Iowa State University
College of Engineering

Energy-Aware Real-Time Scheduling: Comparing EDF, Static EDF, and Cycle-Conserving EDF Algorithms

| Christy Mettry

| Justin Wenzel

| 12/11/24

Iowa State University
College of Engineering

Problem Statement

Challenge of Energy-Efficient Real-Time Scheduling

- Real-time systems (e.g., IoT devices, automotive systems, etc.) operate with **strict deadlines**
- **Current Issues:**
  - High energy consumption in real-time scheduling
  - EDF is effective for ensuring deadlines are met, but energy-inefficient due to fixed worse-case execution time (WCET) assumptions
- **Goal:** Explore EDF algorithms variances to balance **energy efficiency** and **deadline adherence**

Iowa State University
College of Engineering

Solution: Enhancing EDF for Energy Efficiency

Challenge:

- EDF is not energy efficient due to WCET and high-frequency operation

Approach:

- Extend EDF with Static EDF and CC-EDF to optimize energy usage
- Introduce frequency scaling (Static EDF) and dynamic adjustment (CC-EDF)

Ideal Outcomes:

- **Reduced Energy Consumption:** through frequency scaling and slack utilization
- **Maintain Deadlines:** Real-time performance preserved
- **Improved Adaptability:** Response to real-time task execution

Solution: Algorithm Overview

Static EDF:

- Fixed frequency scaling based on task utilization
- Balances energy and deadlines but lacks adaptability

CC-EDF:

- Dynamically frequency adjustments based on real-time slack
- Maximizes energy saving but introduces

Feature	EDF	Static EDF	CC-EDF
Frequency Scaling	None	Fixed	Dynamic
Adaptability	None	Limited	High
Energy Efficiency	Low	Moderate	High
Implementation	Simple	Simple	Complex

Iowa State University
College of Engineering

Implementation: Architecture

Programming Language: Python

Modules Developed:

- config.py: Task definitions, simulation settings, frequency & power settings
- task.py: Define task attributes (name, period, WCET, etc.)
- edf.py: Implements EDF, Static EDF, and CC-EDF algorithm assistance functions
- main.py: Controls simulation workflow and collects results
- interafce.py: GUI interaction for users

Design Principles:

- Modular design for easy maintenance

IOWA STATE UNIVERSITY
College of Engineering

Implementation – Technical Implementation Features

Task Class (task.py):

- Execute: Reduces remaining time
- Reset: Prepares task for next period

Scheduler Class (edf.py):

- Manages a priority queue (heapq) for tasks
- Adjusts CPU frequency dynamically for CC-EDF
- Tracks metrics: energy usage, idle time, missed deadlines

Configurable Settings (config.py):

- Frequency and power levels for CPU scaling
- Execution time variability for CC-EDF (user-adjustable)

Main and Interface:

- GUI for usability and visual results

```python
# Task definitions with execution times and periods
TASKS = [
    {"Sensor": "Task1", "execution_time": 1, "period": 20},
    {"Communication": "Task2", "execution_time": 1, "period": 15},
]

# Simulation settings
SIMULATION_DURATION = 100  # Total simulation time in seconds
TIME_QUANTUM = 1  # Time slice for the scheduler (1 second per unit)

# Processor frequency and power settings (GHz: Watts)
AVAILABLE_FREQUENCIES = {
    1.0: 25,   # 25 watts at 0.5 GHz
    1.5: 50,   # 50 watts at 1.0 GHz
    1.75: 75,   # 75 watts at 1.5 GHz
    2.0: 100,  # 100 watts at 2.0 GHz
}

IDLE_POWER = 10  # Idle power in watts

# CC-EDF Execution Time Variability (Percentage Range)
CC_EDF_EXECUTION_TIME_RANGE = {
    "min_percent": 50,  # Minimum execution time as a percentage of WCET
    "max_percent": 100  # Maximum execution time as a percentage of WCET
}
```

Config.py Example

Testing/Evaluation – Simulation

Workload Categories:

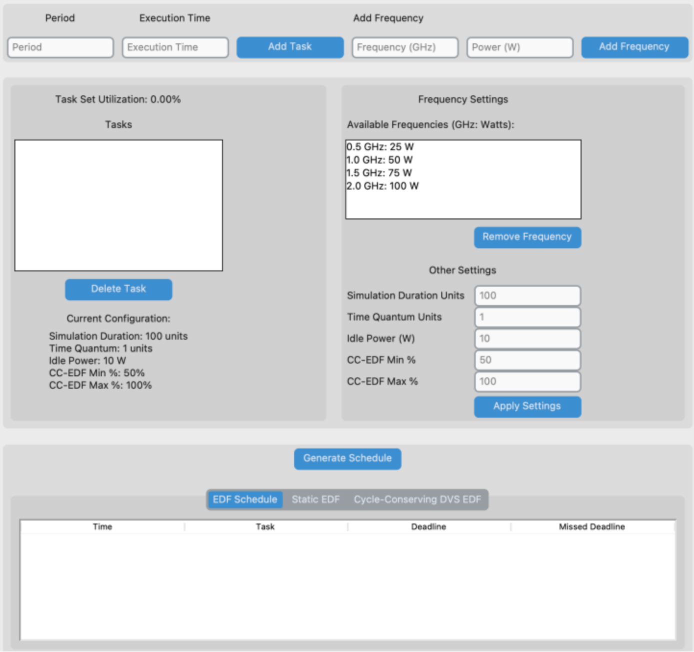
- Low, medium, high utilization

Task Sets:

