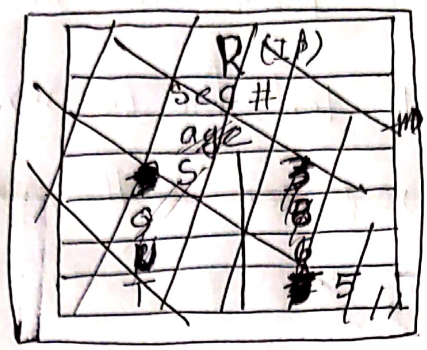
1|

The link-state packet for R is given below



R (IP)	
seq #	
age	
S	3
9	8
U	6
T	5

Lets consider read error has occured at sequence 9. In often every 10 sequence the next sequence ~~occure~~ appear.

10 sec → seq # 1 ✓
10 sec → seq # 2 ✓
⋮ ⋮
10 sec → seq # 9 → here read error occured. it read 999999 instade of 9.

now this will not take value biggen then 999999. Then it will be a huge problem. Every packet will be lost ~~under~~ 9 to 999999.

To solve this problem we insert age.

~~This determines the time to live of a packet.~~

If we set the age value as 60 ms.

Then after 60 ms the a sequence will be vanished means the packet will be vanished.

So sequence 999999 will last for 60 ms. ~~after that the next sequence will come.~~

seq # 999999

seq # 10 ↑ X

seq # 11 ↑ X

seq # 12 ↑ X

seq # 15 ↑ X

seq # 16 ↑

now it will accep seq 16. If we set the value of age as 6 ms then after 6 ms the data will vanish. Then the next data which will come after 10 ms will face ~~have~~ no data to compare with. So we have to set the age value carefully.

This is how dunation of age shont on long, in the link state packets can play a vital role at the time of sequence number ennon.

2017-2-60-096
Ruashed Md. Banket-E-Khuda

4|

here,

present threshold is 500 KB

present congestion window is 64 KB

Time out point is 502 KB

so the next consecutive congestion windows are;

64 x 2 = 128 KB → 1st congestion

128 x 2 = 256 KB → 2nd congestion

256 x 2 = 512 KB

But the threshold is 500 KB

So the 3rd congestion window is 500 KB

so,

4th congestion window = 501 KB

5th congestion window = 502 KB

here the time out point is 502 KB. So we need to set a new threshold. so

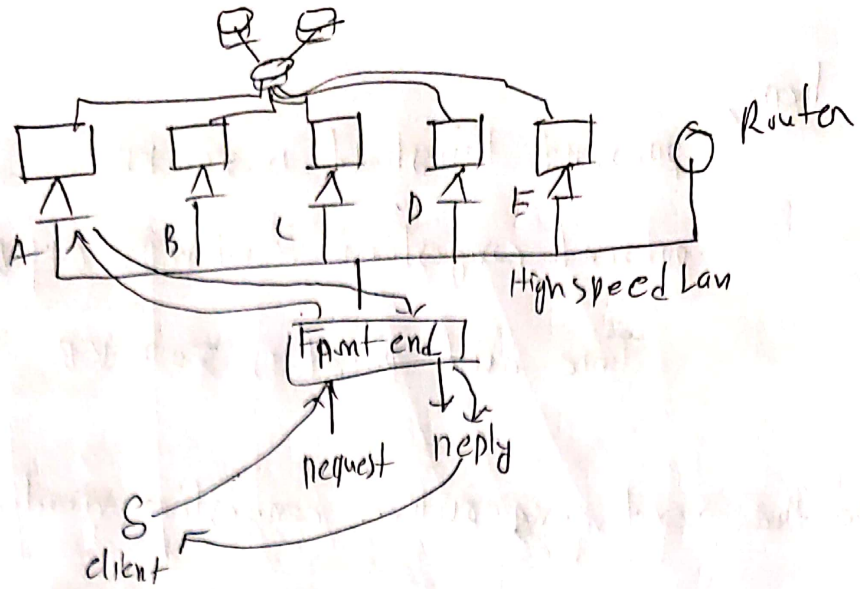
$502 \times \frac{1}{2} = 251$ KB

6th congestion is 1 KB

7th congestion is 2 KB

8th congestion is 2 x 2 = 4 KB

5|



Every processing node has its own cache memory. For having own cache memory it can hold on save more data. when a client request for a specific information it goes through Front end to the server farm. Lets say a client ~~wants~~ want to visit CNN.com. So he searched for it. This information request goes to node A. If the information is ~~on~~ in the cache of node A it will reply fasten. But if hay not in the cache the node A will take it from the HDD. Then it will save

it to its cache. Next time any one search for CMH.com node A can reply fastenly to the client. This will true for all the nodes at the server farm.

