International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 3, Issue 4, July-August 2014 **ISSN 2278-6856**

Smart Traffic Management in High-Speed Networks by Fuzzy Logic Control

Kadanti Theja[1], P. Lakshmi Samyuktha [2]

[1]M. Tech, Dept. of CSE
Sree Vidyanikethan Engineering College
Tirupati-517 102, Andhra Pradesh, India

[2]Assistant Professor, Dept. of CSE
Sree Vidyanikethan Engineering College
Tirupati-517 102, Andhra Pradesh, India

Abstract

In view of the fast-emerging Internet traffic, this paper propose a distributed traffic management framework, in which routers are installed with intelligent data rate controllers to handle the traffic mass. Unlike other explicit traffic control protocols that have to evaluate network parameters in order to calculate the allowed source sending rate, our fuzzy-logic-based controller can calculate the router queue size directly; hence it avoids various potential performance problems arising from parameter estimations while decreasing much consumption of computation and memory resources in routers. As a network parameter, the queue size can be accurately observed and used to proactively decide if action should be taken to control the source sending rate, thus increasing the resilience of the network to traffic congestion. The communication Quality of Service is assured by the good performances of our scheme such as max-min fairness, low queueing delay and good robustness to network dynamics. Simulation results and comparisons have verified the effectiveness and showed that our new traffic management scheme can attain better performances than the existing protocols that depend on the estimation of network parameters.

Key words: Congestion control, fuzzy logic control, quality of service, max-min fairness, robustness, traffic management.

1. Introduction

Network traffic management can prevent a network from severe congestion and degradation in throughput-delay performance. Traffic congestion control is one of the effective approaches to manage the network traffic [1], [2]. Historically, Transmission Control Protocol Reno [3], [4] is a widely deployed congestion control protocol that handle the Internet traffic. It has the important feature that the network is treated as a black box and the source alter its window size based on packet loss signal [5]. However, as an implicit control protocol, TCP confronts various performance problems (e.g., utilization, fairness and stability) when the Internet BDP (Bandwidth-Delay Product) continues to increase. These have been widely investigated with various proposed solutions such as the AQM (Active Queue Management) schemes [6]–[10] whose control protocols are also implicit in nature.

As an alternative, a class of explicit congestion control protocols has been proposed to signal network traffic level more precisely by using multiple bits. Examples are the XCP[6], RCP [11], JetMax [12] and MaxNet [13]. These protocols have their 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. Specifically, JetMax and MaxNet signal network congestion by providing the required fair rate or the maximum link price, and then the final sending rate is decided by sources according to some demand functions or utility functions. 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 that 1) they can explicitly signal link traffic levels without maintaining per-flow state, and 2) the sources can converge their sending rates to some social optimum and achieve a certain optimization objective [12]. However, most of these explicit congestion control protocols have to estimate the bottleneck bandwidth in order to calculate the allowed source sending rate. Recent studies show that misestimation of link bandwidth may easily occur and can cause significant fairness and stability problems [14], [15]. There are some latest protocols on wireless applications such as 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. However, they still have the fundamental problem of inaccurate estimation resulting in performance degradation. In addition, their bandwidth probing speed may be too slow when the bandwidth jumps a lot. Also, they cannot keep the queue size stable due to oscillations, which in turn affects the stability of their sending rates. There are some explicit protocols that appear to compute the sending rates based solely on the queue size, but in fact they still need to estimate the number of active flows in a router, and this consumes CPU and memory resources. Examples are the rate-based controllers for packet switching networks and the ER (Explicit Rate) allocation algorithm for ATM (Asynchronous Transfer Mode) networks. For the API-RCP controller, both the original method (a truncated network model) and the improved method face a memory problem when dealing with many flows (that numbers in millions) arriving to a core router every hour. In some other controllers, the TBO (Target Buffer Occupancy) is designed to be as high as 3 times of the BDP, which can cause large queueing delay and thus degrading network performance, and this becomes even worse in the high-speed networks. Historically, the ER allocation algorithms in ATM networks also share the same problems because they need to evaluate the link bandwidth and/or the numbers of active VCs (Virtual Circuits). Some others adjust the source sending rates in binary-feedback switches or explicit feedback switches according to a few queue thresholds, which may cause unfairness as well as high cell loss rate [2]. Fuzzy Logic Control [16] has been considered for Intelligence Control. 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. Fuzzy logic theory gives 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. Fuzzy logic control has been applied in industrial process control and showed extraordinary and mature control performance in accuracy, transient response, robustness and stability.

