Using Fuzzy Logic Control to Provide Intelligent Traffic Management Service for High-Speed Networks

[1]A Murali Mohan Kumar, [2]R Prabhakar Naidu [3], R A Raja, [4]N K Dinakar, [5]H Syed Ali Fathima

[1,2]Associate Professor, [3,4,5]Assistant Professor
Mother Theresa Institute of Computer Applications,
Mother Theresa Institutions, Palamaner, Chittoor Dt., Andhra Pradesh, India

ABSTRACT: **The new system implements distributed traffic management with intelligent data rate controllers to tackle the traffic mass framework, in which routers are deployed. Unlike other explicit traffic control protocols that have to estimate network parameters (e.g., link latency, bottleneck bandwidth, packet loss rate, or the number of flows) in order to compute the allowed senders sending rate, hence it avoids various potential performance problems arising from parameter estimations while reducing much consumption of computation and memory resources in routers, our fuzzy-logic-based controller can measure the router queue size directly. As a network parameter, the queue size can be accurately monitored and used to proactively decide if action should be taken to regulate the source sending rate, thus increasing the resilience of the network to traffic congestion. The quality standards of communication which in general measured as QoS (Quality of Service) is assured by the good performances of our scheme such as max-min fairness, low queuing delay and good robustness to network dynamics. Simulation results and comparisons have verified the effectiveness and showed that our new traffic management scheme can achieve better performances than the existing protocols that rely on the estimation of network parameters.**

***Keywords*: Congestion Control, Fuzzy logic control, QoS, Robustness, Traffic Management, QFCP**

Network Traffic Management

Network traffic management can prevent network from severe congestion and degradation in throughput delay Performance. TCP (Transmission Control Protocol) is a widely deployed congestion control protocol that tackles the Internet traffic. It has the important feature that the network is treated as a black box and the source adjusts its window size based on packet loss signal. The controllers reside in routers and directly feed link information back to sources so that the link bandwidth could be efficiently utilized with good scalability and stability in high BDP networks for which Protocols like XCP, RCP, JetMax and MaxNet are being used . XCP feeds back the required increment or decrement of the sending rate, while RCP directly signals sources with the admissible sending rate according to which sources pace their throughput. The advantages of these router-assisted protocols are as follows.

1. They can explicitly signal link traffic levels without maintaining per-flow state.
2. The sources can converge their sending rates to some social optimum and achieve a certain optimization objective.

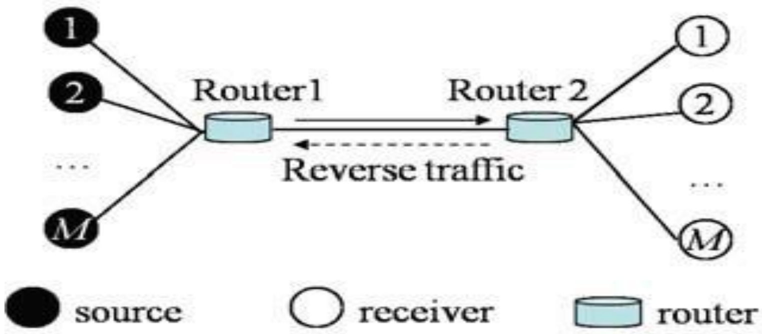
There are some latest protocols on wireless applications such as QFCP (Quick Flow Control Protocol) and the three protocols called Blind, Errors and MAC. They have improved on the estimation error while having high link utilization and fair throughput but they cannot keep the queue size stable due to oscillations, which in turn affect the stability of their sending rates.

Existing System

Current System exhibits that misestimating of linked bandwidth may easily occur and can cause significant fairness and stability problems (e.g., in link sharing networks or wireless networks). The improved on the estimation error while having high link utilization and fair throughput. The latest Protocols wireless applications such as QFCP (Quick Flow Control Protocol) and the three protocols called Blind, Errors and MAC. However, they cannot keep the queue size stable due to oscillations, which in turn affect the stability of their sending rates Also In addition, their bandwidth probing speed may be too slow when the bandwidth jumps a lot hence they still have the fundamental problem of inaccurate estimation resulting in performance degradation.

Proposed System

The new system implements distributed traffic management with intelligent data rate controllers to tackle the traffic mass framework, in which routers are deployed. Unlike other explicit traffic control protocols that have to estimate network parameters (e.g., link latency, bottleneck bandwidth, packet loss rate, or the number of flows) in order to compute the allowed senders sending rate, hence it avoids various potential performance problems arising from parameter estimations while reducing much consumption of computation and memory resources in routers, our fuzzy-logic-based controller can measure the router queue size directly. As a network parameter, the queue size can be accurately monitored and used to proactively decide if action should be taken to regulate the source sending rate, thus increasing the resilience of the network to traffic congestion..



"Fig.1" Architecture:

Procedure:

(1) the router extracts *Req_rate* from the congestion header of the packet. At the time arrival of a packet

(2) To get updated $e(t)$ and $g(e(t))$ it Sample IQSize $q(t)$

(3) Calculate the output $u(t)$ and compare it with *Req_rate*.

(a) the link does not have enough bandwidth to accommodate the requested amount of sending rate If $u(t) < Req\_rate$, it means that the *Req_rate* field in the congestion header is then updated by $u(t)$.

(b) Otherwise the *Req_rate* field remains unchanged.

(4) It update the crisp output $u(t)$ and the output edge value of D, If an operation cycle d is over,

Implementation

The existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods and implementation stage involves careful planning of the investigative methods. The theoretical design is turned out into a working system. It can be considered to be the most challenging stage in achieving a successful new system and to enhance the user, confidence that the new system will work and be effectively implemented.

Problem Statement:

The deterioration in quality of service occurs at times when network congestion increases in data networking or when link node is carrying more data, which occurs Typical effects include queuing delay, packet loss or the blocking of new connections. load lead either only to small increase in network throughput, or to an actual reduction in network throughput occurs when lead load incrementally increases.

