OCPP EV Charging Server with Dynamic Load Balancing

Project Overview

This is a **full-stack EV charging management system** that implements the OCPP 1.6 protocol with advanced Dynamic Load Balancing (DLB) capabilities. The system enables intelligent power distribution across multiple EV chargers while optimizing solar energy usage and preventing grid overload.

Key Features

1. **OCPP 1.6 Protocol Implementation**

- Full WebSocket-based communication with EV chargers
- Remote start/stop charging control
- Real-time meter value monitoring (voltage, current, power, energy)
- Transaction management with session tracking
- Status notifications and heartbeat monitoring

2. **Dynamic Load Balancing (DLB) System**

Four intelligent charging modes:

- **PV Dynamic Balance**: Prioritizes solar energy consumption, adjusts charging based on grid import/export
- **Extreme Mode**: Maximum power charging (32A) regardless of conditions
- **Night Full Speed**: Automatic full-speed charging during off-peak hours (22:00-06:00)
- **Anti Overload**: Safety mechanism preventing main fuse overload (60A limit)

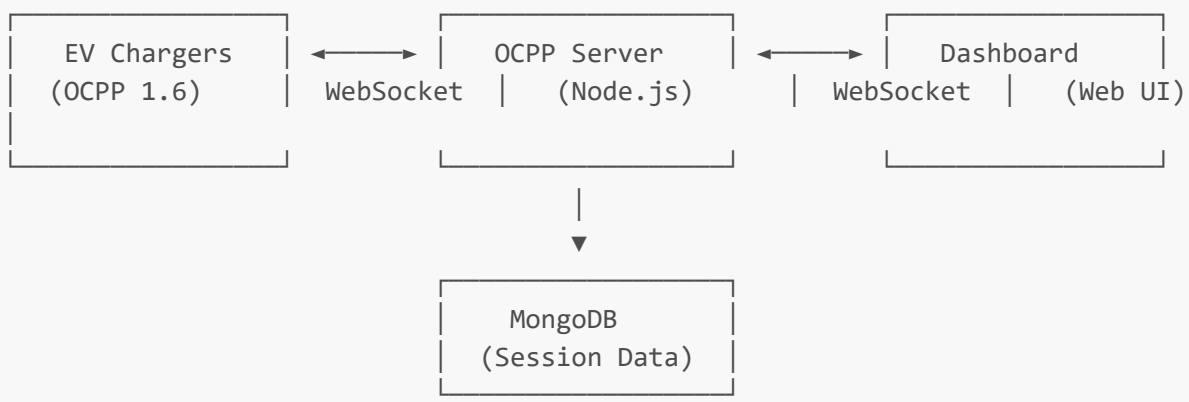
3. **Real-Time Dashboard**

- Live charger status monitoring
- Power flow visualization with animated indicators
- Interactive DLB controls with toggle switches
- Charging history with daily/monthly/yearly views
- Multi-charger support with device list

4. **Data Persistence**

- MongoDB integration for session history
- IST timezone support for Indian users
- Energy consumption tracking and reporting

System Architecture



Components:

1. **server.js** (Core OCPP Server)
    - WebSocket server for charger communication
    - HTTP server for dashboard and API
    - DLB calculation engine
    - Transaction management
2. **dashboard.html** (Web Interface)
    - Real-time charger monitoring
    - DLB visualization with power flow animation
    - History charts and statistics
    - Remote control interface
3. **database.js** (Data Layer)
    - MongoDB connection management
    - Session CRUD operations
    - Period-based data queries
4. **charger-simulator.js** (Testing Tool)
    - Simulates OCPP 1.6 chargers
    - Generates realistic meter values
    - Simulates DLB sensor data (Grid, PV, Home load)

Dynamic Load Balancing Logic

Power Flow Calculation

```
Available Power = (Main Fuse Capacity - Home Load - Safety Margin)
Target Charging Current = f(Solar Power, Grid Power, Mode Settings)
```

Mode Priority Hierarchy

1. **Extreme Mode** (Highest) → Always 32A
2. **Night Full Speed** → 32A during 22:00-06:00
3. **PV Dynamic Balance** → Adjusts based on solar excess
4. **Anti Overload** (Safety Layer) → Caps all modes to prevent overload

Real-Time Adjustment

- DLB recalculates every 10 seconds when meter values arrive
- Sends `SetChargingProfile` OCPP command to charger
- Adjusts power limit dynamically based on:
  - Grid import/export (positive = importing, negative = exporting)
  - Solar generation (0-10kW based on time of day)
  - Home consumption (3-8kW variable)
  - Active charging mode settings

🔧 Technical Stack

Layer	Technology
Backend	Node.js, WebSocket (ws library)
Database	MongoDB (Mongoose)
Frontend	Vanilla HTML/CSS/JavaScript
Protocol	OCPP 1.6 (JSON over WebSocket)
Communication	Bidirectional WebSocket (Server ↔ Charger, Server ↔ Dashboard)

