**11. WAP to Implement Congestion control protocols: Leaky Bucket**

**Steps to Implement the Leaky Bucket Algorithm:**

1. **Define the bucket size**: Maximum capacity of the bucket (in packets or bits).
2. **Define the leak rate**: The rate at which data leaves the bucket (in packets per second).
3. **Incoming Data**: Data arrives at irregular intervals.
4. **Bucket Behavior**:
   * If the bucket is not full, the incoming data is added to the bucket.
   * If the bucket is full, excess data is discarded (overflow).
5. **Leaking Process**: The data leaks out at a constant rate, regardless of the incoming data rate.

**C Program to Implement the Leaky Bucket Algorithm**

This program simulates a leaky bucket where data is arriving at random intervals, and the bucket leaks at a constant rate.

c

Copy code

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define MAX\_BUCKET\_SIZE 10 // Maximum bucket capacity (in packets)

#define LEAK\_RATE 1 // Leak rate (in packets per second)

int bucket\_size = 0; // Current size of the bucket

// Function to simulate data arriving at random intervals

void simulateIncomingData() {

// Simulate the random arrival of data packets

int data\_arrival = rand() % 3 + 1; // Random data arrival between 1 and 3 packets

printf("Incoming data: %d packets\n", data\_arrival);

if (bucket\_size + data\_arrival <= MAX\_BUCKET\_SIZE) {

bucket\_size += data\_arrival; // Add data to the bucket

printf("Data added to the bucket. Current bucket size: %d packets\n", bucket\_size);

} else {

int overflow = (bucket\_size + data\_arrival) - MAX\_BUCKET\_SIZE;

bucket\_size = MAX\_BUCKET\_SIZE; // Bucket is full, add only up to the max size

printf("Bucket overflow! %d packets discarded.\n", overflow);

printf("Current bucket size: %d packets\n", bucket\_size);

}

}

// Function to simulate the leak process

void leakData() {

if (bucket\_size > 0) {

bucket\_size -= LEAK\_RATE; // Leak data at the fixed rate

if (bucket\_size < 0) {

bucket\_size = 0; // Bucket can't be negative

}

printf("Data leaked at rate %d. Current bucket size: %d packets\n", LEAK\_RATE, bucket\_size);

}

}

int main() {

printf("Leaky Bucket Congestion Control Simulation\n");

printf("Max bucket size: %d packets\n", MAX\_BUCKET\_SIZE);

printf("Leak rate: %d packets per second\n", LEAK\_RATE);

while (1) {

// Simulate incoming data every 1 second

simulateIncomingData();

// Simulate the leaking of data every 1 second

leakData();

// Wait for 1 second to simulate time intervals

sleep(1);

}

return 0;

}

**Explanation of the Program:**

1. **Variables**:
   * MAX\_BUCKET\_SIZE: Defines the maximum capacity of the bucket (e.g., 10 packets).
   * LEAK\_RATE: Defines the constant leak rate of the bucket (e.g., 1 packet per second).
   * bucket\_size: Tracks the current size of the bucket.
2. **Functions**:
   * **simulateIncomingData()**:
     + Simulates the arrival of data at random intervals.
     + The data packet arrival is between 1 and 3 packets.
     + If the bucket is not full, data is added. If the bucket overflows, excess data is discarded.
   * **leakData()**:
     + Leaks data from the bucket at a constant rate (LEAK\_RATE).
     + Data is removed every second at the defined leak rate.
3. **Main Function**:
   * Simulates the leaky bucket process by calling simulateIncomingData() and leakData() every second using sleep(1).

**Compilation and Execution:**

**Compilation:**

bash

Copy code

gcc -o leaky\_bucket leaky\_bucket.c

**Execution:**

bash

Copy code

./leaky\_bucket

**Sample Output:**

plaintext

Copy code

Leaky Bucket Congestion Control Simulation

Max bucket size: 10 packets

Leak rate: 1 packets per second

Incoming data: 2 packets

Data added to the bucket. Current bucket size: 2 packets

Data leaked at rate 1. Current bucket size: 1 packets

Incoming data: 3 packets

Data added to the bucket. Current bucket size: 4 packets

Data leaked at rate 1. Current bucket size: 3 packets

Incoming data: 1 packets

Data added to the bucket. Current bucket size: 4 packets

Data leaked at rate 1. Current bucket size: 3 packets

Incoming data: 2 packets

Bucket overflow! 1 packets discarded.

Current bucket size: 5 packets

Data leaked at rate 1. Current bucket size: 4 packets

**Key Points:**

1. **Bucket Overflow**: If the incoming data exceeds the bucket size, excess data is discarded.
2. **Constant Leak Rate**: Data leaks out of the bucket at a constant rate, ensuring smooth and regulated data flow.
3. **Congestion Control**: This algorithm helps prevent congestion by limiting the rate at which data enters the network, ensuring that the system doesn't become overloaded.

**Applications:**

* **Leaky Bucket Algorithm** is used in network traffic management to avoid sudden bursts of traffic, ensuring a smooth and steady data flow.
* **Rate Limiting**: The algorithm can be used in scenarios where a fixed rate of data transmission is required, such as in real-time streaming, communication protocols, etc.