Scope:

1) To work with high speed IP networks explicitly based on rate-based management scheme fuzzy logic theory is designed

2) the application of such a fuzzy logic controller using less performance parameters while providing better performances than the existing explicit traffic control protocols

3) the design of a Fuzzy Smoother mechanism that can generate relatively smooth flow throughput

4) the capability of our algorithm to provide max-min fairness even under large network dynamics that usually render many existing controllers unstable.

Project Implementation:-

Fuzzy Logic Control:

It has been considered for IC (Intelligence Control). In addition, fuzzy logic theory provides a convenient controller design approach based on expert knowledge which is close to human decision making, and readily helps engineers to model a complicated non-linear system. It is a methodology used to design robust systems that can contend with the common adverse synthesizing factors such as system nonlinearity, parameter uncertainty, measurement and modeling imprecision. In fact, the mature control performance in accuracy, transient response, robustness and stability of fuzzy logic control have made this logic to apply widely in industrial process control and showed extraordinary performance.

FLC has found its applications to network congestion control since 1990. In early stage, it was used to do rate control in ATM network, e.g., [40], [41], to guarantee the QoS. FLC was used in RED (Random Early Detection) algorithm in TCP/IP networks to reduce packet loss rate and improve utilization. However, they are still providing implicit or imprecise congestion signaling, and therefore cannot overcome the throughput fluctuations and conservative behavior of TCP sources.

Traffic management:

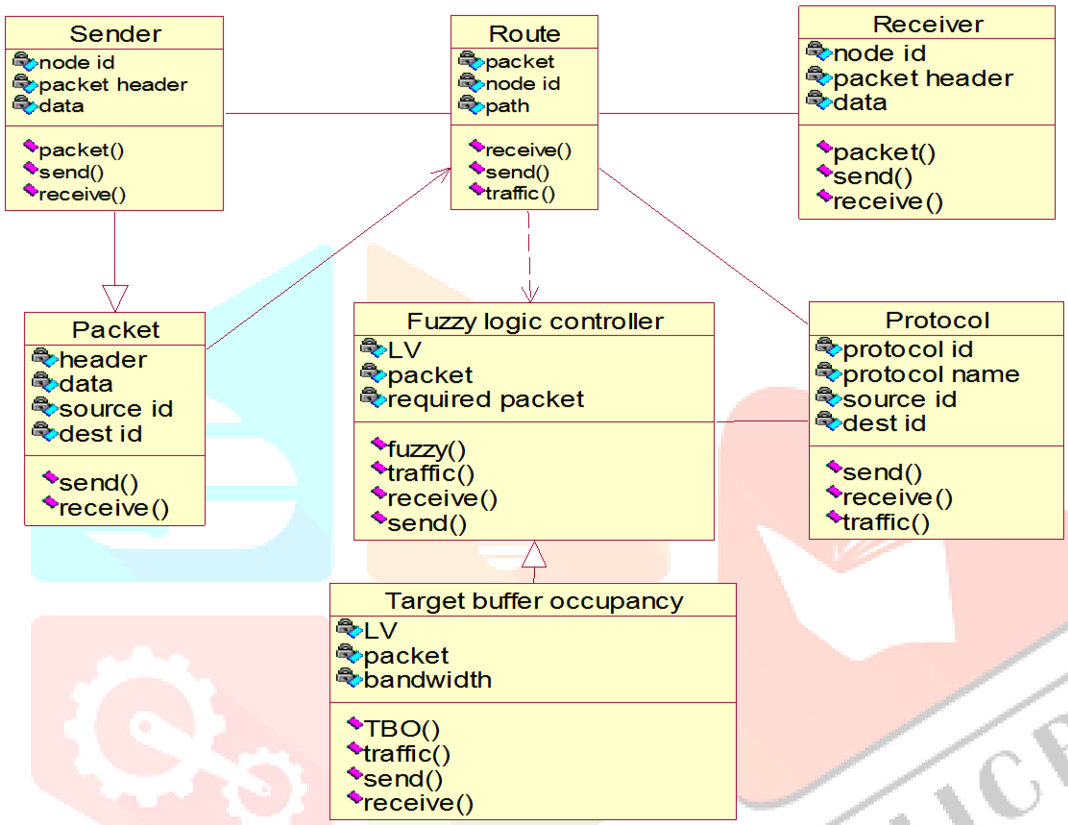
TCP (Transmission Control Protocol) Reno, is a widely deployed congestion control protocol that tackles the Internet traffic. Through throughput delay performance Traffic management can prevent a network from severe congestion and degradation process. It has the important feature that the network is treated as a black box and the source adjusts its window size based on packet loss signal. However, when the Internet BDP (Bandwidth-Delay Product) continues to increase TCP encounters various performance problems (e.g., utilization, fairness and stability)

Bottleneck link

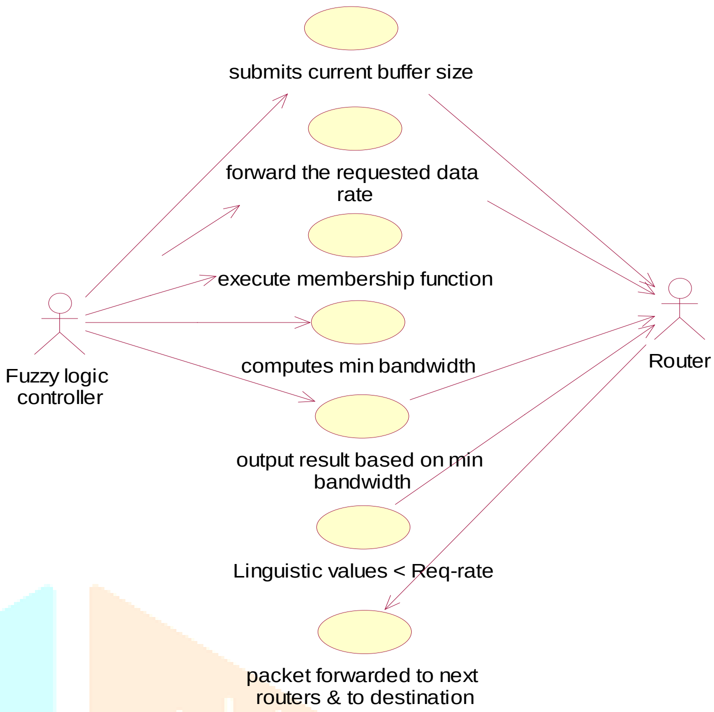
Also note that this definition does not forbid a single link to be a bottleneck for multiple flows. this definition is substantially different from a common meaning of a *bottleneck*. The given data flow achieves maximum data rate network-wide when A *bottleneck link* for a given data flow is a link that is fully utilized (is *saturated*) and of all the flows sharing this link. Link (or bottleneck) utilization is the ratio between the current actual throughput in the bottleneck and the maximum data rate of the bottleneck. A data rate allocation is max-min fair if and only if a data flow between any two nodes has at least one bottleneck link.