Fuzzy Logic Control has found its applications to network congestion control since 1990. In early stage, it was used to do rate control in Asynchronous Transfer Mode network to guarantee the QoS. These control algorithms are explicit in nature, and they depend on absolute queue length (the maximum buffer size) instead of the TBO to adjust the allowed sending rate. Nevertheless, these early designs have various shortcomings including cell loss (even though cell loss is used as a congestion signal to compute the rate factor), queue size fluctuations, poor network latency, stability and low utilization. Later, Fuzzy Logic Control was used in Random Early Detection algorithm in TCP/IP network to reduce packet loss rate and improve utilization. However, they are still providing implicit or imprecise congestion signaling, and therefore can-not overcome the throughput fluctuations and conservative behavior of TCP sources. Specifically, the objectives of this paper are: 1) to design a new rate-based explicit congestion controller based on FLC to avoid estimating link parameters such as link bandwidth, the number of flows, packet loss and network latency, while remaining stable and robust to network dynamics (Hence, we make this controller “intelligent”); 2) to provide max-min fairness to achieve an effective bandwidth allocation and utilization; 3) to generate relatively smooth source throughput, maintain a reasonable network delay and achieve stable jitter performance by controlling the queue size; 4) to demonstrate our controller has a better QoS performance through case study. To achieve the above objectives, our new scheme pays attention to the following methodologies as well as the merits of the existing protocols. Firstly, in order to keep the implementation simple, like TCP, the new controller treats the network as a black box in the sense that queue size is the only parameter it relies on to adjust the source sending rate. The adoption of queue size as the unique congestion signal is inspired by the design experience of some previous AQM controllers (e.g., RED and API-RCP) in that queue size can be accurately measured and is able to effectively signal the onset of network congestion. Secondly, the controller retains the merits of the existing rate controllers such as XCP and RCP by providing explicit multi-bit congestion information without having to keep per-flow state information. Thirdly, we rely on the fuzzy logic theory to design our controller to form a traffic management procedure. Finally, we will employ OPNET modeler to verify the effectiveness and superiority of our scheme. The contributions of our work lie in: 1) fuzzy logic theory is used to design an explicit rate-based traffic management scheme for the high-speed IP networks; 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 controller unstable. For the remainder of the paper, the following notations and symbols pertain.

A	Edge value of MFs (Membership Functions) of e (t), beyond which the MFs of e (t) saturate
B	Buffer capacity
c (t)	Service rate (output link capacity) of a router
C	Edge value of MFs of g(e(t)), beyond which the MFs of g(e(t)) saturate
D	Outermost edge value of MFs of u (t)
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
m	Multiple of TBO to design the width limit for the MFs of input e(t) and g(e(t))
N	Number of LVs (Linguistic Values)
q0	TBO of a router
q (t)	IQSize (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 aggregate controller output)
μPj	Input fuzzy set of the IntelRate controller
μUj	Output fuzzy set of the IntelRate controller
τfi1	Time delay of a packet from source I to a router
τfi2	Time delay of a packet from a router to its destination i

τbi Feedback delay of a packet from destination I back to source i
τpi RTPD (Round Trip Propagation Delay)
τi RTT (Round Trip Time)

