International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

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

G. Ravi[1], Rahul Kumar[2]

[1]JNTU University, MRCET College,
Opp.A.P.Forest Academy, Kompally, Maisammaguda, Dullapally, Hyderabad, Telangana 500100, India

[1]JNTU University, MRCET College,
Opp.A.P.Forest Academy, Kompally, Maisammaguda, Dullapally, Hyderabad, Telangana 500100, India

Abstract*: Unlike other explicit traffic control protocols that have to estimate network parameters in order to compute the allowed source sending rate, our fuzzy-logic-based controller can measure the router queue size directly; hence it avoids various potential performance problems arising from parameter estimations while reducing much consumption of computation and memory resources in routers. 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 communication 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, 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. 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 However, as an implicit control protocol, TCP encounters various performance problems 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) 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 protocol shave 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. The advantages of these router-assisted protocols are that

1) They can explicitly signal link traffic levels without maintaining per fow rate,
2) The sources can converge their sending rates to some social optimum and achieve a certain optimization objective

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 networks.

From the perspective of network and service management, the aforementioned congestion control approaches have QoS (Quality of Service) problems in that they cannot guarantee a certain level of performance under some situations due to design drawbacks. There are many different approaches to improve QoS. For example, admission control, as a network traffic management approach, can guarantee QoS by checking the availability of network bandwidth before establishing a connection. Service priority as another approach can be used to improve QoS by providing different service priorities to different users,. Pricing or routing policies are also found to address QoS problems. However, they are outside the scope of this paper that focuses on congestion control as an approach to address the QoS management problem.

FLC (Fuzzy Logic Control) [36] has been considered for IC (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 modelling imprecision. 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. In fact, fuzzy logic control has been widely applied in industrial process control and showed extraordinary and mature control performance in accuracy, transient response, robustness and stability.

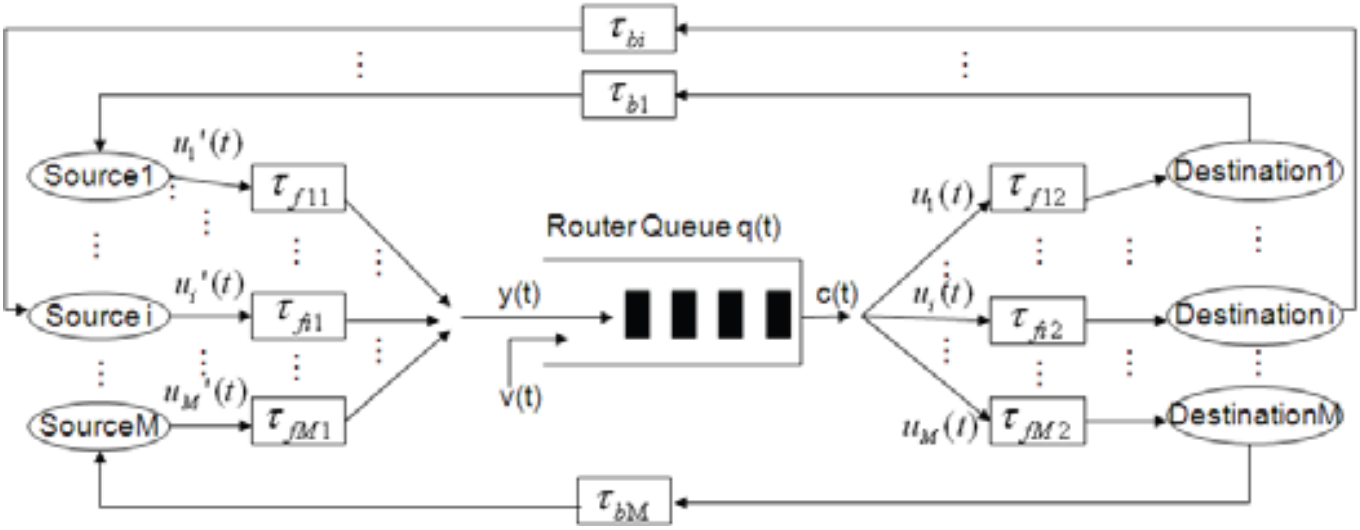
FLC has found its applications to network congestion control since 1990. In early stage, it was used to do rate control in ATM network to guarantee the QoS. These control algorithms are explicit in nature, and they depend on

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 queue size fluctuations, poor network latency, stability and low utilization. Later, 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 signalling, and therefore cannot overcome the throughput fluctuations and conservative schemes.