Source throughput, IQSize, Queuing delay:

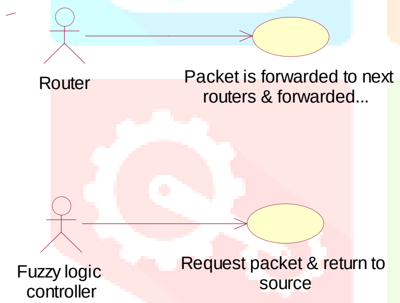
Source throughput is defined as number of bits successfully sent out by a source per second on average, i.e. bits/second. In this scenario a bit is considered as successful when respective bit is part of a packet that has been successfully out sent. Buffer queue (measured in packets) seen by a departing packet of the bottleneck is the length considered as IQSize. Queuing delay is the waiting time of a packet in the router queue before its service. Measurements are taken from the time the first bit of a packet is received at the queue until the time the first bit of the packet is transmitted.



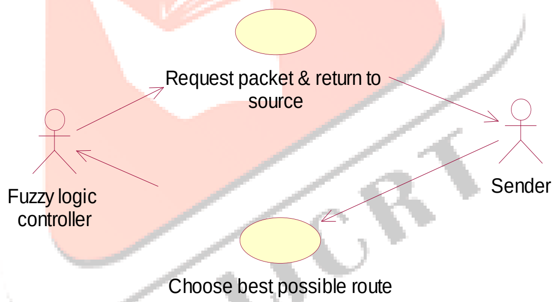
"Fig.2" Class Diagram



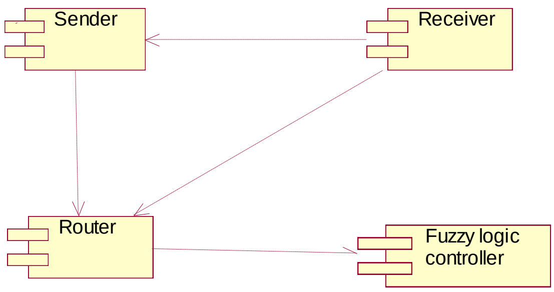
"Fig.2" Use case Diagram 1



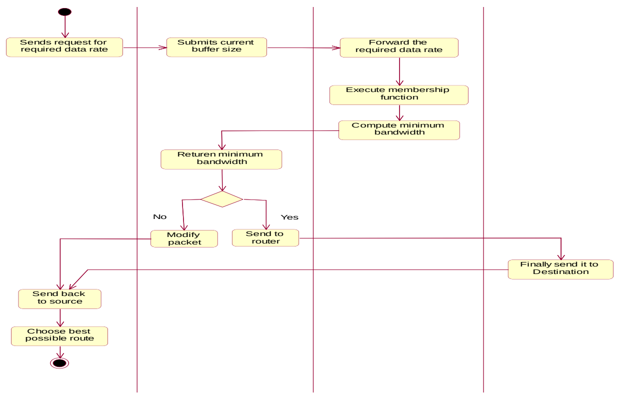
"Fig.3" Use case Diagram 2



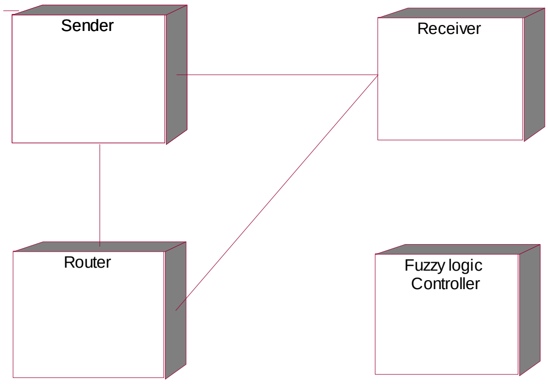
"Fig.4" Use case Diagram 3



"Fig.5" Component Diagram

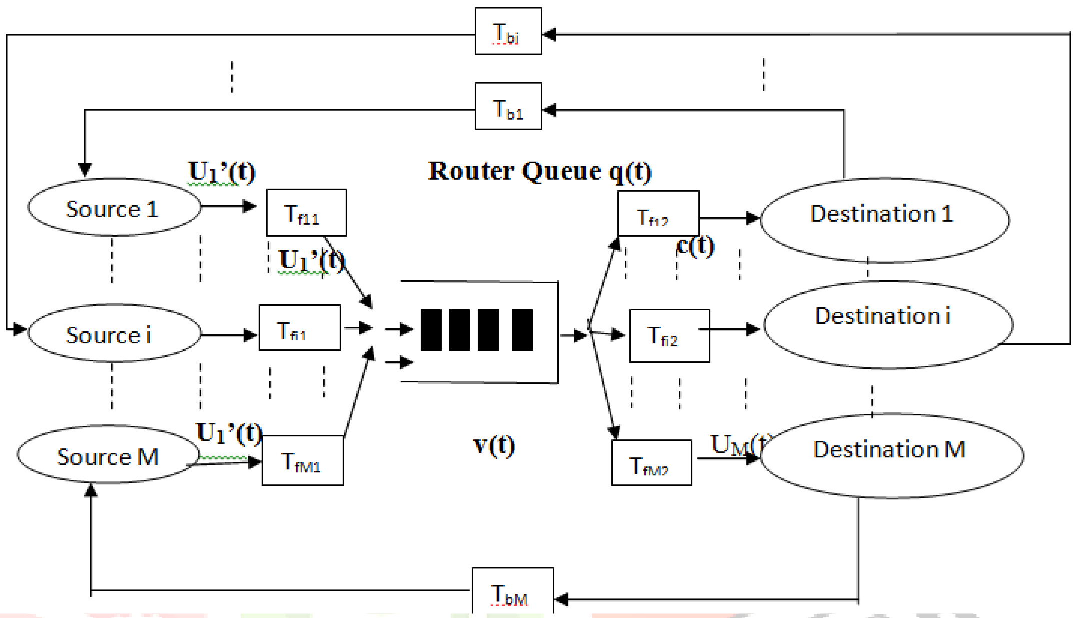


"Fig.6" Activity Diagram



"Fig.7" Deployment Diagram

SYSTEM CODING AND IMPLEMENTATION IMPLEMENTATION



"Fig 8" : SYSTEM MODEL

From the Figure 8. We have

c(t) Service rate (output link capacity) of a router
e(t) Queue error which is one input of the IntelRate controller
g(e(t)) Integration of e(t) which is the other input of the IntelRate controller
q0 TBO of a router
q (t) IQ Size (Instantaneous Queue Size) of a router
u(t) The controller crisp output for each flow
u'(t) Current source sending rate
v(t) Aggregate uncontrolled incoming traffic rate to a router
y(t) Aggregate controlled incoming traffic rate to a router (also the aggregate controller output)
Tfi1 Time delay of a packet from source I to a router
Tfi2 Time delay of a packet from a router to its destination i
Tbi Feedback delay of a packet from destination I back

The IntegRate Controller Design