They will also act as the same type of request will handed over the same machine.

For having own cache mamony they will reply fasten to the client.

If there are many request from the clients or their are mang more client we can use tcp hand off. then the server will directly reply to the client.

3|

Here scenario 2 has low jitter. Because the diffrance of delay between packets are low here than scenario 1. so scenario 2 has low jitter.

Low jitter is necessary for multimedia communication. In multimedia communication both sides interaction happens. So here we need a very smooth flow of data. If the variation is high means the jitter is high then there will be synchronization problem of listening and seeing at the same time. If the variation is low means the jitter is Low then we can easily communicate and able to understand what the other person is saying from the other side.

So Low jitter is important in multimedia communication.

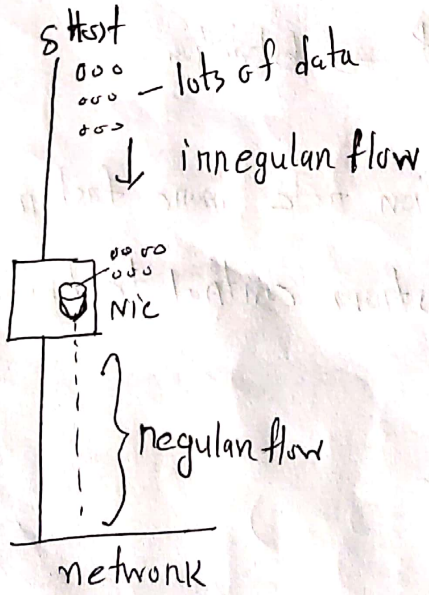
2017-2-60-096
Ruashed Md. Barkat-E-Khuda

2]

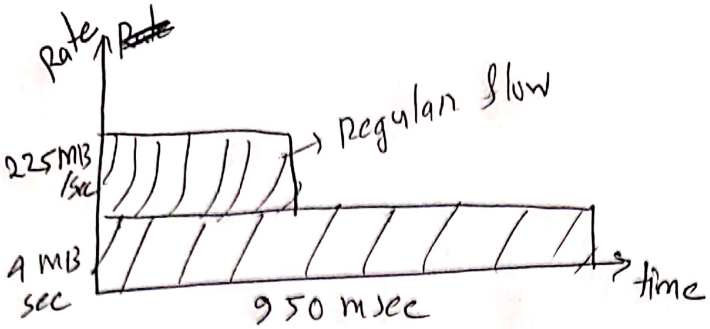
The punpose of leaky bucket algonithm is to have a negulan flow of data.

To achive good quality of service we can use leaky bucket algonithm, of traffic shaping technique.

when lots of data entens a network with innegulan flow this Leaky bucket algonithm can be implemented. Then a negulan flow can be achieved. When the bucket is full the data will be in the queue. ~~And~~ The bucket will act as buffen. Data as needed will flow from the Leak to the network.



2017-2-60-096



here,

input = 225 MB/sec with 950 msec.

network data rate 4 MB/sec.

so,

$$\frac{255\ MB}{sec} \times 950\ msec$$

$$= \frac{\cancel{255}\ 45\ MB}{\cancel{1000}\ \cancel{sec}\ msec} \times \overset{19}{\cancel{950}}\ msec$$

$$\cancel{20}\ 4$$

$$= 213.75\ MB$$

$$time = \frac{total\ data}{rate} = \frac{213.75\ MB \times sec}{4\ MB}$$

$$= \frac{213.75\ MB \times 1000\ msec}{4\ MB}$$

$$= 53.4375\ msec$$

Ans.