Smart Parking System Architecture Design

## System Overview

The proposed smart parking system leverages a computing continuum architecture that strategically distributes computational tasks across three distinct layers: device, edge, and cloud, utilizing LoRaWAN for energy-efficient communication and MQTT for reliable edge-to-cloud messaging. This approach optimizes performance, reduces latency, minimizes energy consumption, and enhances scalability for urban smart city applications.

## Architectural Overview

A screenshot of a computer

AI-generated content may be incorrect.The system follows a three-tier computing continuum architecture designed to optimize performance, minimize energy consumption, and ensure scalability.

Figure : System architecture

### Device layer

The device layer consists of intelligent parking sensors (ultrasonic or magnetic vehicle detection sensors) deployed at individual parking spaces, enhanced with microcontrollers that allow basic data preprocessing (noise filtering, debouncing).

### Edge layer

The edge layer provides local intelligence and reduces cloud dependency through distributed processing.

* Components:
* LoRaWAN gateways with enhanced processing capabilities
* Local database for temporary data storage
* Real-time analytics engines
* MQTT communication modules
* Responsibilities:
* Data aggregation from multiple sensors
* Event filtering and duplicate removal
* Real-time occupancy analytics
* Local decision making for parking guidance
* Edge-based reservation handling
* Data forwarding to cloud via MQTT
* Key features:
* Low-latency processing for time-critical operations
* Local fault tolerance and redundancy
* Adaptive data forwarding based on network conditions
* Support for offline operation during cloud connectivity issues

### Cloud layer

The cloud layer provides scalable infrastructure for data storage, user management, and advanced analytics.

* Components:
* API servers for external integrations
* Centralized database systems
* User application backends
* Payment processing systems
* Responsibilities:
* Long-term data storage and archival
* User account management and authentication
* Integration with external services (navigation, payment)
* Historical trend analysis and reporting
* Key features:
* Scalable infrastructure with auto-scaling capabilities
* High availability and disaster recovery
* Advanced analytics and business intelligence
* Integration of APIs for third-party services

## Protocol Integration

The system integrates two complementary communication protocols to ensure reliable, energy-efficient, and scalable data exchange across the computing continuum:

* LoRaWAN was selected for device-to-edge communication because it provides long-range coverage (kilometres in urban areas) while consuming very little power, enabling battery-powered parking sensors to operate for years without maintenance. Its adaptive data rate and AES-128 encryption ensure efficient and secure transmission even in dense urban environments.
* MQTT was chosen for edge-to-cloud communication because it is lightweight, efficient, and well-suited for unreliable or bandwidth-limited networks. Its publish/subscribe model allows gateways to send only relevant, aggregated data to the cloud, reducing bandwidth usage and latency. MQTT’s Quality of Service (QoS) levels also allow dynamic reliability control depending on network stability.

LoRaWAN handles low-power, long-range sensing data up to the gateway, while MQTT efficiently carries processed, high-value data from the gateway to the cloud. This combination minimizes energy use, network load, and response times across the continuum.

## Task Distribution

The system implements intelligent task distribution across the three-layer computing continuum based on computational complexity, latency requirements, and resource constraints.

* Device layer (sensors + microcontrollers): Handles lightweight, time-critical preprocessing directly at the parking spot. Tasks:
  + Collect raw occupancy data (ultrasonic/magnetic detection)
  + Apply noise filtering, debouncing, and threshold-based occupancy
  + Compress data before transmission via LoRaWAN
  + Battery monitoring
* Edge layer (LoRaWAN gateways + local processing): Provides real-time decision-making close to the data source, reducing cloud dependency. Tasks:
  + Aggregate and correlate data from multiple sensors
  + Remove duplicates and filter events
  + Real-time parking space availability mapping
  + Local parking guidance algorithms
  + Short-term parking pattern analysis (last 1-2 hours)
  + Prioritize and forward selected data to the cloud over MQTT
* Cloud layer (centralized platform): Manages long-term storage, analytics, and integration with user-facing applications. Tasks:
  + Long-term historical data storage and analysis
  + User account management and authentication
  + Payment processing integration
  + Mobile/web application backend services
  + Integration with navigation systems (Google Maps, Waze, etc.)
  + Provide dashboards and reports for administrators and planners

## Orchestration Policies

The orchestration system employs a policy-based engine to dynamically allocate computational tasks across the device, edge, and cloud layers in the computing continuum. Decisions are based on latency requirements, resource availability, network conditions, and operational context, ensuring optimal performance, energy efficiency, and scalability.