The IntegRate is the components which work as our fuzzy logic traffic controller in the network system. it is a TISO (Two-Input Single-Output) controller. The TBO (Target Buffer Occupancy) q0>0 is the queue size level we aim to achieve upon congestion. the two inputs of the controller of queue deviation is e (t) = q0−q (t). In order to remove the steady state error, we choose the integration of e(t) as the other input of the controller i.e. g(e(t)) =e(t)dt The aggregate output is y(t) =ui(t − Ti). Under heavy traffic situations, the IntegRate controller would compute an allowed sending rate ui(t) for flow i according to the current IQSize so that q(t) can be stabilized around q0. In our design, IQ Size q(t) is the only parameter each router needs to measure in order to complete the closed-loop control. FLC is a non-linear mapping of inputs into outputs, which consists of four steps, i.e., rule base building, Fuzzyfication, inference and defuzzification.

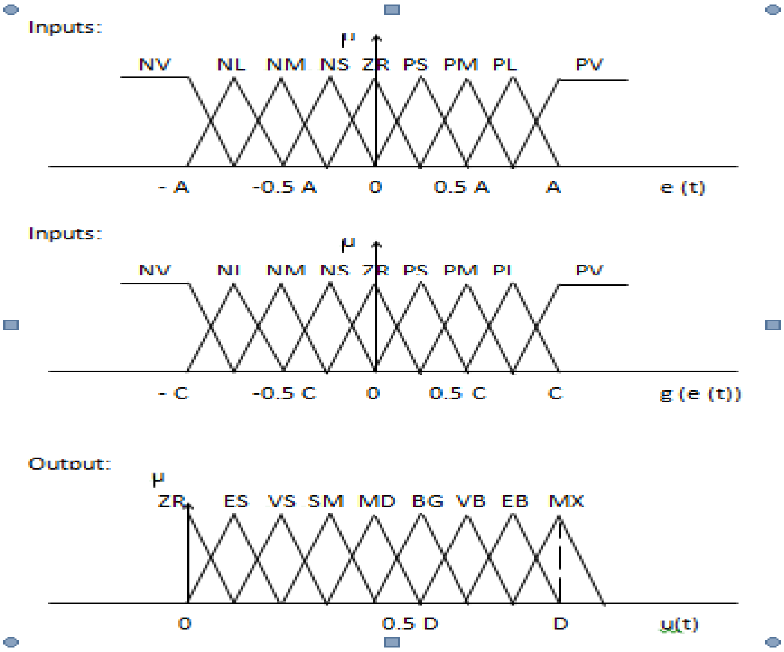
The theories of fuzzy set and logic of FLC were introduced in 1965 by Zadeh, and it was basically extended from two-valued logic to the continuous interval by adding the intermediate values between absolute TRUE and FALSE. Interested readers are referred to some standard tutorials/texts like for the details of the fuzzy logic theory. In the new controller by following those four steps along with designing the fuzzy linguistic descriptions and the membership functions. The parameter design issues and the traffic control procedure are also discussed at the end of the section.

Linguistic Description and Rule Base

We define the crisp inputs e(t), g(e(t)) and output u(t) with the linguistic variables e(t), g(e(t)) and u(t), respectively. There are N (N= 1,2,3,...) LVs (Linguistic Values) assigned to each of these linguistic variables. Specifically, let be the input LVs with i = 1 for e(t) and i = 2 for (e(t)), and let U =Uj, where j = 1,2,⋯ ,N for u(t). For example, when N = 9, the inputs e(t) and g(e(t)) these values are obtained as, P1i= Negative Very Large (NV), P2i= Negative Large (NL), P3i= Negative Medium (NM), P4i= Negative Small (NS), P5i=Zero (ZR), P6i= Positive Small (PS), P7i=Positive Medium (PM), P8i=Positive Large (PL) and P9i= Positive Very Large (PV), i = 1,2. Similarly, the output when N = 9 can form the linguistic values as follows.U1= Zero (ZR), U2= Extremely Small (ES), U3= Very Small (VS), U4= Small (SM), U5= Medium (MD), U6= Big (BG), U7= Very Big (VB), U8= Extremely Big (EB) and U9= Maximum (MX).