📊 Data Flow

Charging Session Flow

```
1. Charger connects → BootNotification
2. Dashboard sends RemoteStartTransaction
3. Charger responds → StartTransaction
4. Meter values sent every 10s → MeterValues
5. DLB calculates optimal power → SetChargingProfile
6. Dashboard sends RemoteStopTransaction
7. Charger responds → StopTransaction
8. Session saved to MongoDB
```

DLB Data Flow

```
1. Charger sends MeterValues with DLB measurands:
   - Power.Active.Import.Grid (Grid power)
   - Power.Active.Import.PV (Solar power)
   - Power.Active.Import.Home (Home load)

2. Server calculates target charging current based on active modes

3. Server sends SetChargingProfile to charger with power limit

4. Server broadcasts DLB state to all dashboards via WebSocket

5. Dashboard updates power flow visualization
```

Dashboard Features

1. **Device List View**

- Shows all connected chargers
- Real-time status (Online, Charging, Offline)
- Current power, voltage, current readings
- Session energy and duration

2. **DLB Visualization**

- Hub-and-spoke power flow diagram
- Animated dots showing power direction
- Real-time power values for Grid, PV, Charger, Home
- Bidirectional grid flow (import/export)

3. **History View**

- Per-charger session history
- Daily/Monthly/Yearly energy charts
- Total energy consumption statistics
- Session details (start/end time, duration, energy)

4. **DLB Controls**

- Toggle switches for all 4 modes
- Real-time mode updates across all dashboards
- Visual feedback for active modes

Testing & Simulation

Charger Simulator Features

- Simulates OCPP 1.6 protocol messages

- Generates realistic charging data:
  - Voltage: 230V ± 5V
  - Current: 15-20A (when charging)
  - Power: Voltage × Current
- Simulates time-based solar generation (peak at 10:00-14:00)
- Variable home load (3-8kW)
- Calculated grid import/export

Testing Workflow

1. Start server: `npm start`
2. Start simulator: `npm run simulator` or `node charger-simulator.js CP001`
3. Open dashboard: `http://localhost:9000`
4. Test remote start/stop
5. Toggle DLB modes and observe power adjustments

Use Cases

1. **Solar-Optimized Charging**

- Maximizes self-consumption of solar energy
- Reduces grid import costs
- Automatically adjusts charging based on solar availability

2. **Multi-Charger Management**

- Supports multiple chargers simultaneously
- Fair power distribution across chargers
- Prevents total load from exceeding capacity

3. **Time-of-Use Optimization**

- Night Full Speed mode for cheap off-peak electricity
- Automatic mode switching based on time

4. **Grid Safety**

- Anti Overload prevents circuit breaker trips
- Real-time monitoring of total site load
- Safety margins to account for load spikes

Deployment

Local Development

```
npm install
npm start
```

Production Considerations

- Use environment variables for MongoDB connection
- Implement authentication for dashboard access
- Add HTTPS/WSS for secure communication
- Configure firewall rules for charger access
- Set up monitoring and logging

Security Features

- WebSocket connection validation
- Charger ID-based session management
- Transaction ID verification
- Error handling and recovery
- Automatic reconnection logic

Future Enhancements

- [ ] User authentication and authorization
- [ ] Multi-site support
- [ ] Mobile app integration
- [ ] Advanced scheduling (calendar-based)
- [ ] Cost optimization algorithms
- [ ] Integration with energy tariff APIs
- [ ] Push notifications for session events
- [ ] Export reports (PDF, CSV)

Learning Outcomes

This project demonstrates:

- Real-time bidirectional communication (WebSocket)
- Protocol implementation (OCPP 1.6)
- Complex state management
- Database integration and querying
- Responsive web design
- Algorithm design (DLB logic)
- System architecture and design patterns
- Testing and simulation strategies

Project Stats

- **Lines of Code**: ~3,500+
- **Files**: 4 main files (server.js, dashboard.html, database.js, charger-simulator.js)

- **Supported Chargers**: Unlimited (scalable)
- **Real-time Updates**: Every 10 seconds
- **Database**: MongoDB with session persistence
- **Timezone**: IST (UTC+5:30) support

Key Achievements

- [x] Full OCPP 1.6 protocol implementation
- [x] 4 intelligent DLB modes with real-time switching
- [x] Bidirectional power flow visualization
- [x] Multi-charger support with independent control
- [x] Session history with timezone-aware timestamps
- [x] Automated testing with realistic simulator
- [x] 100% test pass rate for DLB toggles
- [x] Responsive, modern web dashboard

Innovation Highlights

1. **Bidirectional Grid Animation**: Visual representation of import/export with direction-aware animated dots
2. **Real-Time DLB Sync**: All dashboards update instantly when modes change
3. **Solar-Aware Charging**: Intelligent power adjustment based on PV generation
4. **Safety-First Design**: Anti Overload mode prevents electrical hazards
5. **IST Timezone Support**: Localized timestamps for Indian deployment