2. Traffic Management Principle and Modeling

We consider a backbone network interconnected by a number of geographically distributed routers, in which hosts are attached to the access routers which cooperate with the core routers to enable end-to-end communications. Congestion occurs when many flows traverse a router and cause its IQSize (Instantaneous Queue Size) to exceed the buffer capacity, thus making it a bottleneck in the Internet. Since any router may become bottleneck along an end-to-end data path, we would like each router to be able to manage its traffic. Below is the general operation principle of our new traffic management/control algorithm. Inside each router, our distributed traffic controller acts as a data rate regulator by measuring and monitoring the IQSize. As per its application, every host (source) requests a sending rate it desires by depositing a value into a dedicated field Req_rate inside the packet header. This field can be updated by any router en route. Specifically, each router along the data path will compute an allowed source transmission rate according to the IQSize and then compare it with the rate already recorded in Req_rate field. If the former is smaller than the latter, the Req_rate field in the packet header will be updated; otherwise it remains unchanged. After the packet arrives at the destination, the value of the Req_rate field reflects the allowed data rate from the most congested router along the path if the value is not more than the desired rate of the source. The receiver then sends this value back to the source via an ACK (ACKnowledgment) packet, and the source would update its current sending rate accordingly. If no router modifies Req_rate field, it means that all routers en route allow the source to send its data with the requested desired rate. In order to implement our new controller in each router, we model a typical AQM router in Figure 1 with M sources sending their Internet traffic to their respective destinations. For i = 1, 2, . . . , M, $u^{-}_{i}(t)$ is the current sending rate of source i; $u_i(t)$ is the sending rate of source i determined by the routers along the end-to-end path; y(t) is the incoming aggregate controlled flow rate; v(t) is the incoming aggregate uncontrolled flow rate, and c(t) is the link bandwidth (measured in bps). For a particular source-destination pair i, τ_{fi1} is the time delay of a packet from source i to the router, and τ_{fi2} is the time delay of the packet of source i from the router to the destination i, while τ_{bi} is the feedback delay from destination i back to source i. Obviously, $\tau_{pi} = \tau_{fi1} + \tau_{fi2} + \tau_{bi}$ is the RTPD (Round Trip Propagation Delay). Considering other delays en route (e.g., queueing delay), source i may update its current rate $u^{-}(t)$ according to the $u_i(t)$ when the ACK packet arrives after one RTT (Round Trip Time) τ_i. Considering the possible dynamics of both incoming traffic and link bandwidth in the router in Figure 1, we model the bottleneck link with a queue in which both the controlled arrival rate y(t) and the service rate c(t) may vary with respect to time. Let q(t) be the router IQSize. The variations in y(t) and/or c(t) can cause changes in the queue size of a router, as expressed in the following differential equation.

$$q(t) = \begin{array}{ll} y(t) + v(t) - c(t) & q(t) > 0 \\ [y(t) + v(t) - c(t)]^{+} & q(t) = 0 \end{array}$$

where $[x]^{+} = \max(0, x)$.

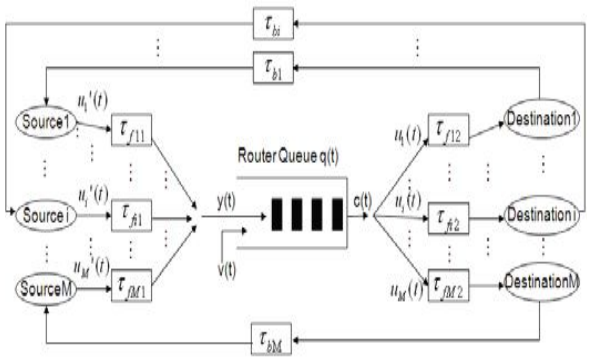


Figure 1: System model of an AQM router

3. The Intel Rate Controller Design

Figure 2 depicts the components of our fuzzy logic traffic controller for controlling traffic in the network system defined in Figure 1. Called the IntelRate, 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 queue deviation e(t) = q0 −q(t) is one of the two inputs of the controller.

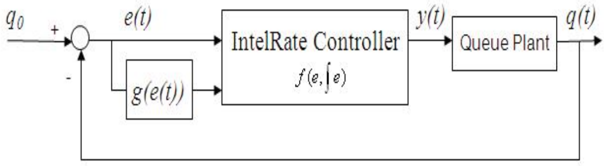


Figure 2: The IntelRate closed-loop control system

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 − τi). Under heavy traffic situations, the IntelRate 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, IQSize 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, fuzzification, inference and defuzzification. The concepts 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 sequel, we formulate our new con-troller by following those four steps along with designing the fuzzy linguistic descriptions and the membership functions.