Membership Function, Fuzzyfication and Reference

Our IntelRate controller employs the isosceles triangular and trapezoid-like functions as its MFs (Membership Functions). The above Figure describes the MFs used to determine the certainty of a crisp input or output. We let P1 be the UoD1 for the input p1 = e(t), and P2 be the input p2= g(e(t)). The value of MFs (i.e., the certainty degree) for crisp inputs pi(i = 1,2) is designated by μPj. Similarly, we let Z be the output z = u(t). Thus the input and output fuzzy sets can be defined with Pj i=pi,μPji(pi) : pi∈ Pi, i = 1,2 and , Uj= {(z,μUj (z)) : z ∈ Z}, j = 1,2,...,N, respectively.



"Fig 9": Membership Functions without FS

The Control Procedure

The new field in the packet congestion header that we need to support our controller algorithm for the operation principle mentioned in Figure16. The congestion header a new field called Req_rate to carry the desired sending rate from the source and which will be continuously updated by the allowed sending rate when the packet passes each router.

the IntegRate controller in a router is as given below through the traffic-handling procedure is being handled.

(1) the router extracts *Req_rate* from the congestion header of the packet. At the time arrival of a packet

(2) To get updated *e*(*t*) and *g*(*e*(*t*)) it Sample IQSize *q*(*t*)

(3) Calculate the output *u*(*t*) and compare it with *Req_rate*.

(a) the link does not have enough bandwidth to accommodate the requested amount of sending rate If *u*(*t*) < *Req_rate*, it means that the *Req_rate* field in the congestion header is then updated by *u*(*t*).

(b) Otherwise the *Req_rate* field remains unchanged.

(4) It update the crisp output *u*(*t*) and the output edge value of *D*, If an operation cycle *d* is over,

This procedure allows the router to perform max-min fairness in that greedy flows are restricted to u(t) by router under heavy traffic conditions.

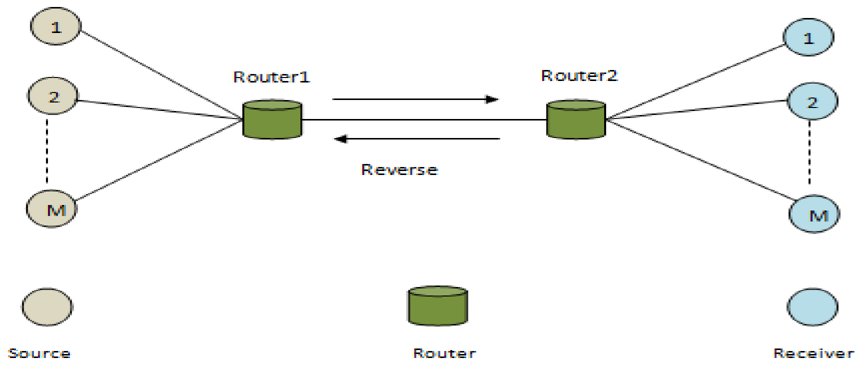


Figure 17: Simulation Network

SAMPLE CODE

Node.java

```
package TrafficControl;
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import java.util.*;
import java.awt.*;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.Socket;
import javax.swing.JEditorPane;
public class Node
{
public static int L,l;
public LV TBO=LV.MX;

private LocalDataStructure localDS=new LocalDataStructure();
private boolean covered=false;
private String ip;
private String port;
private ArrayList<Packet> buff=new ArrayList<Packet>();
private Vector neighbours=new Vector();
private String id;
private long len;
private Point location;
private Color color=Color.DARK_GRAY;
private Vector lengths=new Vector();
private GraphCanvas graphcanvas;
private Node pred;
private int init_traffic;
private float init_entropy;
private Node parentnode;
int capacity=500;
private String attackerstatus="N/A";
public void constructPClaim(Packet p,Node n,int pi,int ti,String source)
{
P_Claim pclaim=new P_Claim();
pclaim.setSource(source);
pclaim.setCp(pi);
pclaim.setT(2);
pclaim.setHm(p.hashCode());
pclaim.setSigS((pclaim.getHm()+pclaim.getSource()+pclaim.getCp()+pclaim.getT()).hashCode());
p.setPclaim(pclaim);
this.localDS.getPclaims().put(p.getPacketid(), pclaim);
T_Claim tclaim=new T_Claim();
```

