($\geq$5 times/min), the TDCP protocol reliably handles reconnection, keeping scheduling jitter within $\pm$60 ms without tuning loss or command failures. Overall, the system demonstrates stable and continuous operation in heterogeneous, high-frequency scheduling environments, supporting its practical deployment in real grid scenarios.

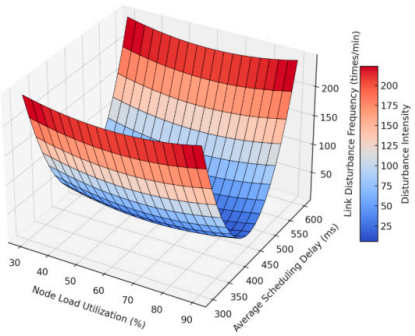

Figure 7. 3D visual mapping of system scheduling performance.

6. Conclusion

This study constructs a multilayer heterogeneous scheduling architecture for smart grids based on domestic chips, integrating hardware optimization, lightweight protocol stacks, and edge-cloud collaboration. The architecture demonstrates strong capabilities in real-time task distribution, link adaptability, and scheduling elasticity. Experimental results confirm its multitasking performance and communication reliability in dynamic grid environments. While the system achieves closed-loop operation across sensing, decision-making, and feedback, improvements are still needed in model generalization, anomaly detection, and high-frequency data processing. Future work will explore graph neural networks and cross-layer multimodal fusion to enhance applicability in scenarios such as high-density node deployment, ultra-low-latency control, and power event prediction.

References

1. Li J, Gu C, Xiang Y, et al. Edge-cloud computing systems for smart grid: state-of-the-art, architecture, and applications[J]. Journal of Modern Power Systems and Clean Energy, 2022, 10(4): 805-817.
2. Liu J, Chen J, Liu D, et al. Research on High Performance Domestic Chip Architecture Based on Digital Grid Edge Computing[C]//Proceedings of the 5th International Conference on Computer Science and Software Engineering. 2022: 695-701.
3. Su X, An L, Cheng Z, et al. Cloud–edge collaboration-based bi-level optimal scheduling for intelligent healthcare systems[J]. Future Generation Computer Systems, 2023, 141: 28-39.
4. Jin X, Wang J, Wang Z, et al. Joint multi-server cache sharing and delay-aware task scheduling for edge-cloud collaborative computing in intelligent manufacturing[J]. Wireless Networks, 2025, 31(1): 261-280.
5. Zhang Y, Tang B, Luo J, et al. Deadline-aware dynamic task scheduling in edge–cloud collaborative computing[J]. Electronics, 2022, 11(15): 2464.
6. Amer A A, Talkhan I E, Ahmed R, et al. An optimized collaborative scheduling algorithm for prioritized tasks with shared resources in mobile-edge and cloud computing systems[J]. Mobile Networks and Applications, 2022, 27(4): 1444-1460.

 doi:10.20944/preprints202505.2141.v1

7. Lin K, Gao J, Han G, et al. Intelligent blockchain-enabled adaptive collaborative resource scheduling in large-scale industrial internet of things[J]. IEEE Transactions on Industrial Informatics, 2022, 18(12): 9196-9205.
8. Feng X, Yi L, Liu N, et al. An Efficient Scheduling Strategy for Collaborative Cloud and Edge Computing in System of Intelligent Buildings[J]. Journal of Advanced Computational Intelligence and Intelligent Informatics, 2023, 27(5): 948-958.
9. Laili Y, Guo F, Ren L, et al. Parallel scheduling of large-scale tasks for industrial cloud–edge collaboration[J]. IEEE Internet of Things Journal, 2021, 10(4): 3231-3242.
10. Wang Y, Yang S, Ren X, et al. IndustEdge: A time-sensitive networking enabled edge-cloud collaborative intelligent platform for smart industry[J]. IEEE Transactions on Industrial Informatics, 2021, 18(4): 2386-2398.