3.1 Linguistic Description and Rule Base

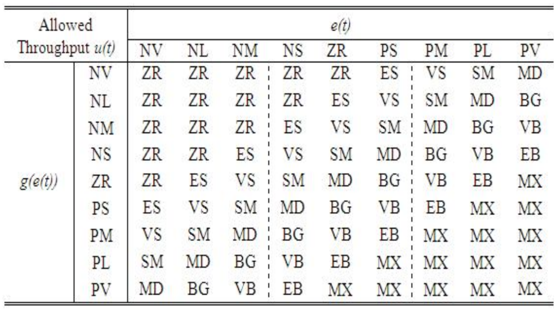
We define the crisp inputs e(t), g(e(t)) and output u(t) with the linguistic variables _e (t), _g (e (t)) and _u (t), respectively. There are N(N = 1, 2, 3, . . .) LVs (Linguistic Values) assigned to each of these linguistic variables. Specifically ,we let $P_i = \{P_i^j : j = 1, 2, \dots ,N \}$ be the input LVs with i = 1 for e_(t) and i = 2 for g_(e (t)), and let $U = \{U^j : j = 1, 2, \dots ,N \}$ for _u (t). For example, when N = 9, we can assign the following values or both the inputs e(t) and g(e(t)). P_i^1="Negative Very Large (NV)," P_i^2="Negative Large (NL)," P_i^3= "Negative Medium (NM)," P_i^4="Negative Small (NS)," P_i^5="Zero (ZR)," P_i^6="Positive Small (PS)," P_i^7="Positive Medium (PM)," P_i^8="Positive Large (PL)," and P_i^9 ="Positive Very Large (PV)," i = 1, 2. Similarly, we can designate the output when N = 9 with the following linguistic values. U^1 ="Zero (ZR)," U^2 ="Extremely Small (ES)," U^3="Very Small (VS)," U^4 ="Small (SM)," U^5 ="Medium (MD)," U^6 ="Big (BG)," U^7 ="Very Big (VB)," U^8 ="Extremely Big (EB)," and U^9 ="Maximum (MX)." Table I illustrates the controller rule base using N = 9. The rule base is the set of linguistic rules used to map the inputs to the output using the "If. . . Then. . . " format, e.g. "If e(t) is ZR (Zero) and g(e(t)) is PS (Positive Small), Then u(t) is BG (Big)." In the following sections, we refer to a rule in this able by the notation (P_1^j,P_2^k,U^l), where j, k, l = 1, 2, . . .,N, e.g.(P_1^5,P_2^2,U^2)= (ZR,NL,ES).

3.2 Membership Function, Fuzzification and Reference

Our IntelRate controller employs the isosceles triangular and trapezoid-like functions as its MFs (Membership Func-tions).Figure 3 describes the MFs used to determine the certainty of a crisp input or output. We let P_1 be the UoD[1] for the input $p_1 = e(t)$, and P_2 be the UoD for the input $p_2 = g(e(t))$. The value of MFs (i.e., the certainty degree) for crisp inputs $p_i(i = 1, 2)$ is designated by $\mu_{Pi}j$ (p_i). Similarly, we let

TABLE I

RULE TABLE FOR INTELRATE CONTROLLER(9 LVS)



Allowed Throughput u(t)		e(t)								
		NV	NL	NM	NS	ZR	PS	PM	PL	PV
g(e(t))	NV	ZR	ZR	ZR	ZR	ZR	ES	VS	SM	MD
	NL	ZR	ZR	ZR	ZR	ES	VS	SM	MD	BG
	NM	ZR	ZR	ZR	ES	VS	SM	MD	BG	VB
	NS	ZR	ZR	ES	VS	SM	MD	BG	VB	EB
	ZR	ZR	ES	VS	SM	MD	BG	VB	EB	MX
	PS	ES	VS	SM	MD	BG	VB	EB	MX	MX
	PM	VS	SM	MD	BG	VB	EB	MX	MX	MX
	PL	SM	MD	BG	VB	EB	MX	MX	MX	MX
	PV	MD	BG	VB	EB	MX	MX	MX	MX	MX

