Experimental Investigation of Two Important Parameters of the IntelRate Controller for High-Speed Networks

Jungang Liu* and Oliver W.W. Yang

EECS, University of Ottawa, Canada
*Corresponding Author

jliu115@uottawa.ca, yang@eecs.uottawa.ca

Abstract

By relying only on the router IQSize (Instantaneous Queue Size), the IntelRate controller has shown better performances than the existing explicit congestion control protocols. It fundamentally overcomes the defects of the existing protocols that can potentially misestimate network parameters and consume too much router computation resources. However, researchers are also interested in the design of certain controller parameters and the choice of their values because they can affect the performance of the controller. In this paper, we will experimentally investigate in details the effects of two important parameters of the fuzzy logic controller: the width of membership functions and the number of linguistic variables. Using extensive OPNET simulations, we show the performance tradeoffs when choosing values for these two parameters.

Keywords*: Fuzzy Logic Control, Network Congestion, Time Response, OPNET Modeler*

1. Introduction

The implicit congestion control protocol, TCP Reno [1], is a widely deployed TCP (Transmission Control Protocol) version that makes internetworking sustainable, scalable and robust in present Internet. It has the important feature that the source adjusts its window size based on the notion that the network is a black box [2], *i.e.*, without knowing much information about the network such as link bandwidth and network traffic loads. All it needs is just the packet loss signal, upon which it decreases transmission rate to the half of its current congestion window [3]. Nevertheless, as mentioned in many papers, the TCP mechanism causes severe periodic oscillations in the window size and it even induces chaotic behavior into the network, thus adversely affecting overall network performance such as utilization, fairness and stability [4, 5].

In order to overcome the drawbacks of the implicit protocol TCP, a class of control protocols that can signal the network congestion level more explicitly were proposed. Examples are the XCP (eXplicit Control Protocol) [6], RCP (Rate Control Protocol) [7], API-RCP (Adaptive Proportional-Integral Rate Control Protocol) [8], JetMax [9] and MaxNet [10]. They record link information in a dedicated field of packet header and feed it back to a source so that the bandwidth could be efficiently utilized. Specifically, JetMax, MaxNet and utility function-based method signal the required fair rate or link price back to the source, but the final sending rate is decided by the source itself. XCP feedbacks the required increment or decrement of the sending rate, while RCP and API-RCP directly signal the admissible sending rate to the sources. The advantages of these protocols are that they provide the explicit information to signal the dynamic traffic levels in the link without keeping per-flow

state. In addition, the fully explicit congestion control may allow the design of a fair, stable, low loss, low delay and high utilization network [11].

However, the good performance of these existing explicit protocols requires an accurate estimation of network parameters such as the network bandwidth and the number of flows in the router. The parameter estimation in these protocols can basically cause the following two problems. One is the parameter mis-estimation due to the highly dynamics of the network and some inherent features of the networks such as intention or interferences [12]. The congestion control performance such as stability will be degraded upon mis-estimation in the existing protocols. The other problem is the parameter estimation consumes router computation resources such as CPU intensity and memory.

The fuzzy logic-based IntelRate controller [13, 14] has been proposed to address the abovementioned shortcomings of the existing explicit congestion control protocols, and it avoids these problems by only depending on the router IQSize (Instantaneous Queue Size) which can be accurately measured to gauge the traffic level in a bottleneck link.

In the design, the IntelRate controller resorts to the intelligent control approach which well tackles non-linear systems, instead of the traditional PI (Proportional and Integral) control which can only deal with linear system models. Specifically, the IntelRate controller combines the heuristic feature of the FLC (Fuzzy Logic Control) [15] and some prudent design ideas conceived from FLC elements. Performance evaluation [13] has shown that the IntelRate controller can achieve better performances in link utilization, throughput convergence, packet loss rate and robustness upon network changes when compared with the existing protocols. Various design parameters of the controller such as TBO (Target Buffer Occupancy) and bandwidth capacity have been well researched along with the experiments in [13, 14]. However, two parameters in the IntelRate controller design remain to be further investigated. They are the width of the MFs (Membership Functions) parameter m and the number of the LVs (Linguistic Variables) parameter N. They have been chosen as the basic design parameters after some investigations. Since researchers/readers have shown interest in their effect on the performance of the IntelRate controller, we shall present our investigations. As a contribution, to the best of our knowledge, we have not found other FLC-based congestion control schemes have investigated the effect of these two parameters on the congestion control performance.

In this paper, we will use the powerful network simulation tool, i.e., the OPNET (Optimized Network Engineering Tools) modeler [16], to experimentally investigate the effect of the parameters m and N on the IntelRate controller performance. The OPNET provides a large collection of detailed high-fidelity simulation models for different network technologies, protocols, and applications. To do our investigation, a single-bottleneck network model will be illustrated in the text and used to show the controller performance under different values of the design parameters. Through the extensive experiments, we shall demonstrate how the design parameter choices are made based on the performance trade-offs.

The paper is organized as follows. After a brief introduction in Section 2 about the IntelRate controller, the network simulation setup will be discussed in Section 3, based on which the investigations on the parameter m and N will be conducted in Section 4 and 5. With Section 6, we conclude the paper.

2. The IntelRate Controller

We shall summarize the IntelRate controller design below. The interested readers can refer to [13] for the design details and to [15] for more fundamental FLC theory.

Table 1 is one design of our controller rule base using N=9 LVs (Linguistic Variables) for the crisp inputs $e(t)$, $g(e(t))$. and output $u(t)$ as defined in [13]. The rules between the bold

lines in Table 1 reflects the rules that the IntelRate controller operates on, in which $e(t)$ can take on LVs "NS", "ZR" and "PS" only, while $g(e(t))$ can evolve from NV to PV (or vice versa) covering all the linguistic values. Therefore, the LVs of $u(t)$ is able to vary from ZR to MX (or vice versa) according to different combinations of $e(t)$ and $g(e(t))$.

