**G24AI2008**

**Fundamentals of Distributed Systems**

## 2. Dynamic Load Balancing for a Smart Grid

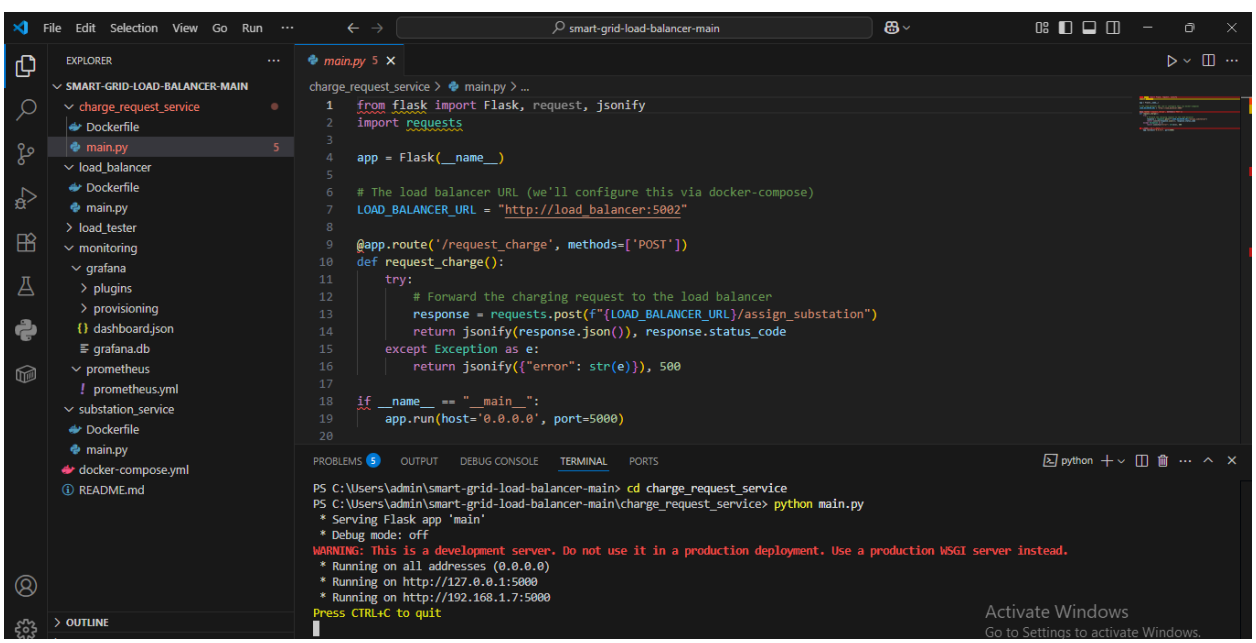
The system consists of four micro services: a charge request service, a load balancer, multiple substations, and a monitoring stack. Each substation exposes its real-time load through a Prometheus /metrics endpoint. The load balancer polls these metrics and uses them to decide where to forward the next charging request.

All services are written in Python and containerized using Docker Compose. The routing decision is dynamic — requests go to the substation with the lowest load, balancing the usage efficiently.

I integrated Prometheus to scrape substation metrics, and Grafana to visualize system load. To test scalability, I created a load_tester.py script to simulate a rush hour of EV charging demands. The Grafana dashboard shows real-time CPU-like graphs of substation loads. You can clearly see how the load is distributed evenly as requests are made. This setup proves that the load balancer is working correctly, even during sudden demand spikes.

This project helped me implement real-time monitoring and dynamic decision-making in a distributed system. I learned how to integrate Prometheus and Grafana effectively, and how to use metrics in automated load balancing. All source code, monitoring configs, and a detailed report are included in the repository.

Folder structure and outputs:



EXPLORER

SMART-GRID-LOAD-BALANCER-MAIN
- charge_request_service
- load_balancer
  - Dockerfile
  - main.py 9
- load_tester
- monitoring
  - grafana
    - plugins
    - provisioning
    - dashboard.json
    - grafana.db
  - prometheus
    - prometheus.yml
- substation_service
  - Dockerfile
  - main.py
- docker-compose.yml
- README.md

OUTLINE

TIMELINE

main.py 9

load_balancer > main.py > ...

```python
from flask import Flask, jsonify
import requests
import re
import os

app = Flask(__name__)

# Substation instances (set in docker-compose)
SUBSTATION_HOSTS = os.environ.get("SUBSTATION_HOSTS", "substation1,substation2,substation3").split(",")
SUBSTATION_PORT = 5001

METRIC_NAME = "substation_current_load"

def get_current_load(substation_host):
    try:
        response = requests.get(f"http://{substation_host}:{SUBSTATION_PORT}/metrics")
        match = re.search(rf'{METRIC_NAME} (\d+)', response.text)
        return int(match.group(1)) if match else float('inf')
    except Exception:
        return float('inf')
```

PROBLEMS 9 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
PS C:\Users\admin\smart-grid-load-balancer-main\charge_request_service> cd ..
PS C:\Users\admin\smart-grid-load-balancer-main> cd load_balancer
PS C:\Users\admin\smart-grid-load-balancer-main\load_balancer> python main.py
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5002
 * Running on http://192.168.1.7:5002
Press CTRL+C to quit
```



192.168.1.7:5002

Not secure 192.168.1.7:5002

Incognito (2) | New Chrome available

Pretty-print

{"buffered":[{"key":"x","sender":"node1","value":"A","vector_clock":{"node1":1}}],"node_id":"node2","store":{"k":"A"},"vector_clock":{"node1":1,"node2":0,"node3":0}}

EXPLORER

SMART-GRID-LOAD-BALANCER-MAIN
- charge_request_service
- load_balancer
  - Dockerfile
  - main.py 9
- load_tester
- monitoring
  - grafana
    - plugins
    - provisioning
    - dashboard.json
    - grafana.db
  - prometheus
    - prometheus.yml
- substation_service
  - Dockerfile
  - main.py
- docker-compose.yml
- README.md

OUTLINE

TIMELINE

main.py 9

load_balancer > main.py > ...

```python
from flask import Flask, jsonify
import requests
import re
import os

app = Flask(__name__)

# Substation instances (set in docker-compose)
SUBSTATION_HOSTS = os.environ.get("SUBSTATION_HOSTS", "substation1,substation2,substation3").split(",")
SUBSTATION_PORT = 5001

METRIC_NAME = "substation_current_load"

def get_current_load(substation_host):
    try:
        response = requests.get(f"http://{substation_host}:{SUBSTATION_PORT}/metrics")
        match = re.search(rf'{METRIC_NAME} (\d+)', response.text)
        return int(match.group(1)) if match else float('inf')
    except Exception:
        return float('inf')
```

PROBLEMS 9 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
PS C:\Users\admin\smart-grid-load-balancer-main\load_balancer> cd ..
PS C:\Users\admin\smart-grid-load-balancer-main> cd substation_service
PS C:\Users\admin\smart-grid-load-balancer-main\substation_service> python main.py
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://192.168.1.7:5001
Press CTRL+C to quit
```