Z be the UoD of the output z = u(t), and the certainty degree of the crisp output z is designated by $\mu_U j$ (z). The above μ_P j (p_i) or $\mu_U j$ (z) is obtained by the "Singleton Fuzzification" [i] method [36]. Thus the input and output fuzzy sets can be defined with P_i^j= {(pi, μPj (pi) : pi Pi)}, i = 1, 2 and , Uj = {(z, μUj (z)} : z ∈ Z}, j = 1, 2, . . .,N, respectively. To determine how much a rule is certain to a current situation, our controller applies the Zadeh AND logic to perform the inference mechanism, e.g. for crisp inputs p1 and p2, the final certainty (also referred to as the firing level) of a rule is computed with $\mu_{P1^m \cap P2^m}$= min $\mu_{P1^m}(p_1)$, $\mu_{P2^m}(p_2)$: p [i] P_i i = 1, 2 and m, n = 1, 2, . . .,N, where min is the minimum operation in the Zadeh AND logic. While Figure 3 is used to illustrate the design of a general FS, the designated values actually come from an example of N = 9 LVs with the absolute values of both the upper and lower limits of g(e(t)) set to mq0. Since e(t) is bounded by the physical size of a queue, we have the boundaries according to the limits q0−B ≤ e(t) ≤ q0. The vertical dashed lines in Figure 4 denote those boundaries of inputs or output. Accordingly, the dashed box in Table I contains the rules that the IntelRate controller operates on. There are three LVs that e(t) can take (i.e., "NS," "ZR" and "PS") compared with the nine values (from NV to PV covering all the LVs) for g(e(t)). As shown, nine different LVs (from ZR to MX) are used to give a gradual change in the output u(t) for each combination of e(t) and g(e(t)).

3.3 Defuzzification

For the defuzzification algorithm, the Intel Rate controller applies the COG (Center of Gravity) method to obtain the crisp output with the equation u (t) = $(\Sigma^k_{j=1}$ cjSj) / $(\Sigma^k_{j=1}$ Sj), where k is the number of rules; cj is the bottom centroid of a triangular in the output MFs, and Sj is the area of a triangle with its top chopped off as per $\mu_{P1^m \cap P2^m}$ discussed above. Since each parameter in the crisp input pair (p1, p2) can take on two different values in the IntelRate controller, we have altogether k = 4 rules for defuzzification each time.

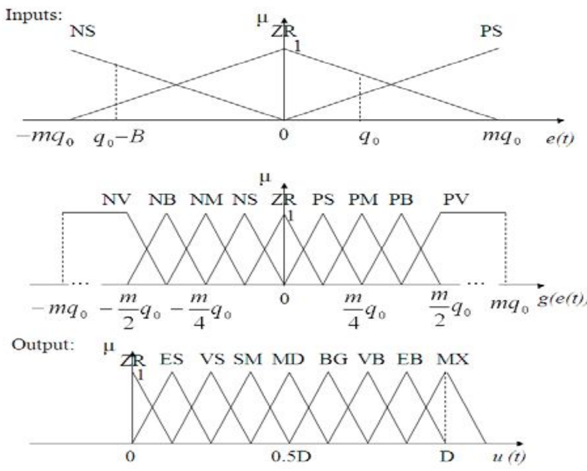


Figure 3: Membership functions with FS

3.4 The Control Procedure

Figure 4 shows the new field in the packet congestion header that we need to support our controller algorithm for the operation principle mentioned in Section 2. As

International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 3, Issue 4, July-August 2014 **ISSN 2278-6856**

discussed, we need to include in 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.

Req_rate

Figure 4: Congestion header

Below is the traffic-handling procedure of the IntelRate controller in a router.

(1) Upon the arrival of a packet, the router take Req_rate from the congestion header of the packet.
(2) Sample IQSize q(t) and update e(t) and g(e(t)).
(3) Calculate the output u(t) and compare it with Req_rate.
(3a) If u(t) < Req_rate, it means that the link does not have enough bandwidth to accommodate the requested amount of sending rate. The Req_rate field in the congestion header is then updated by u(t).
(3b) Otherwise the Req_rate field remains unchanged.
(4) If an operation cycle d is over, update the crisp output u(t) and the output edge value of D.

Note that this procedure actually allows the router to perform the max-min fairness in that the greedy flows are always restricted to u(t) by a router under heavy traffic conditions while those small flows whose desired sending rate are smaller than u(t) along their data path have no such a restriction. As mentioned in the operation principle (Section 2), when the packet arrives at the destination, the receiver extracts Req_rate from the header and records it into the ACK packet before sending it back to the source.

4. Explicit Congestion Notification

In current TCP/IP networks, TCP relies on packet drops as the indication of congestion. The TCP source detects dropped packets either from the receipt of three duplicate acknowledgements (ACKs) or after the timeout of a retransmit timer, and responds to a dropped packet by reducing the congestion window. the role of the router in generating ECN messages. ECN messages could be generated either by an IP router or by a boundary router for an ATM network that carries TCP/IP traffic. The advantages of ECN include a reduction in packet drops and packet delay for low-bandwidth delay-sensitive TCP traffic and a prompt notification to end nodes of network congestion. A main advantage of ECN mechanisms is in avoiding unnecessary packet drops, and therefore avoiding unnecessary delay for packets from low-bandwidth delay sensitive TCP connections.