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 modeller to verify the effectiveness and superiority of our scheme.

The contributions of our work lie in:

1) Using fuzzy logic theory to design an explicit rate-based traffic management scheme (called the IntelRate controller) for the high-speed IP networks;
2) The application of such a fuzzy logic controller using less performance parameters while providing better performances
3) Than the existing explicit traffic control protocols;
4) The design of a Fuzzy Smoother mechanism that can generate relatively smooth flow throughput;
5) The capability of our algorithm to provide max-min fairness even under large network dynamics that usually render many existing controllers unstable.



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 QSize. As per its application, every host (source) requests a sending rate it desires by depositing a value into a dedicated field *eq_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.

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)
q_0 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)
μ_{Pj} Input fuzzy set of the IntelRate controller
μ_{Uj} Output fuzzy set of the IntelRate controller
τ_{fi1} Time delay of a packet from source I to a router
τ_{fi2} Time delay of a packet from a router to its destination i
τ_{bi} Feedback delay of a packet from destination i back to source i
τ_{pi} RTPD (Round Trip Propagation

The *Req_rate* field in the packet header will be pdated; 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.

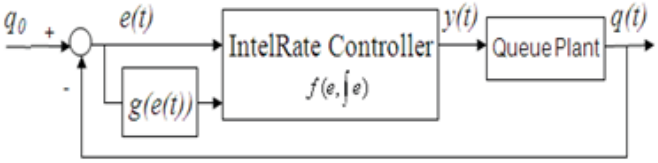
In order to implement our new controller in each router, we model a typical AQM router in Fig. 1 with M sources sending their Internet traffic to their respective destinations. For $i = 1, 2, \ldots, M$, $u\_i(t)$ is the current sending rate of source i; $u_i(t)$ is the sending rate of source i determined 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 Fig. 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.

$$q(t) = y(t) + v(t) - c(t) \quad q(t) > 0$$
$$q(t) = [y(t) + v(t) - c(t)]^+ \quad q(t) = 0$$

The following assumptions for the remainder of this paper pertain.

1) Every source requests a desired sending rate from the network according to its application.
2) A destination always has enough buffer space to receive data from its source. This is because we do not want the destination to impose any constraint on the source sending rate when we verify the effect of our new control scheme in a bottlenecked router.
3) The propagation delay and the queuing delay along the data path components like processing delay of a packet in routers or hosts are negligible in comparison.
4) The queueing discipline of routers is FIFO (First-In-First-Out).
5) Long-lived flows with infinitely long files are used to approximate the greedy behavior of a source when active. This would generate the severest traffic in order for us to verify the robustness of the new scheme.



3. 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

Table 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), (except 2-column illustrations may cross the gap). If your figure has two parts, include the labels "(a)" and "(b)".

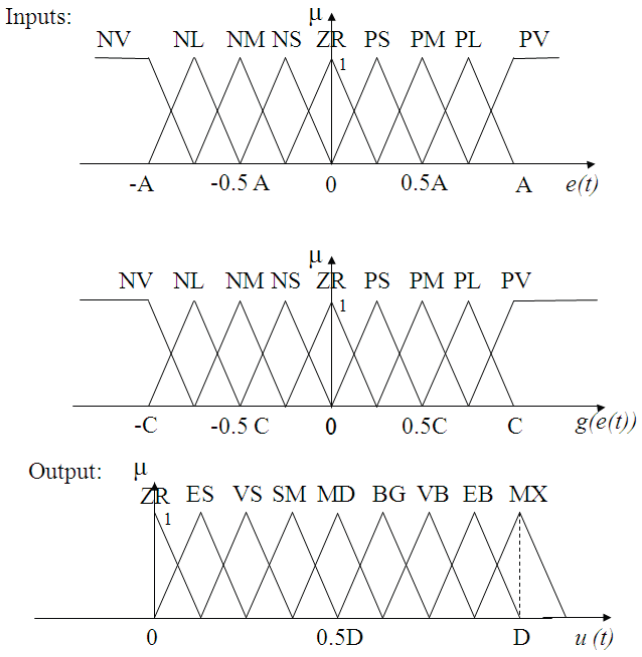
A. Linguistic Description and Rule Base We define the crisp inputs $e(t)$, $g(e(t))$ and output $u(t)$ with the linguistic variables $\_e(t)$, $\_g(e(t))$ and $\_u(t)$, respectively. There aren $N(N = 1, 2, 3, \ldots)$ LVs (Linguistic Values) assigned to each of these linguistic variables. Specifically,
we let $P_i = P_{ji} : j = 1, 2, \cdots, N\_$ be the input LVs with $i\_ = 1$ for $e\_(t)$ and $i = 2$ for $g\_(e(t))$, and let $U = U_j : j\ 1, 2, \ldots, N\_$ for $\_u(t)$. For example, when $N = 9$, we can assign the following values for both the inputs $e(t)$ and $g(e(t))$. P_{1i}="NegativeVery Large (NV)," P_{2i}="Negative.