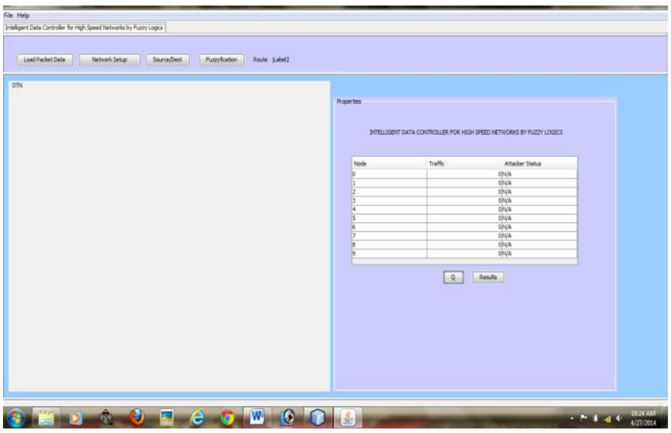
```
tclaim.setA(source);
tclaim.setCp(ti);

tclaim.setT(2);
tclaim.setHm(p.hashCode());
tclaim.setSigS((tclaim.getHm()+tclaim.getA()+tclaim.getCp()+tclaim.getT()).hashCode());
p.setTclaim(tclaim);
this.localDS.getTclaims().put(p.getPacketid(), tclaim);
}
public boolean InconsistancyCheck(Packet p,Node n)
{
boolean allowed=false;
P_Claim pclaim1=this.localDS.getPclaims().get(p.getPacketid());
P_Claim pclaim2=p.getPclaim();
T_Claim tclaim1=this.localDS.getTclaims().get(p.getPacketid());
T_Claim tclaim2=p.getTclaim();
if (pclaim1.getSigS() == pclaim2.getSigS()) {
System.out.println("Sig Verification:OK");
if (pclaim2.getCp() <= Node.L) {
System.out.println("P-claim Verification:OK");
if (tclaim2.getCp() <= Node.l) {
System.out.println("T-claim Verification:OK");
allowed = true;
}
Else
{
System.out.println("T-Claim Verification:Failed");
}
}
else
{
System.out.println("P-Claim Verification:Failed");
}
}

else
{
System.out.println("Sig Verification:Failed");
}
if(!allowed)
{
n.setAttackerstatus("Detected");
}
return allowed;
}
//membership
public LV requestTBO(Packet p)
{
int m=this.capasity/2;
int q0=this.getBuff().size();
float ut=m-q0;
LV TBO=LV.ZR;
if(ut==0)
{TBO=LV.ZR;}
else if(ut<(-m/1))
{TBO=LV.ES;}
else if(ut<(-m/2))
{TBO=LV.VS;}
else if(ut<(-m/3))
{TBO=LV.SM;}
else if(ut<(-m/4))
{TBO=LV.MD;}
else if(ut<(m/4))
```

```
{TBO=LV.BG;}
else if(ut<(m/3))
{TBO=LV.VB;}
else if(ut<(m/2))

{TBO=LV.EB;}
else if(ut<(m/1))
{TBO=LV.MX;}
this.TBO=TBO;
System.out.println("\nTBO at Node : "+this.getId()+" is "+TBO+" - Q0("+q0+")");
if(p.TBO.ordinal()>TBO.ordinal())
{ p.TBO=TBO;
}
/*
System.out.println(-m/1);
System.out.println(-m/2);
System.out.println(-m/3);
System.out.println(-m/4);
System.out.println(m/4);
System.out.println(m/3);
System.out.println(m/2);
System.out.println(m/1);
*/
return TBO;
}
public Node(Point loc,String id,Color color)
{
this.location=loc;
this.id=id;
this.color=color;
}
public Node()
{
}
Node(Point nodepoint, String id, GraphCanvas aThis) {
this.location=nodepoint;
this.id=id;

this.graphcanvas=aThis;
}
public void addToLengths(int length)
{
getLengths().addElement(new Integer(length));
}
public void addToNeighbourlist(Node n)
{
boolean alreadyadded=false;
for(int i=0;i<getNeighbours().size();i++)
{
Node c=(Node)getNeighbours().get(i);
if(c.getId().equals(n.getId()))
{
alreadyadded=true;
return;
}
}
getNeighbours().addElement(n);
}
public String getId()
{
return id;
}
public void setlen(long len)
```

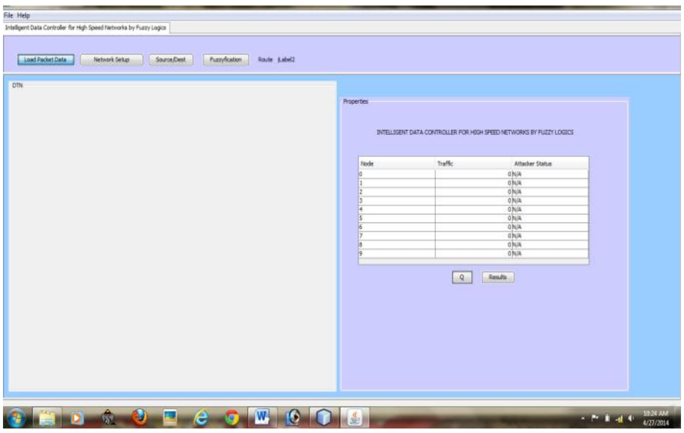
```
{
this.setLen(len);
}
public Point getLocation()
{
return location;
}

public long getlen()
{
return getLen();
}
public long getLen() {
return len;
}
/**
* @param len the len to set
*/
public void setLen(long len) {
this.len = len;
}
/**
* @param location the location to set
*/
public void setLocation(Point location) {
this.location = location;
}
public static int getDistance(Node a,Node b)
{
int x1=a.getLocation().x;
int y1=a.getLocation().y;
int x2=b.getLocation().x;
int y2=b.getLocation().y;
double dist=Math.sqrt(Math.pow(x2-x1,2)+Math.pow(y2-y1,2));
return ((int)dist-(26));
}
public void setNeighbours(Vector neighbours) {
this.neighbours = neighbours;
}
```

7. RESULTS

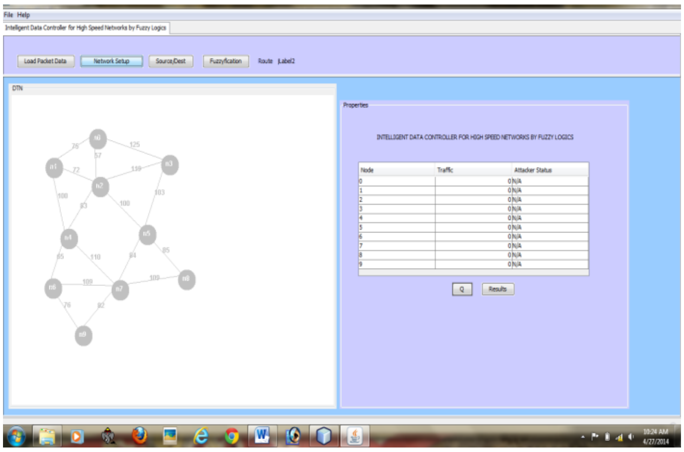
Step 1: Initially the output screen looks as follows