5. Performance Evaluation

The single bottleneck network in Figure 5 is used to investigate the controller behaviour of the most congested router. We choose Router 1 as the only bottleneck in the network, whereas Router 2 is configured to have sufficiently high service rate and big buffer B so that congestion never happens there. The numbers in Figure 5 are the IDs of the subnets/groups attached to each router. Their configuration is summarized in Table II, in which there are M = 11 subnet pairs, which form the source-destination data flows in the network, and they run various Internet applications such as the long-lived ftp, short-lived http, or the unresponsive UDP-like flows. Since the link bandwidth we want to simulate have a magnitude of Giga bits per second, we need to use 20 flows in each subnet to generate enough traffic to produce congestion. All flows within each group have the same RTPD and behaviour, but different from the flows of other groups. The RTPD includes the forward path propagation delay and the feedback propagation delay, but does not include the queueing delay, which may vary according to our settings of TBO size in the experiments. The reverse traffic is generated by the destinations when they piggyback the ACK information back to the sources.

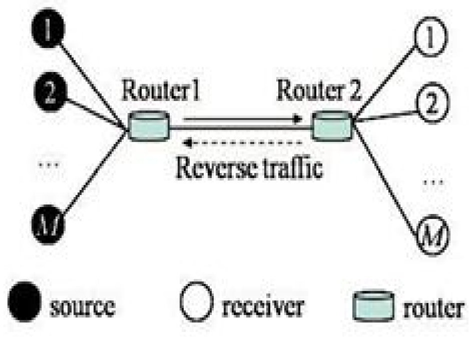


Figure 5: Simulation network

TABLE II
SOURCES CHARACTERISTICS

Subnet ID	Source ID	Flow NO.	RTPD(ms)
ftp group 1	1-20	ftp 1-20	80
ftp group 2	21-40	ftp 21-40	120
ftp group 3	41-60	ftp 41-60	160
ftp group 4	61-80	ftp 61-80	200
ftp group 5	81-100	ftp 81-100	240
http group 1	101-120	http 1-20	80
http group 2	120-140	http 21-40	120
http group 3	141-160	http 41-60	160
http group 4	161-180	http 61-80	200
http group 5	181-200	http 81-100	240
uncontrolled ftp	201	UDP 1	160

