ESPX - Assignment 1

Savvas Tzanetis - 10889

May 2025

1 Preperations

This report is part of the first assignment for the **Real Time Embedded Systems** class of the **Electrical and Computer Engineering** department at the **Aristotle University of Thessaloniki**.

The task of this assignment was to modify an existing implementation the *Consumer-Producer Problem* written in the **C** programming language, in order to produce "work functions" that the consumer threads must execute.
More specifically, the modified codes producer threads parse to the shared queue work functions that compute the sin of a set of *10* angles that are also parsed to the queue as function arguments.

2 Results Evaluation

Using **gettimeofday()**, we record the time that a *work function* is added to the queue from the producer thread, as well as the time the same function is removed from the queue from a consumer. The results are gathered using 10 or 20 producer threads with a varying amount of consumer threads.

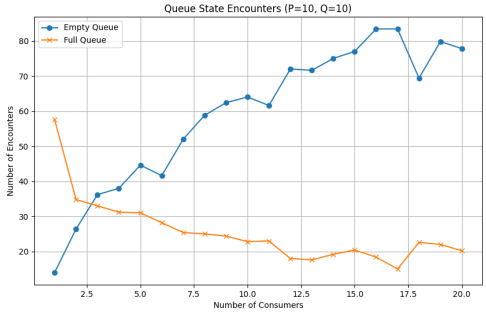

Figure 1: Queue state with 10 producers

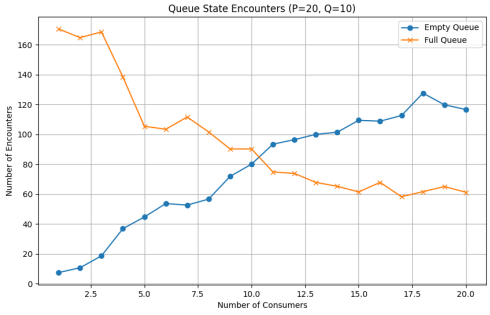

Figure 2: Queue state with 20 producers

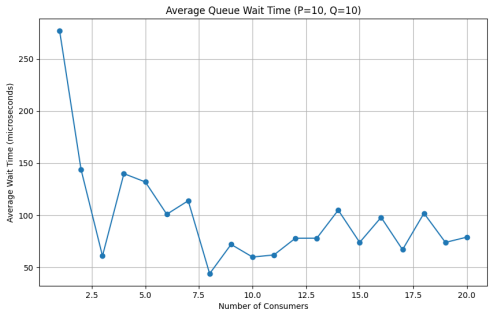

Figure 3: Average waiting time with 10 producers

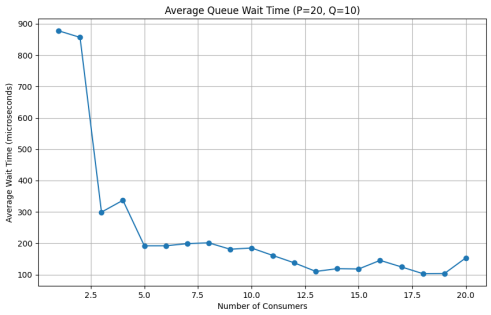

Figure 4: Average waiting time with 20 producers

2.1 Results Analysis

Analyzing the presented figures reveals several key insights into the performance characteristics of our producer-consumer implementation.

In figures **1** and **2**, we show the queue state statistics with 10 and 20 producers respectively. In both scenarios, we find that the frequency of contacts with an empty queue increases continuously as the number of consumer threads increases. The trend gets more noticeable when 20 producers are used. Encounters in the full queue begin much higher and decline more gradually.

Figures **3** and **4** shows us expected results, revealing the relationship between consumer and producer thread count. With 10 producers, the average wait time shows an overall declining trend as consumer threads increase, with notable variability that could be suggesting diminishing returns and potentially increased scheduling overhead, while with 20 producers, we can observe significantly higher overall wait times, particularly with fewer consumers, while also dropping wait times more dramatically as more consumers are added.

These results provide practical guidelines for tuning thread counts in producer-consumer systems:

- For optimal performance with fixed queue sizes, consumer threads should typically be close to equal to the number of producer threads.
- Systems should avoid extreme imbalances between producer and consumer threads, as these lead to either queue saturation or resource wastage.

2.2 System Specifications

The evaluation results where produced on a **Windows 11** machine with the following specifications:

- **6-Core** Intel Core i5 CPU with **hyper-threading** running at **2.4Ghz**
- **16GB** RAM