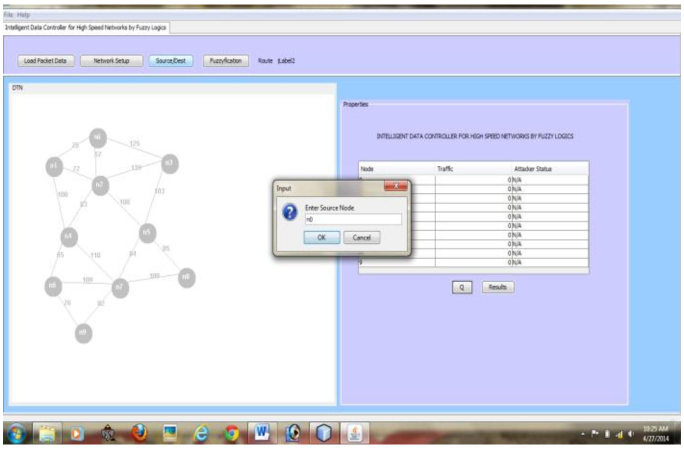
Step 2: Select the Load Packet Data



Step 3: Click on the Network setup to view the sample network

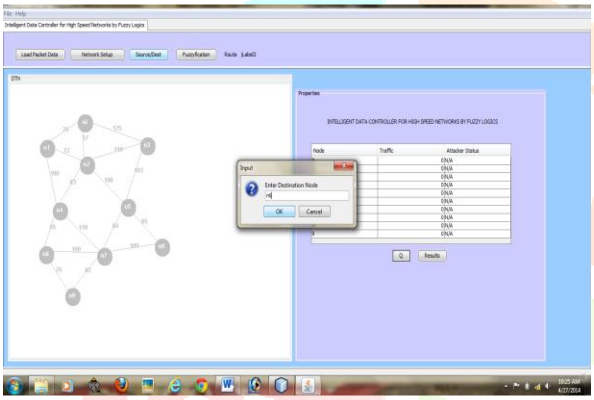


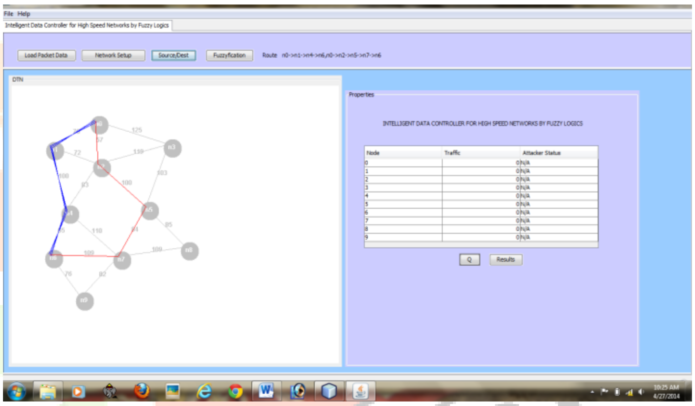
Step 4: Select the Source and Destination nodes. Initially declare the Source node.



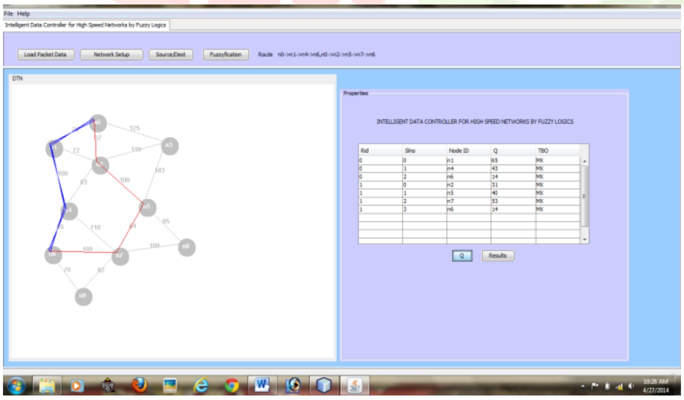
Then Declare the Destination node.

Step 5: Thus the two possible paths along with the respective queue sizes and TBO is displayed

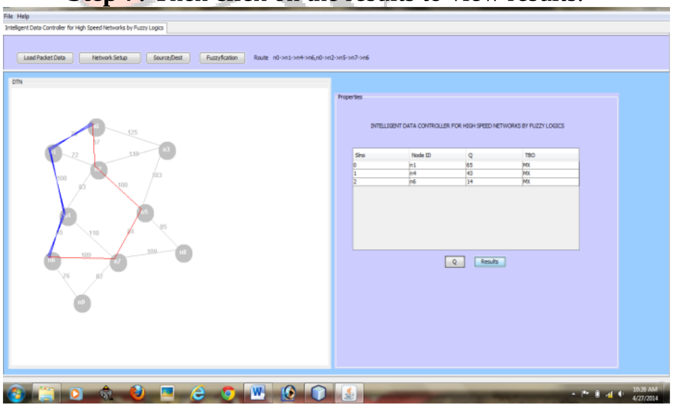




Step 6: Then click on the Q to view the Queue Size.



Step 7: Then click on the results to view results.



Step 7: Then click on the Fuzzyfication for further process.

Step 8: Later the route changes with the change ensued with the queue size and TBO values.