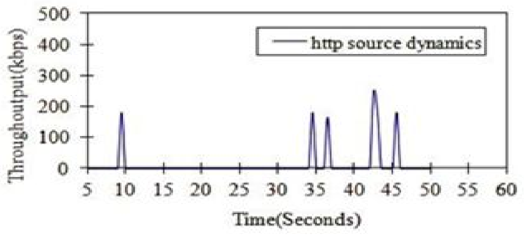


Figure 6: Http sessions example

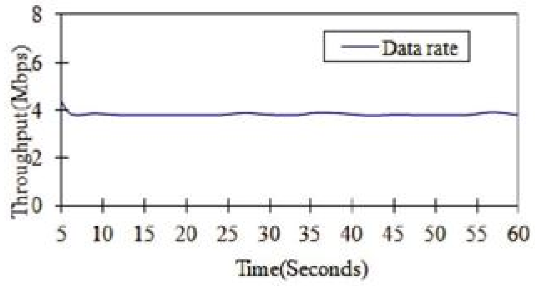


Figure 7: Uncontrolled ftp flow

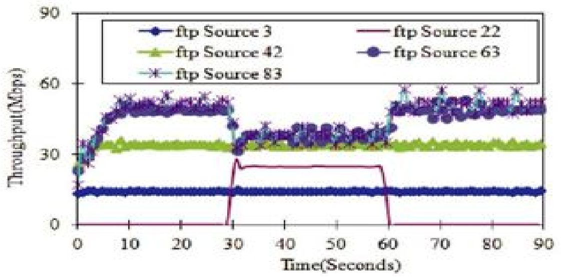


Figure 8(a): Source throughput dynamics under traffic change

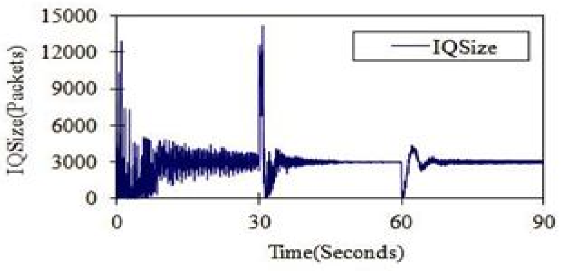


Figure 8(b): IQSize dynamics under traffic change

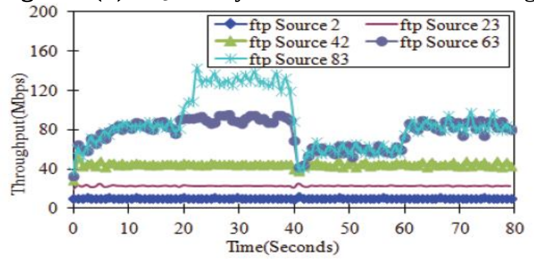


Figure 9(a): Source throughput dynamics

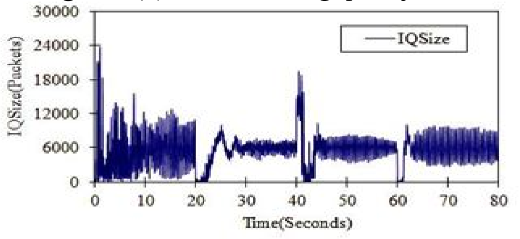


Figure 9(b): IQ Size dynamics

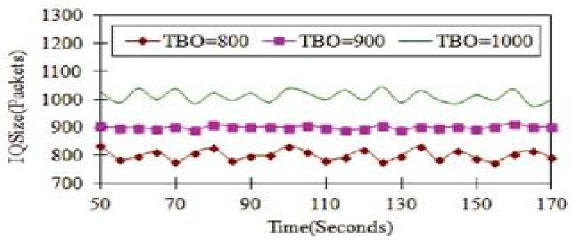


Figure 10(a): IQSize under different TBOs

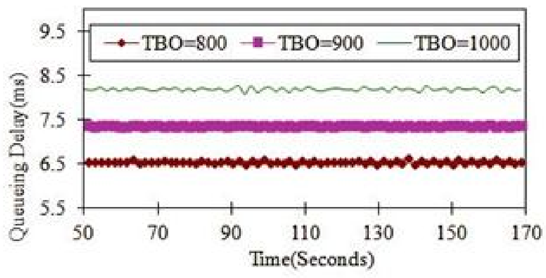


Figure 10(b): Queueing delay under different TBOs

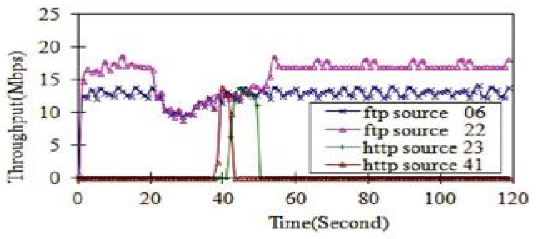


Figure 11: Flow throughput dynamics

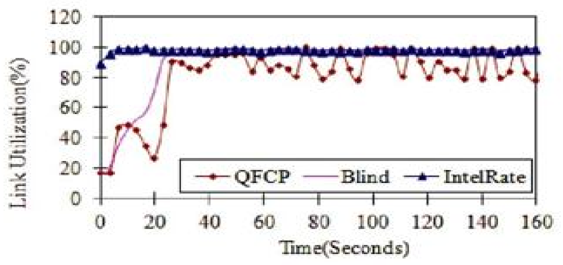


Figure 12: Link utilization

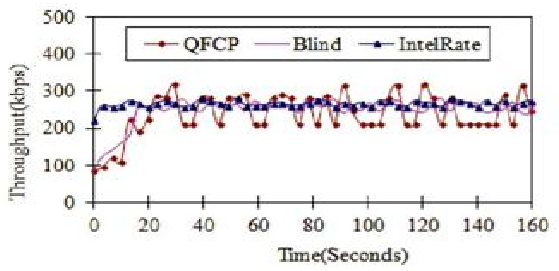


Figure 13: Source throughput of the three controllers

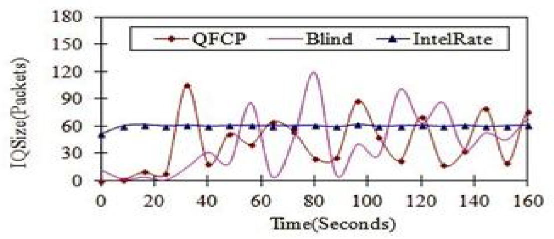


Figure 14: IQSize of the three controllers

6. Conclusion

A novel traffic management scheme, called the IntelRate controller, has been proposed to manage the Internet conges-tion in order to assure the quality of service for different ser-vice applications. The controller is designed by paying atten-tion to the disadvantages as well as the advantages of the exist-ing congestion control protocols. As a distributed operation in networks, the IntelRate controller uses the instantaneous queue size alone to effectively throttle the source sending rate with max-min fairness. To verify the effectiveness and superiority of the IntelRate controller, extensive experiments have been conducted in OPNET modeler. In addition to the feature of the FLC being able to intelligently tackle the non-linearity of the traffic control systems, the success of the IntelRate controller is also attributed to the careful design of the fuzzy logic elements.

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] V. Jacobson, “Congestion avoidance and control,” in Proc. 1988 SIG-COMM, pp. 314–329.
[4] V. Jacobson, “Modified TCP congestion avoidance algorithm,” Apr. 1990.
[5] K. K. Ramakrishnan and S. Floyd, “Proposals to add explicit congestion notification (ECN) to IP,” RFC 2481, Jan. 1999.