B. Membership Function, Fuzzification and Reference
Our IntelRate controller employs the isosceles triangular and rapezoid -like functions a sits MFs describes the MFs used to determine the certainty of a crisp input or output. We let p_1 bethe UoD_1 for the input $p_1 = e(t)$, and P_2 be the UoD for the nput $p_2 = g(e(t))$. The value of MFs for crisp inputs $p_i (i = 1, 2)$ is designated by $\mu_{P_j}(p_i)$. Similarly,we let Z be the UoD of the output $z = u(t)$, and the certainty degree of the crisp output z is designated by $\mu_{U_j}(z)$.

C. Defuzzification:for the defuzzification algorithm, the IntelRate controller applies the COG (Center of Gravity) method to obtain the crisp output with the equation here k is the number of rules; c_j is the bottom centroid of a triangular in the output MFs, and S_j is the area of a triangle with its top chopped off as per μ_{P_m} ove. Since each parameter in the crisp input pair $(p1, p2)$ can take on two different values in the IntelRate controller,

D. Design Parameters From our design above, one can see there are different parameters which ultimately will affect the performance of our traffic controller. Below are the discussions of some important design issues we have experienced. Some of them were determined via our extensive experiments.

a) TBO
From the perspective of the queueing delay, the TBO value should be as small as possible. This is especially true under the heavy traffic conditions when the queue is to be stabilized at TBO. Therefore, a bigger TBO will result in longer steady state queueing delay, which is not desirable to some Internetapplications such as the real-time video. On the other hand, too small TBOs are not preferred in that it may result in the IntelRate controller either a restricted throughput or an under-utilized bandwidth performance. For easy illustration, we choose $q_0 = 0$ as the extreme case. It requires the controller to vacate the queue in whatever traffic conditions. In such a case, the LVs of $g(e(t))$ can only make transitions among ZR, NS, NM, NL and NV because the queue size cannot be greater than zero. Accordingly the controller output (i.e., the allowed source sending rate) can at most reach BG, and never goes to the MX .



b) The number N of LVs
The choice of N has to consider the trade-off between the throughput performance and the computation complexity of the controller. Big N complicates the controller in the sense that it has to do more logic computations on choosing the allowed sending rate according to the rules. Such a computation complexity can affect the rise time in the transient response of the controller as well as the control performance in the case of network parameter changes (e.g. the settling time during large bandwidth variation). On the other hand, a small N may lead the controller output to oscillate due to the big partitions of the LVs.

c) The Output Edge Value D
With reference to Fig. 4, the outermost edge value D in the output MFs corresponds to the maximum sending rate that the controller can output. This parameter is chosen to be the maximum value of the *Req_rate* field among the active incoming flows, i.e. $D = \max(\_u_1, \_u_2, \ldots \_u_R)$, where $\_u_i$, $i = 1, 2, \ldots, R$, is the value recorded in *Req_rate* of each packet.

d) The Width Limit m
The parameter m defines the base width of each membership function in the FS. Since it also affects the extent of overlapping between the adjacent MFs, the basic consideration to choose an appropriate parameter m is to have a smaller TBO while remaining the controller output smooth. An inappropriate m may have similar side effects like the parameter N. Too big of an m value leads to small partitions along $e(t)$ or $g(e(t))$, and thus may affect the response time of the controller. On the other hand, having m too small may cause fluctuations in the controller output due to the too big partitions along $e(t)$ and $g(e(t))$.