- 15 total set (5 per category)

Metrics Evaluated:

- Energy Consumption
- Idle Time (CPU Inactive)
- Missed Deadlines



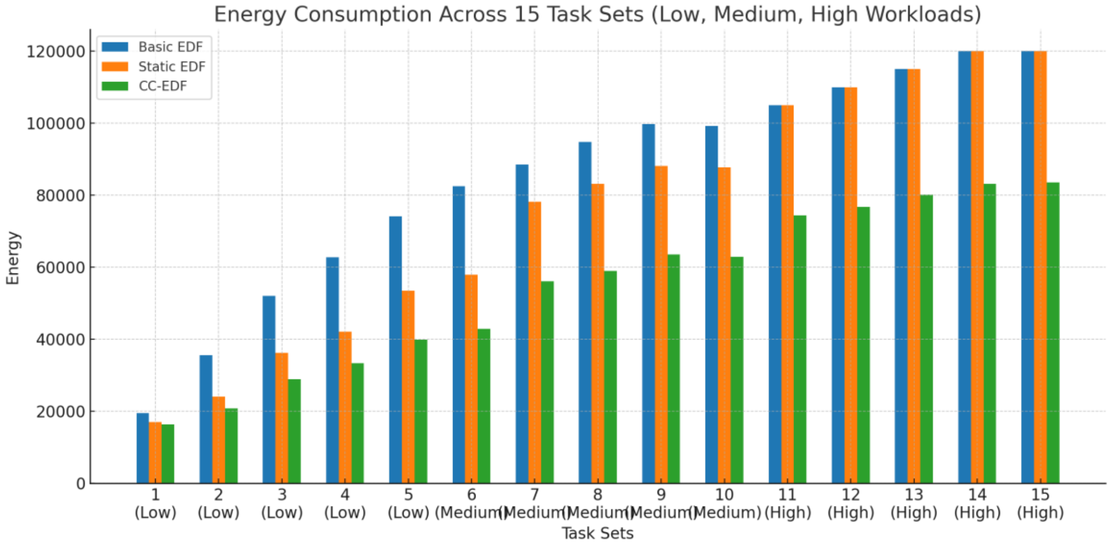
Simulator GUI

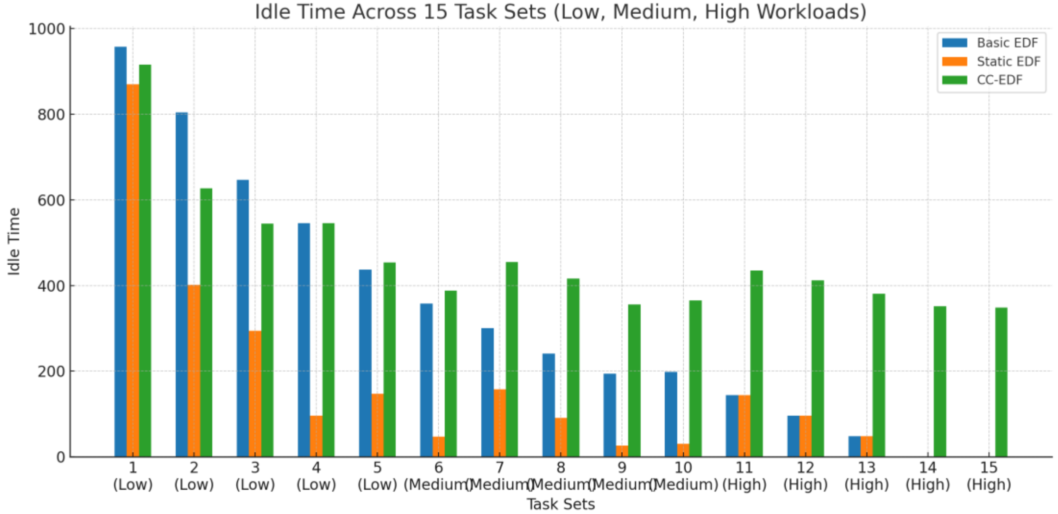
Simulator Setup:

Parameter	Value
Time Quantum	0.1 units
Simulation Duration	1000 units
Idle Power	15 units
CC-EDF Variability	Tasks finish in 50-80% of their WCET

Iowa State University
College of Engineering

Testing/Evaluation – Results





Conclusion

Basic EDF:

- High energy consumption and idle time

Static EDF:

- Reduced energy consumption compared to Basic EDF
- Lower idle time with fixed frequency scaling

CC-EDF:

- Achieve the **lowest energy consumption** with dynamic adjustment

Key Takeaways:

- All algorithms met deadlines across the workload
- **CC-EDF** is the most efficient for energy savings, especially with high utilization
- **Static EDF** reduces idle time significantly at times

Iowa State University
College of Engineering

Learning and Takeaways

Further Real-Time Scheduling:

- Learned scheduling algorithms in depth (CC-EDF)
- Explored frequency scaling affects with energy consumption and idle time

Key Insights:

- Dynamic adjustments can make a major impact in achieving optimal energy savings
- Trade-offs exist in each algorithms for energy efficienty, idle time, and balancing system needs

Practical Experiences:

- Designed and tested a simulator to evaluated real-time scheduling algorithms
- Insights toward power-aware computing and real-time scheduling