HOCHSCHULE
SRH HEIDELBERG

Simulation & Modeling of Token Bucket Traffic Shaper

Gundala Dharaneesh (11015580)

11015580@stud.hochschule-heidelberg.de

Chamala Sai Nikhil Reddy (11015237)

11015237@stud.hochschule-heidelberg.de

Under the guidance of

Prof. Dr. Zarrar Yousaf

CODE

NED FILE:

```
simple StationA
{
    parameters:
        @display("i=block/routing");
    gates:
        output out;
}

simple TrafficShaper
{
    parameters:
        @display("i=block/routing");
    gates:
        input in;
        output out;
}

simple StationB
{
    parameters:
        @display("i=block/process");
    gates:
        input in;
}

network Tictoc8
{
    @display("bgb=699,402");
    submodules:
        stationA: StationA {
            parameters:
                @display("i=,cyan;p=30,209");
        }
        trafficShaper: TrafficShaper {
            parameters:
                @display("i=,white;p=300,209");
        }
        stationB: StationB {
            parameters:
                @display("i=,gold;p=601,209");
        }
    connections:
        stationA.out --> {  delay = 100ms; } --> trafficShaper.in;
        stationB.in <-- {  delay = 100ms; } <-- trafficShaper.out;
}
```

SOURCE FILE:

```
#include <stdio.h>
#include <string.h>
#include <omnetpp.h>
#include <queue>

using namespace omnetpp;

//From Station A to traffic shaper
#define VBR ((rand() % 6) + 1)
#define BURST_PACKET_SIZE ((rand() % 5) + 1)

//From traffic shaper to Station B
#define CBR 3
#define TOKEN_RATE 4
#define MAX_QUEUE_SIZE 6
#define DATA_MESSAGE "DATA"
#define INTERVAL_MESSAGE "INTERVAL"
#define START_LETTER 'a'


class StationA : public cSimpleModule
{
    private:
        cMessage *message, *interval;
        simtime_t delay = 1;
        char counter = START_LETTER;
    protected:
        virtual void initialize() override;
        virtual void handleMessage(cMessage *msg) override;
};


void StationA::initialize()
{
    interval = new cMessage(INTERVAL_MESSAGE, BURST_PACKET_SIZE);
    scheduleAt(simTime(), interval);
}
void StationA::handleMessage(cMessage *msg)
{

    int burst_size = msg->getKind();

    for(int i = 0; i < burst_size; i++){

        message = new cMessage(DATA_MESSAGE, counter);

        send(message, "out");

        EV <<" A: " << counter << " at time " << simTime() << endl;
        counter++;

        if(counter >= (START_LETTER + 26))

            counter = START_LETTER;
    }
    int burst_rate = BURST_PACKET_SIZE;

    interval = new cMessage(INTERVAL_MESSAGE, burst_rate);

    simtime_t next_burst = simTime() + VBR;

    EV << "Next burst scheduled at " << next_burst << " of packet size
" << burst_rate;

    scheduleAt(next_burst, interval);
}

Define_Module(StationA);

class TrafficShaper : public cSimpleModule
{
    private:
        std::queue<char> Queue;

        cMessage *message, *interval;

    protected:
        virtual void initialize() override;

        virtual void handleMessage(cMessage *msg) override;
};
```

```
void TrafficShaper::initialize()
{
    interval = new cMessage(INTERVAL_MESSAGE, 0);
    scheduleAt(simTime() + CBR, interval);
}

void TrafficShaper::handleMessage(cMessage *msg)
{
    if(!strcmp(msg->getName(), INTERVAL_MESSAGE)){

        if(Queue.size() >= TOKEN_RATE){

            char temp;

            do{

                for(int i = 0; i < TOKEN_RATE ; i++){

                    temp = Queue.front();

                    Queue.pop();

                    message = new cMessage(DATA_MESSAGE, temp);

                    send(message, "out");

                    EV << "Traffic Shaper sent " << temp << " at time
" << simTime() << endl;

                }

            } while(Queue.size() >= TOKEN_RATE);
        }

        scheduleAt(simTime() + CBR, interval);

    }else{

        // Queue incoming message from Station A
        char data_message = msg->getKind();

        if(Queue.size() < MAX_QUEUE_SIZE){
            Queue.push(data_message);

        }else{
            EV << "Traffic Shaper dropping packet " << data_message <<
" as queue is full!";
        }
    }
}

Define_Module(TrafficShaper);

class StationB : public cSimpleModule
{
    private:
        cMessage *event;

    protected:
        //virtual void initialize() override;

        virtual void handleMessage(cMessage *msg) override;
};

void StationB::handleMessage(cMessage *msg)
{
    if(!strcmp(msg->getName(), DATA_MESSAGE)){

        char temp = msg->getKind();

        EV <<" B: Received " << temp << " at time " << simTime();
    }
    else
        EV << " B: Empty message received at time " << simTime();
}

Define_Module(StationB);
```

<u>**INITIALIZE FILE:**</u>

```
[General]
network =Tictoc8
record-eventlog = true
```