Congestion Control in Data Networks

Congestion

A state occurring in network layer when the message traffic is so heavy that it slows down network response time.

Effects of Congestion

- As delay increases, performance decreases.
- If delay increases, retransmission occurs, making situation worse.

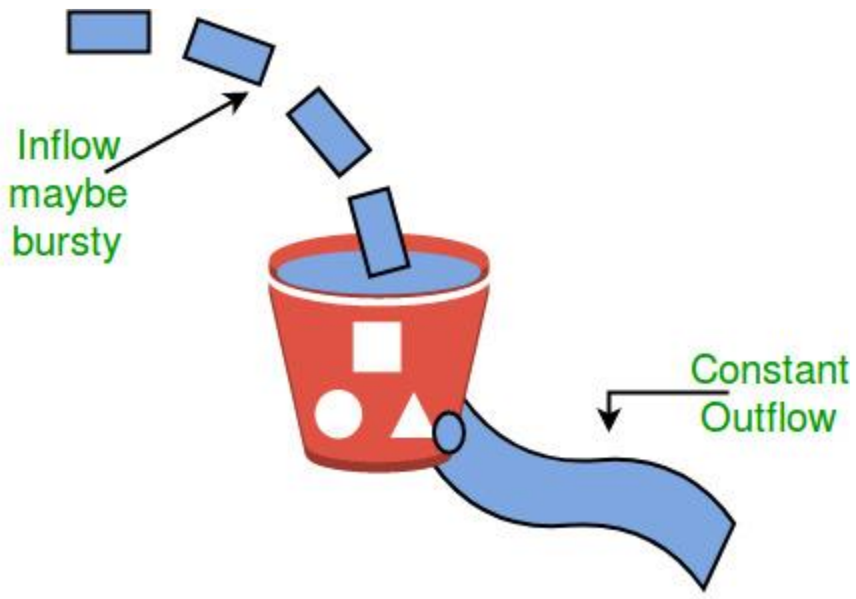
Congestion control algorithms

- Congestion Control is a mechanism that controls the entry of data packets into the network, enabling a better use of a shared network infrastructure and avoiding congestive collapse.
- Congestive-Avoidance Algorithms (CAA) are implemented at the TCP layer as the mechanism to avoid congestive collapse in a network.
- There are two congestion control algorithm which are as follows:

- **Leaky Bucket Algorithm**
- The leaky bucket algorithm discovers its use in the context of network traffic shaping or rate-limiting.
- A leaky bucket execution and a token bucket execution are predominantly used for traffic shaping algorithms.
- This algorithm is used to control the rate at which traffic is sent to the network and shape the burst traffic to a steady traffic stream.
- The disadvantages compared with the leaky-bucket algorithm are the inefficient use of available network resources.
- The large area of network resources such as bandwidth is not being used effectively.

Let us consider an example to understand

Imagine a bucket with a small hole in the bottom.No matter at what rate water enters the bucket, the outflow is at constant rate.When the bucket is full with water additional water entering spills over the sides and is lost.



Similarly, each network interface contains a leaky bucket and the following **steps** are involved in leaky bucket algorithm:

1. When host wants to send packet, packet is thrown into the bucket.
2. The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
3. Bursty traffic is converted to a uniform traffic by the leaky bucket.
4. In practice the bucket is a finite queue that outputs at a finite rate.

- **Token bucket Algorithm**
- The leaky bucket algorithm has a rigid output design at an average rate independent of the bursty traffic.
- In some applications, when large bursts arrive, the output is allowed to speed up. This calls for a more flexible algorithm, preferably one that never loses information. Therefore, a token bucket algorithm finds its uses in network traffic shaping or rate-limiting.
- It is a control algorithm that indicates when traffic should be sent. This order comes based on the display of tokens in the bucket.
- The bucket contains tokens. Each of the tokens defines a packet of predetermined size. Tokens in the bucket are deleted for the ability to share a packet.
- When tokens are shown, a flow to transmit traffic appears in the display of tokens.
- No token means no flow sends its packets. Hence, a flow transfers traffic up to its peak burst rate in good tokens in the bucket.