Req_rate

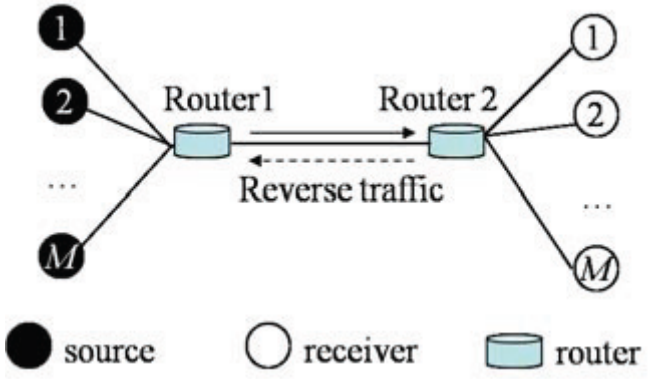
e) Buffer Size B

The determination of buffer size B is closely related to the chosen value of TBO. Although one can choose $B = q_0$, the smallest possible value, this is usually not desirable for the following two basic reasons: 1) a controller usually has various steady state error issues [49], and it is impossible that the queue size can be exactly pegged at TBO; 2) the dynamic Internet traffic can sometimes cause a surge on the queue size, e.g., due to a sudden traffic swarm-in or an unexpected bandwidth reduction. Therefore, the B should be greater than
the TBO.

E. The Control Procedure

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

1) Upon the arrival of a packet, the router extracts *Req_rate* from the congestion header of the packet.
2) Sample IQSize $q(t)$ and update $e(t)$ and $g(e(t))$.
3) Compute the output $u(t)$ and compare it with *Req rate*.



4. Simulation Network

Table : Sources Characteristics

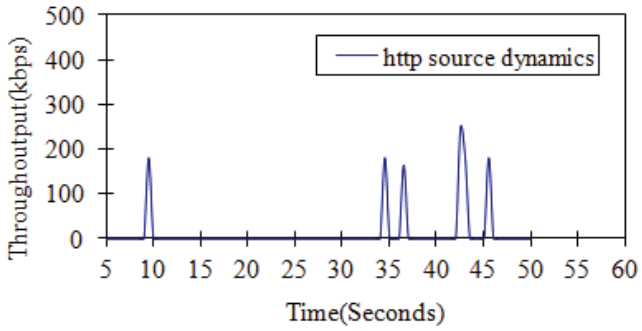
Subnet ID	Source ID	Flow NO.	RTPD(ms)
ftp group 1	1-20	ftp 1-20	80
ftp group 2	21-40	ftp 21-40	120
ftp group 3	41-60	ftp 41-60	160
ftp group 4	61-80	ftp 61-80	200
ftp group 5	81-100	ftp 81-100	240
http group 1	101-120	http 1-20	80
http group 2	120-140	http 21-40	120
http group 3	141-160	http 41-60	160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

5. Performance Evaluation

The capability of the IntelRate controller is demonstrated by performance evaluations through a series of experiments. We will first describe the simulated network and performance measures of a case study in Section A. Section B demonstrates the system robustness upon large network changes. Queuing jitter control and the effect of short-lived traffic will be discussed in Sections C and D. Section E evaluates the bandwidth utilization and packet loss rate of the IntelRate controller. Finally, Section F discusses the choices of some design parameters.

A. Simulated Network

The single bottleneck network in Fig. 6 is used to investigate the controller behavior 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 Fig. 6 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



The TBO and the buffer capacity B in Router 1 in each experiment are set according to the approaches discussed in Section III. We also adopt some typical values from the experiments of existing works so that we can make our experiments more meaningful. In particular, all the ftp packets have the same size of 1024 bytes [19] while the http packet size is uniformly distributed in the range of [800, 1300] bytes.