1. Latency based orchestration:
   * Critical operations such as tasks requiring response times <500ms are executed at the edge or device layer (including real-time parking availability updates and local guidance).
   * Standard operations (<2s) like reservation requests or route guidance are executed at the edge to balance responsiveness and resource use; may fall back to cloud if edge is overloaded.
   * Background operations (>5s) like analytics and reporting or historical data processing are executed in the cloud.
2. Resource based orchestration:
   * CPU load thresholds: Tasks migrate to higher layers if local processing exceeds defined CPU/memory usage limits (e.g., edge overload triggers cloud execution).
   * Battery-aware tasking: For battery-powered devices, intensive preprocessing is offloaded to the edge when battery level drops below a threshold.
   * Bandwidth adaptation: When LoRaWAN link quality degrades or bandwidth is constrained, non-critical data is queued locally or aggregated before transmission.
3. Context aware orchestration:
   * Traffic demand patterns: During peak demand, edge nodes prioritize real-time decision-making and suppress non-urgent cloud transmissions.
   * Fault tolerance mode: In case of cloud connectivity loss, the edge layer switches to autonomous operation, retaining local control over reservations and availability updates until reconnection.
   * Environmental factors: Adapts task distribution based on sensor reliability, interference levels, or gateway availability.
4. Policy enforcements mechanism:
   * Rule prioritization: Policies are applied hierarchically, with latency constraints taking precedence over resource optimization.
   * Monitoring loop: Continuous health checks on network latency, CPU load, memory, and power status inform orchestration adjustments in near real-time.
   * Fallback protocols: MQTT QoS levels are dynamically adjusted (e.g., switching from QoS 0 to QoS 1) when network reliability decreases, ensuring critical message delivery.

## Workflow Integration

The end-to-end workflow integrates all components as shown in the sequence diagram:

Sequence diagram


Figure : System sequence diagram showing component interaction

1. User Request: The user initiates a parking search through the mobile app, sending a Request parking info command.
2. Mobile App → Cloud Service: The mobile app forwards a Query availability request to the cloud service to get the most up-to-date parking data.
3. Cloud Service → Edge Gateway: The cloud service sends a Get latest data request to the edge gateway to ensure it has current occupancy information.
4. Edge Gateway → Parking Sensor: The edge gateway polls the parking sensor for its latest status.
5. Parking Sensor → Edge Gateway: The parking sensor sends back its occupancy status (e.g., occupied or free).
6. Edge Gateway → Cloud Service: The edge gateway aggregates sensor data into aggregated data and forwards it to the cloud service.
7. Cloud Service → Mobile App: The cloud service processes the aggregated data, determines available spots, and sends this list back to the mobile app.
8. Results Display: The mobile app displays the results to the user, showing available parking spaces.

## Use Cases and Actors

* Primary actors:
  + Driver: Individual seeking parking space
  + Administrator: System operator managing the parking infrastructure
  + City planner: Official using data for urban planning decisions
  + Business owner: Commercial entity integrating parking data

A diagram of parking system

AI-generated content may be incorrect.

Figure : Use case diagram

* Use case 1: Find available parking

Description: Driver searches for available parking spaces in a specific area.

Preconditions:

- User has mobile app installed and registered

- GPS location services are enabled

- Internet connectivity is available

Main flow:

- Driver opens mobile application

- System retrieves current location via GPS

- System queries real-time parking availability within search radius

- System displays available spaces

- Driver selects preferred parking option

- System provides navigation directions to selected space

Postconditions:

- Driver receives navigation to available parking space

- System tracks space selection for optimization

Alternative flows:

- No spaces available - system suggests nearby alternatives

- GPS unavailable - user manually enters location

- Network connectivity issues - system uses cached data

* Use case 2: Reserve parking space

Description: Driver reserves a specific parking space for future use.

Preconditions:

- User is authenticated in the system

- Selected parking space supports reservations

- Payment method is configured

Main flow:

- Driver selects "Reserve" option for chosen space

- System displays availability calendar and pricing

- Driver selects desired time slot and duration

- System calculates total cost

- Driver confirms reservation and payment method

- System processes payment and confirms reservation

- Driver receives confirmation with reservation details

Postconditions:

- Parking space is reserved for specified time

- Payment is processed and confirmed

- Reservation notification is sent to driver

* Use case 3: Monitor system health

Description: Administrator monitors overall system performance and health.

Preconditions:

- Administrator has valid credentials and access privileges

- Monitoring dashboard is operational

Main flow:

- Administrator logs into administrative dashboard

- System displays real-time system health metrics

- Administrator reviews sensor status, gateway connectivity, and cloud services

- System highlights any anomalies or issues requiring attention

- Administrator can drill down into specific components for detailed analysis

- Administrator can acknowledge alerts and assign maintenance tasks

Postconditions:

- System health status is updated

- Any issues are documented and tracked

## Functional Requirements

The system should:

* be able to detect vehicle presence/absence with at least 95% accuracy
* update parking status in real-time
* support parking space reservations
* provide secure user registration and authentication
* prevent double-booking of spaces
* provide navigation to parking spaces
* provide real-time dashboard monitoring

## Non-Functional Requirements

* Response time
* Throughput
* Scalability
* System availability
* Fault tolerance
* Data integrity
* Authentication and authorization
* Data protection

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a phone

AI-generated content may be incorrect.