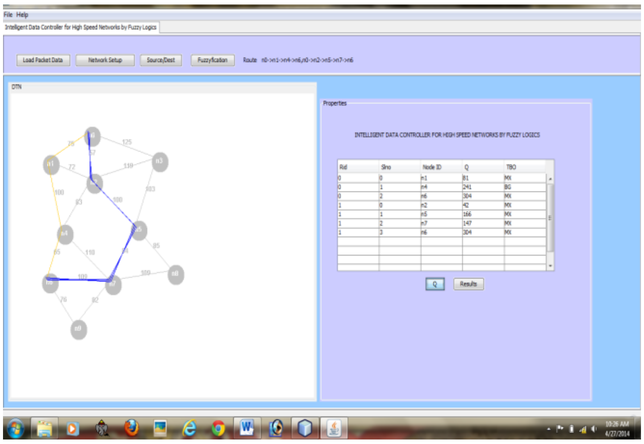


Figure 26: Fuzzyfication

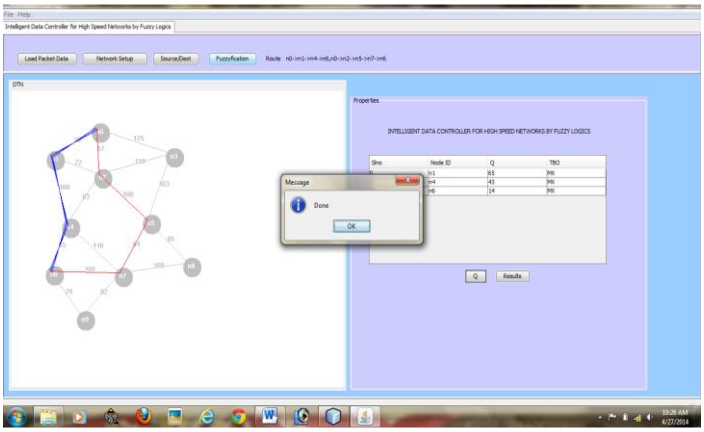


Figure 27: Check the queuesize

Step 9: Click on the results to view the results.

Step 10: Continue the process from Fuzzyfication until it reaches the maximum Q value.

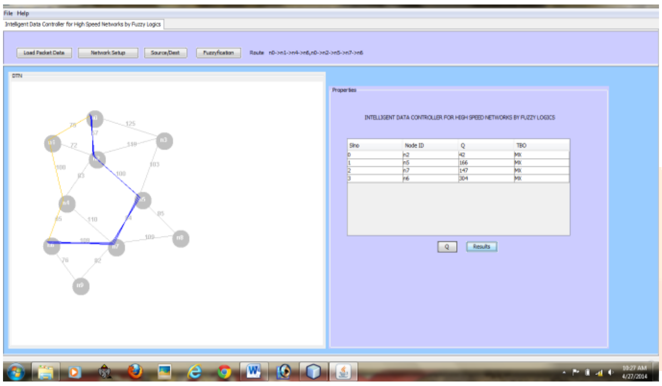


Figure 28: Route results

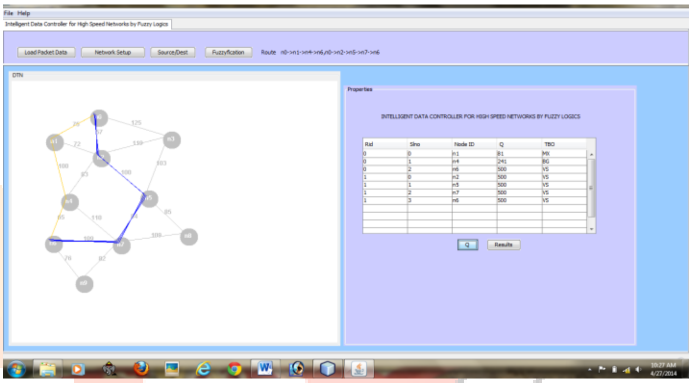


Figure 29: Fuzzyfication

Step 11: Finally click on the results to view the final route results.

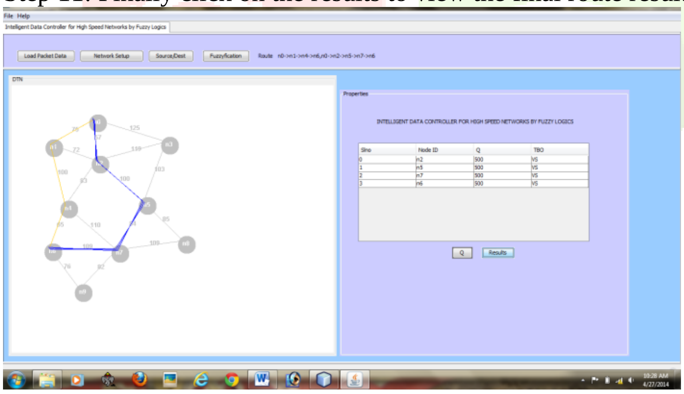


Figure 30: Final route

Conclusion

The controller is designed by paying attention to the disadvantages as well as the advantages of the existing congestion control protocols. The Intel Rate Controller has been proposed to manage the Internet congestion in order to assure the quality of service for different service applications. To verify the effectiveness and superiority of the Intel Rate Controller, FLC has ability to intelligently tackle the nonlinearity of the traffic control systems.

Future Enhancements

The implementation of fuzzy logic control will be widely accepted and implemented in industrial process control with control performance in accuracy, transient response, robustness and stability and with extraordinary mature. Fuzzy logic theory provides a convenient controller design approach based on expert knowledge which is close to human decision making, and readily helps engineers to model a complicated non-linear system.

References

[1] M. Welzl, Network Congestion Control: Managing Internet Traffic. John Wiley & Sons Ltd., 2005.

[2] R. Jain, "Congestion control and traffic management in ATM networks: recent advances and a survey," Computer Networks ISDN Syst., vol. 28, no. 13, pp. 1723–1738, Oct. 1996.

[3] K. M. Passino and S. Yurkovich, Fuzzy Control. Addison Wesley Longman Inc., 1998.

[4] C. Chang and R. Cheng, "Traffic control in an ATM network using fuzzy set theory," in Proc. 1994 IEEE INFOCOM, vol. 3. pp. 1200–1207.

[5] "OPNET modeler manuals," OPNET version 11.5, OPNET Technologies Inc., 2005.

[6] W. Hu and G. Xiao, "Design of congestion control based on instantaneous queue size in the routers," in Proc. 2009 IEEE GLOBECOM, pp.

[7] J. Liu and O. Yang, "Stability analysis and evaluation of the IntelRate controller for high-speed heterogeneous networks," in Proc. 2011 IEEE ICC, pp.

[8] http://www.f5.com/it-management/solutions/intelligent-traffic-management

[9] http://www.ieeexplore.org

[10] http://www.techreview.com