**12. WAP to Implement Congestion control protocols: Token Bucket**

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

1. **Define the token rate**: The rate at which tokens are generated (tokens per second).
2. **Define the bucket size**: The maximum number of tokens the bucket can hold.
3. **Generate tokens**: Tokens are added to the bucket at a constant rate.
4. **Data Transmission**: Data can only be sent if there are sufficient tokens in the bucket.
5. **Overflow**: If the bucket reaches its maximum size, new tokens are discarded.

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

This program simulates the token bucket algorithm, where tokens are generated at a fixed rate, and data is transmitted based on the availability of tokens.

c

Copy code

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define MAX\_BUCKET\_SIZE 10 // Maximum number of tokens in the bucket

#define TOKEN\_RATE 1 // Number of tokens generated per second

#define DATA\_SIZE 1 // Data size in packets (fixed size for simplicity)

int bucket\_tokens = 0; // Current number of tokens in the bucket

int max\_bucket\_size = MAX\_BUCKET\_SIZE; // Maximum capacity of the token bucket

int token\_rate = TOKEN\_RATE; // Tokens generated per second

// Function to simulate the generation of tokens

void generateTokens() {

if (bucket\_tokens < max\_bucket\_size) {

bucket\_tokens += token\_rate; // Generate tokens at a fixed rate

if (bucket\_tokens > max\_bucket\_size) {

bucket\_tokens = max\_bucket\_size; // Bucket cannot hold more than max size

}

printf("Tokens generated: %d. Current tokens in the bucket: %d\n", token\_rate, bucket\_tokens);

} else {

printf("Bucket is full. No new tokens generated.\n");

}

}

// Function to simulate data transmission

void transmitData() {

if (bucket\_tokens >= DATA\_SIZE) {

bucket\_tokens -= DATA\_SIZE; // Consume tokens for data transmission

printf("Data transmitted: 1 packet. Remaining tokens in the bucket: %d\n", bucket\_tokens);

} else {

printf("Insufficient tokens. Data transmission paused.\n");

}

}

int main() {

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

printf("Max bucket size: %d tokens\n", max\_bucket\_size);

printf("Token rate: %d tokens per second\n", token\_rate);

while (1) {

// Simulate token generation every second

generateTokens();

// Simulate data transmission if tokens are available

transmitData();

// Wait for 1 second to simulate time intervals

sleep(1);

}

return 0;

}

**Explanation of the Program:**

1. **Variables**:
   * MAX\_BUCKET\_SIZE: Defines the maximum number of tokens the bucket can hold (e.g., 10 tokens).
   * TOKEN\_RATE: Defines the number of tokens generated per second (e.g., 1 token per second).
   * bucket\_tokens: Tracks the current number of tokens in the bucket.
2. **Functions**:
   * **generateTokens()**: Simulates the generation of tokens at the specified rate.
     + If the bucket is not full, new tokens are added to the bucket.
     + If the bucket is full, new tokens are discarded.
   * **transmitData()**: Simulates the data transmission process.
     + If there are enough tokens, a data packet is transmitted, and tokens are consumed.
     + If there are not enough tokens, data transmission is paused.
3. **Main Function**:
   * The main function simulates the token bucket process by calling generateTokens() and transmitData() every second using sleep(1).

**Compilation and Execution:**

**Compilation:**

bash

Copy code

gcc -o token\_bucket token\_bucket.c

**Execution:**

bash

Copy code

./token\_bucket

**Sample Output:**

plaintext

Copy code

Token Bucket Congestion Control Simulation

Max bucket size: 10 tokens

Token rate: 1 tokens per second

Tokens generated: 1. Current tokens in the bucket: 1

Data transmitted: 1 packet. Remaining tokens in the bucket: 0

Tokens generated: 1. Current tokens in the bucket: 1

Data transmitted: 1 packet. Remaining tokens in the bucket: 0

Tokens generated: 1. Current tokens in the bucket: 1

Data transmitted: 1 packet. Remaining tokens in the bucket: 0

Tokens generated: 1. Current tokens in the bucket: 1

Data transmitted: 1 packet. Remaining tokens in the bucket: 0

...

**Key Points:**

1. **Token Generation**: Tokens are generated at a constant rate and added to the bucket.
2. **Token Consumption**: Each data packet requires a token to be transmitted. If no tokens are available, data transmission is paused until more tokens are generated.
3. **Overflow**: If the bucket is full, new tokens are discarded. This prevents the system from being overwhelmed by excessive data.
4. **Regulated Data Flow**: The token bucket algorithm ensures that data is transmitted in a controlled manner, preventing sudden bursts that could lead to congestion.

**Applications:**

* **Traffic Shaping**: The token bucket algorithm is widely used for traffic shaping in networks to ensure smooth data flow.
* **Rate Limiting**: It can be used to implement rate limiting, ensuring that data is transmitted at a fixed or controlled rate.
* **Congestion Control**: It helps in managing congestion in network systems by controlling the rate of data transmission.