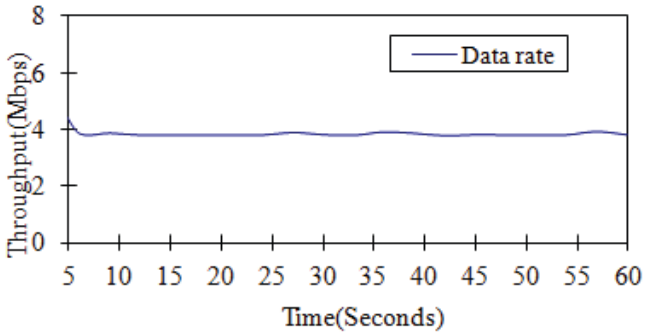


Figure: Uncontrolled ftp flow.

The controller is evaluated by the following performance measures.
1) Source throughput (or source sending rate) is defined to be the average number of bits successfully sent out by a source per second, i.e. bits/second. Here, a bit is considered to be successfully sent out if it is part of a packet that has been successfully sent.
2) IQSize is the length of the bottleneck buffer queue seen by a departing packet.
3) 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.
4) Queuing jitter is the variation of queuing delay due to the queue length dynamics, and is defined as the variance of the queuing delay.
5) Link (or bottleneck) utilization is the ratio between the current actual throughput in the bottleneck and the maximum data rate of the bottleneck. It is expressed as a fraction less than one or as a percentage.
6) Packet loss rate is the ratio between the number of packet dropped and the number of total packets received per second by the bottleneck.
7) Max-min fairness: A feasible allocation of rates is 'maximum fair' if and only if an increase of any rate within the domain of feasible allocations must be at the cost of a decrease of some already smaller or equal rates.

B. Robustness to Large Network Changes

The real world Internet traffic is always dynamic. Below we shall investigate the performance of our controller's when faced with drastic network changes such as the variations of the number of flows or the available bandwidth.

a) Sudden Traffic Changes

Sudden traffic change may occur when the numbers of long-lived or short-lived flows or some uncontrolled flows change at the ingress/egress. To test the controller capability to handle the dynamics of the real Internet traffic, we conduct an experiment in which the bottleneck bandwidth is 3Gbps. TBO is set to be 3000 packets, which results in a queuing delay of about 8.19ms when the queue size stabilizes at TBO (i.e., 3000* 1024* 8/3Gbps=8.19 ms). The buffer capacity B=30000 packets. The source desired rates of ftp groups 1 to 5 are 14.75 Mbps, 24.55 Mbps, 34.41 Mbps, 49.15 Mbps and 65.54Mbps, respectively. In the first 30 seconds, only ftp flows in groups 1, 3, 4 and 5 are active, and at t = 30s, 20 ftp flows in group 2 join in the bottleneck traffic and then drop out at t = 60s. The objective of this experiment is to see how our controller responds to traffic changes and if the system can maintain stable.

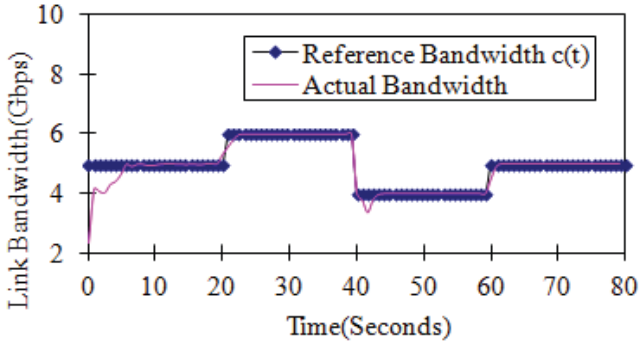
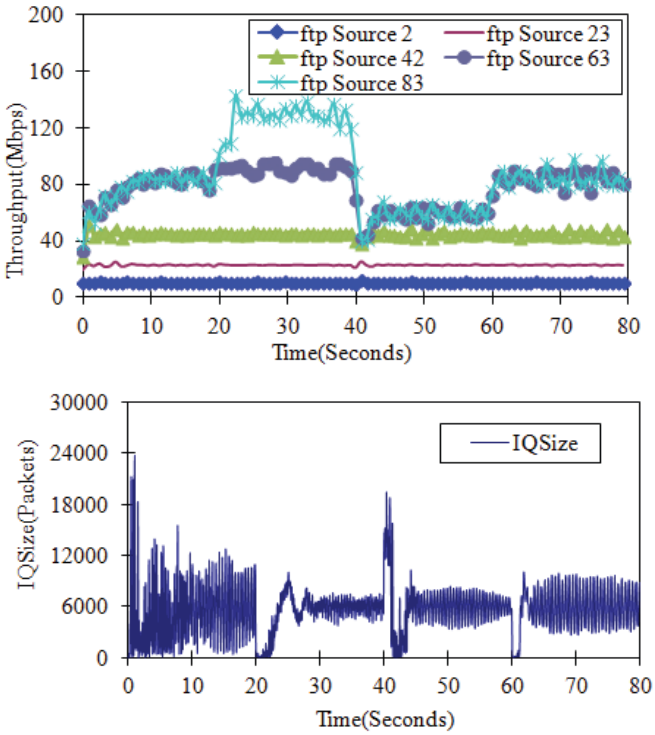


