Name: Vedant Bhutada

Roll: 69

Batch: A4

Practical:9

Aim: Implement Congestion control mechanism - Leaky bucket and Tocken bucket

LEAKY BUCKET ALGORITHM

```
# initial packets in the bucket
storage = 0

# total no. of times bucket content is checked
no_of_queries = 4

# total no. of packets that can
# be accommodated in the bucket
bucket_size = 10

# no. of packets that enters the bucket at a time
input_pkt_size = 4

# no. of packets that exits the bucket at a time
output_pkt_size = 1
for i in range(0, no_of_queries): # space left

  size_left = bucket_size - storage
  if input_pkt_size <= size_left:
  # update storage
    storage += input_pkt_size
  else:
    print("Packet loss = ", input_pkt_size)

  print(f"Buffer size= {storage} out of bucket size = {bucket_size}")

  # as packets are sent out into the network, the size of the storage decreases
  storage -= output_pkt_size
```

```
Buffer size= 4 out of bucket size = 10
Buffer size= 7 out of bucket size = 10
Buffer size= 10 out of bucket size = 10
Packet loss =  4
Buffer size= 9 out of bucket size = 10
```

TOKEN BUCKET ALGORITHM

```python
import time

class TokenBucket:
    def __init__(self, capacity, rate):
        self.capacity = capacity  # Bucket capacity in bytes
        self.tokens = capacity    # Current number of tokens in the bucket
        self.rate = rate          # Token arrival rate in bytes/second
        self.last_refill_time = time.time()

    def _refill_bucket(self):
        current_time = time.time()
        time_passed = current_time - self.last_refill_time
        tokens_to_add = time_passed * self.rate

        self.tokens = min(self.capacity, self.tokens + tokens_to_add)
        self.last_refill_time = current_time

    def send_packet(self, packet_size):
        self._refill_bucket()

        if packet_size <= self.tokens:
            self.tokens -= packet_size
            return True  # Packet sent successfully
        else:
            return False  # Insufficient tokens, packet not sent

token_bucket = TokenBucket(capacity=1000, rate=10)  # 1000 bytes capacity, 10 bytes/second rate

for i in range(15):
    packet_size = 80
    if token_bucket.send_packet(packet_size):
        print(f"Packet {i + 1} sent successfully.")
    else:
        print(f"Packet {i + 1} discarded due to insufficient tokens.")
    time.sleep(0.5)  # Simulating time between packet transmissions
```

```
Packet 1 sent successfully.
Added a token. Total: 2
Added a token. Total: 3
Packet 2 sent successfully.
Added a token. Total: 4
Packet 3 sent successfully.
Added a token. Total: 5
Added a token. Total: 6
Packet 4 sent successfully.
Added a token. Total: 7
Added a token. Total: 8
Packet of size 5 arrived
Forwarding packet
Packet 5 sent successfully.
Added a token. Total: 4
Packet 6 sent successfully.
Added a token. Total: 5
Added a token. Total: 6
Packet 7 sent successfully.
Added a token. Total: 7
Added a token. Total: 8
Packet 8 sent successfully.
Added a token. Total: 9
Packet 9 sent successfully.
Added a token. Total: 10
Packet 10 sent successfully.
Packet 11 sent successfully.
Packet of size 7 arrived
Forwarding packet
Packet 12 sent successfully.
Added a token. Total: 4
Added a token. Total: 5
Packet 13 sent successfully.
Added a token. Total: 6
Added a token. Total: 7
Packet 14 discarded due to insufficient tokens.
Packet of size 6 arrived
```