

# Traffic Control Simulation: Producer-Consumer Design

This document outlines the design, data structures, and concurrency strategy for implementing the multi-threaded bounded-buffer producer-consumer pattern to simulate traffic control analysis.

## 1. Data Structures and Thread Safety

Data Structure	Purpose	Location	Thread-Safe?	Concurrency Strategy
<b>TrafficData Struct</b>	Holds one traffic signal measurement (timestamp, ID, count).	Shared & Local	Yes (Immutable)	Passed by value/const reference.
<b>Bounded Buffer (std::queue&lt;TrafficData&gt;)</b>	The shared queue between producers and consumers.	Shared	Yes (Blocking)	Protected by std::mutex and controlled by two std::condition_variables.
<b>std::mutex (Buffer Lock)</b>	Protects the bounded buffer's critical sections (push and pop).	Shared		Used with std::unique_lock for both blocking and non-blocking access.
<b>std::condition_variable (Full/Empty)</b>	Blocks producers when the buffer is full and consumers when the buffer is empty.	Shared		Ensures efficient waiting without busy-looping.
<b>Congestion Map (std::map&lt;int, long long&gt;)</b>	Tracks the total car count for every traffic_light_id.	Shared	Yes (Non-Blocking)	Protected by a separate std::mutex (Map Lock) for quick, non-blocking updates.
<b>std::mutex (Map Lock)</b>	Protects the shared Congestion Map	Shared		Ensures atomic update of traffic counts.

	during consumers' update operations.			
--	--------------------------------------	--	--	--

## 2. Producer-Consumer Mechanism

### Bounded Buffer Implementation

The buffer is implemented using a std::queue with a fixed CAPACITY (set to 50 in the code).

- **Blocking Producer:**

1. Acquires lock on the buffer.
2. Waits on the **cond\_producer** condition variable if the buffer is **full**.
3. Pushes the data onto the queue.
4. Notifies the **cond\_consumer** condition variable to wake up waiting consumers.

- **Blocking Consumer:**

1. Acquires lock on the buffer.
2. Waits on the **cond\_consumer** condition variable if the buffer is **empty**.
3. Pops the data from the queue.
4. Notifies the **cond\_producer** condition variable to wake up waiting producers.

This design ensures that threads only consume cycles when there is work to do, preventing race conditions and deadlocks.

## 3. Simulation Parameters

- **Traffic Signals (X):** 20
- **Measurements per hour:** 12 (one every 5 minutes)
- **Data File:** Reads data from traffic\_data.txt.
- **Top N:** The consumers collaboratively track the top 5 most congested lights.