Need of token bucket Algorithm:-

The leaky bucket algorithm enforces output pattern at the average rate, no matter how bursty the traffic is. So in order to deal with the bursty traffic we need a flexible algorithm so that the data is not lost. One such algorithm is token bucket algorithm.

Steps of this algorithm can be described as follows:

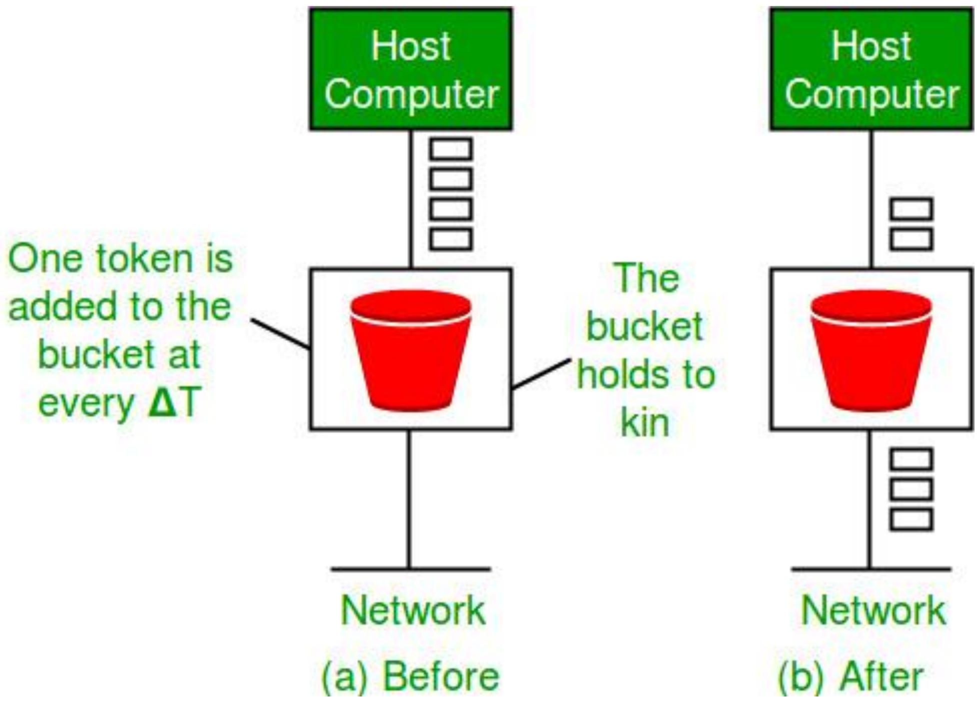
1. In regular intervals tokens are thrown into the bucket. ƒ
2. The bucket has a maximum capacity. ƒ
3. If there is a ready packet, a token is removed from the bucket, and the packet is sent.
4. If there is no token in the bucket, the packet cannot be sent.

Let's understand with an example,

In figure (A) we see a bucket holding three tokens, with five packets waiting to be transmitted. For a packet to be transmitted, it must capture and destroy one token. In figure (B) We see that three of the five packets have gotten through, but the other two are stuck waiting for more tokens to be generated.

Ways in which token bucket is superior to leaky bucket: The leaky bucket algorithm controls the rate at which the packets are introduced in the network, but it is very conservative in nature. Some flexibility is introduced in the token bucket algorithm. In the token bucket, algorithm tokens are generated at each tick (up to a certain limit). For an incoming packet to be transmitted, it must capture a token and the transmission takes place at the same rate. Hence some of the busty packets are transmitted at the same rate if tokens are available and thus introduces some amount of flexibility in the system.

Formula: $M * s = C + \rho * s$ where S – is time taken M – Maximum output rate ρ – Token arrival rate C – Capacity of the token bucket in byte
Let's understand with an example,


Host Computer
One token is added to the bucket at every ΔT
The bucket holds to kin
Network
(a) Before
Host Computer
Network
(b) After