Figure: Bandwidth variations.

b) Sudden Bandwidth Changes

Since a deterministic amount of bandwidth is usually not available in contention-based multi-access networks or wireless networks, their bottleneck bandwidth can change suddenly. To demonstrate how our IntelRate controller can be robust to link bandwidth dynamics, we conduct an experiment with a bandwidth of $c(t)$ = 5Gbps, q_0 = 6000 packets and B = 60000 packets in the bottleneck. The source desired rates of ftp groups 1 to 5 are 11.47Mbps, 22.94Mbps, 45.88Mbps, 91.75Mbps and 131.07 Mbps, respectively



Source throughput and IQSize dynamics.

C. Queueing Jitter Control

One main source of the network latency oscillations comes from the dynamics of queuing delay in the routers. In most cases, whenever the queue size is out of control or oscillating too much, the queuing jitter performance degrades. In this experiment, we want to check under heavy traffic conditions how the queuing delay fluctuates under the different TBOs in the IntelRate controller, and how big the queuing jitters can be. The experiment is conducted in a 1Gbps bottleneck with B =10000 packets

D. Effect of Short-lived Traffic

So far, our discussion above mainly focuses on the long-lived ftp flows. As shown in Fig. 7, short-lived http flows have a sporadic arrival (following a think-time) usually with a small transfer rate. Their existence is usually negligible compared to the rate of ftp flows which is in the order of Mbps in our experiments. So even in the worst case when 100 http flows are transferring data at the same time at a rate of 200kbps and for a duration time of 1ms, they only consume 2Mbps of the bandwidth in a split second.

E. Utilization and Packet Loss Rate

We evaluate the utilization and packet loss rate performance of the IntelRate controller with respect to bottleneck bandwidth or the different settings of TBOs. First we check the system utilization and packet loss rate under the different bottleneck bandwidth from 45Mbps to 10Gbps. The simulation results show that the IntelRate controller is able to maintain the ideal zero packet loss rate with 100% link utilization despite the different bottlenecks (here we omit the plots due to space limit). The reason of the zero packet loss

is that the IntelRate controller can always control the variations of the IQSize around the TBO position. Therefore, the buffer never overflows and packets are never lost upon heavy traffic. In the meanwhile, the stable feature in IQSize and throughput guarantees the full bandwidth utilization.

F. Discussion

The good performances above demonstrated by the IntelRate controller also justify the rationale , the experiments under different link bandwidths above show that a choice of the TBO value works well in terms of the throughput performance and queuing delay.

On the other hand, the max-min fairness has shown that the IntelRate controller can guarantee the maximum output according to the biggest rate recorded in *req_rate* among all passing flows. This verifies that the controller meets our design objective of choosing the outmost edge value *D* of the output.

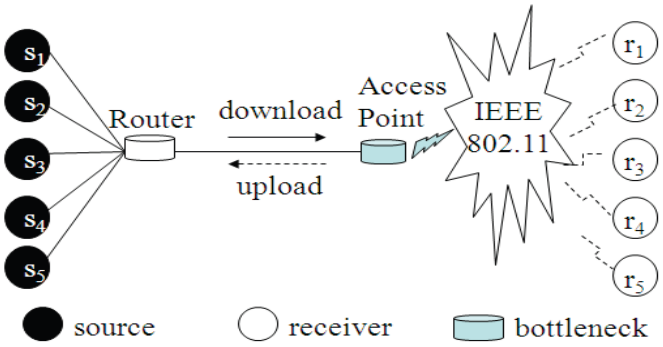


Figure: Wireless LAN.