[6] D. Katabi, M. Handley, and C. Rohrs, “Congestion control for high bandwidth-delay product networks,” in Proc. 2002 SIGCOMM, pp. 89– 102.
[7] S. H. Low, F. Paganini, J. Wang, et al., “Dynamics of TCP/AQM and a scalable control,” in Proc. 2002 IEEE INFOCOM, vol. 1, pp. 239–248.
[8] S. Floyd, “High-speed TCP for large congestion windows,” RFC 3649, Dec. 2003.
[9] W. Feng and S. Vanichpun, “Enabling compatibility between TCP Reno and TCP Vegas,” in Proc. 2003 Symp. Applications Internet, pp. 301– 308.
[10] M. M. Hassani and R. Berangi, “An analytical model for evaluating utilization of TCP Reno,” in Proc. 2007 Int. Conf. Computer Syst. Technologies, p. 14-1-7.
[11] N. Dukkipati, N. McKeown, and A. G. Fraser, “RCP-AC congestion control to make flows complete quickly in any environment,” in Proc. 2006 IEEE INFOCOM, pp. 1–5.
[12] Y. Zhang, D. Leonard, and D. Loguinov, “JetMax: scalable max-min congestion control for high-speed heterogeneous networks,” in Proc. 2006 IEEE INFOCOM, pp. 1–13.
[13] B. Wydrowski, L. Andrew, and M. Zukerman, “MaxNet: congestion control architecture for scalable networks,” IEEE Commun. Lett., vol. 7, no. 10, pp. 511–513, Oct. 2003.
[14] Y. Zhang and M. Ahmed, “A control theoretic analysis of XCP,” in Proc. 2005 IEEE INFOCOM, vol. 4, pp. 2831–2835.
[15] Y. Zhang and T. R. Henderson, “An implementation and experimental study of the explicit control protocol (XCP),” in Proc. 2005 IEEE INFOCOM, vol. 2, pp. 1037–1048.
[16] K. M. Passino and S. Yurkovich, Fuzzy Control. Addison Wesley Longman Inc., 1998

Author

Kadanti Theja received the B.Tech degree in Computer Science and Engineering from Sree Vidyanikethan Engineering College affiliated to JNTUA, Tirupati, Andhra Pradesh, in 2012. He is currently doing M.Tech in Computer Networks and Information Security from the Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, Tirupati, Andhra Pradesh, during 2012-2014.His current research interests include network security and cloud computing.

P. Lakshmi Samyuktha received the M.Tech degree in Computer Science and Engineering from Acharya Nagarjuna University, Andhra Pradesh, in 2010. She is currently Working as Assistant Professor in Computer Science Department at Sree Vidyanikethan Engineering College, Tirupati, Andhra Pradesh. Her current research interests include Computer Networks, Network Security and Software Engineering.