6. Comparison

Our preliminary results in [53], [54] demonstrate that the IntelRate controller is superior to other explicit congestion controllers. For examples, the IntelRate controller has better robustness and link utilization than the XCP upon bandwidth variations; it has a lower requirement on computational and memory resources than API-RCP while having equivalent and even better performances (the comparisons of the computational intensity and memory requirement with other controllers will be presented in our other papers). Here we pick the QFCP and Blind (we didn't choose ErrorS or MAC from the same paper to do the comparison because ErrorS is an interchangeable algorithm of Blind and faces the same problem while MAC is too complicated to be put into practice so far) for further comparison, We use the same IEEE 802.11 wireless LAN (Local Area Network) as done in QFCP and Blind to do the comparison.

As shown in Fig. 14, the wireless LAN consists of 5 source destination flows (i.e., *si*−*ri*, *i* = 1, 2. . . 5). The bandwidth between *si* and the router is 100Mbps. The backhaul has 1Gbps bandwidth with 100ms propagation delay. The nominal bandwidth of the wireless is 11Mbps which needs to be probed by the controllers. In such a network, the wireless interface of the AP (Access Point) is the bottleneck when traffic flows from the wired network to the wireless network. The congestion controller resides in the AP to prevent the congestion. of each flow greedy by setting its desired sending rate to infinity. The parameters of QFCP and Blind controllers in the simulation are the same as those used in fig respectively. The buffer size *B* is set to 600 packets in all the controllers.

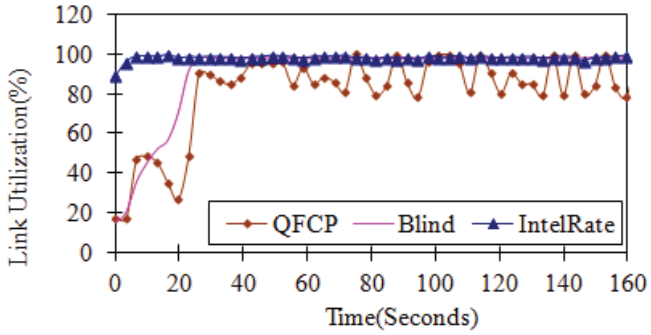


Figure: Link utilization.

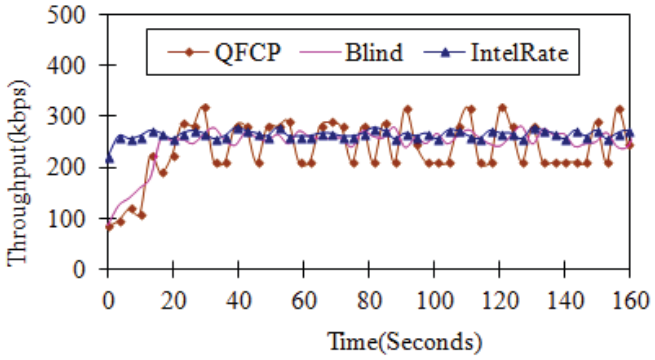


Figure: Source throughput

7. Future Scope

Any technique that provides zero packet loss can be recommended in the future work. In the future any techniques that minimizes the packet loss will be will be compatible, here this techniques also helps in minimising propagation delay for fast transmission without packet loss.

8. Conclusion

A novel traffic management scheme, called the IntelRate controller, has been proposed to manage the Internet congestion in order to assure the quality of service for different service applications. The controller is designed by paying attention to the disadvantages as well as the advantages of the existing 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. Unlike the existing explicit traffic control protocols that potentially suffer from performance problems or high router resource consumption due to the estimation ofthe network parameters, the IntelRate controller can overcome those fundamental deficiencies. To verify the effectiveness and superiority of the IntelRate controller, extensive experiments have been conducted in OPNET modeller. In addition to the feature of the FLC being able to intelligently tackle the nonlinearity 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 SIGCOMM*, 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

Author Profile

G .Ravi received his B.Tech degree from JNTU HYDERABAD and M. tech. degree from Pondicherry University and presently working in MRCET College as Assistant professor in Computer science Department.

Rahul Kumar his B.E degree in computer science and engineering from VTU University Belguam Karnataka, India