Table 1. Rule table for the IntelRate controller

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

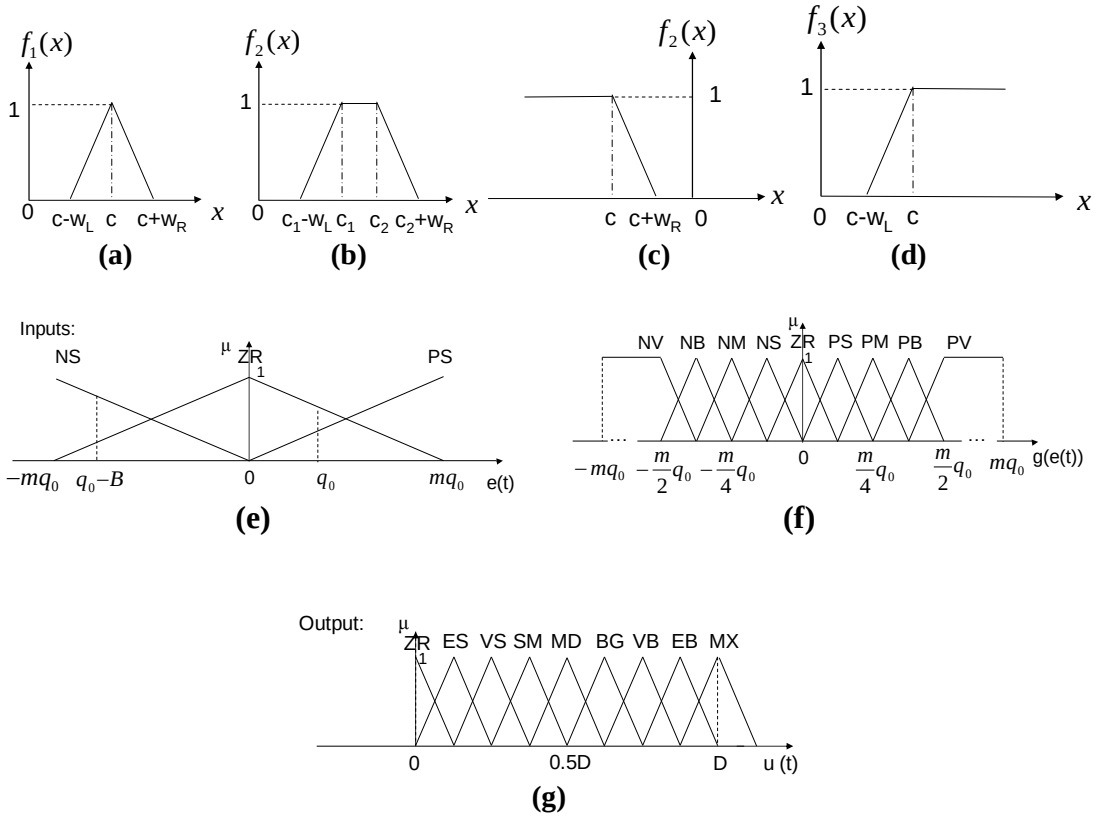


Figure 1. Membership Functions

The IntelRate controller employs the above mentioned isosceles triangular and trapezoid-like functions as MFs (Membership Functions), which are depicted in Figures 1(e) to 1(g). Figure 1(a) depicts a triangular MF (Membership Function) given by

$$f_1(x,c,w_L,w_R)=\begin{cases}\dfrac{x-c}{w_L}+1 & c-w_L<x\le c\\ \dfrac{c-x}{w_R}+1 & c<x\le c+w_R\\ 0 & Otherwise\end{cases} \quad (1)$$

where c is the centroid of triangle (*i.e.*, the center point of the bottom of a triangle); w_L (w_R) is the left (right) width of the triangular function.

Figure 1(b) depicts a trapezoidal MF given by the following equation.

$$f_2(x,c_1,c_2,w_L,w_R)=\begin{cases}\dfrac{x-c_1}{w_L}+1 & c_1-w_L<x\le c_1\\ 1 & c_1<x\le c_2\\ \dfrac{c_2-x}{w_R}+1 & c_2<x\le c_2+w_R\\ 0 & Otherwise\end{cases} \quad (2)$$

where c_1 and c_2 are shown in Figure 1(b), w_L (w_R) is the left (right) width of the trapezoidal function.

Since we just use trapezoid-like membership functions as show in Figure 1(c) above, their equations are as follows.

$$f_2(x,c,w_R)=\begin{cases}\dfrac{c-x}{w_R}+1 & c<x\le c+w_R\\ 1 & x\le c\\ 0 & Otherwise\end{cases} \quad (3)$$

where c is shown in Figure 1(c), and w_R is the distance between (c, c+ w_R).

$$f_3(x,c,w_L)=\begin{cases}\dfrac{x-c_1}{w_L}+1 & c-w_L<x\le c\\ 1 & x\ge c\\ 0 & Otherwise\end{cases} \quad (4)$$

where c is shown in Figure 1(d), and $w_{:L}$ is the distance between (c- w_L , c).

From Equations (1) and (2), the membership functions (N=9) for an input $e(t)$ in Figure 1 can be expressed as follows:

$$\mu_{NS}(e(t))=f_1(e(t),NS_0,NSw_L,NSw_R); \quad (5)$$

$$\mu_{ZR}(e(t)) = f_1(e(t), ZR_0, ZRw_L, ZRw_R); \quad (6)$$

$$\mu_{PS}(e(t)) = f_1(e(t), PS_0, PSw_L, PSw_R), \quad (7)$$

where the parameters in $f_1(.)$ have the same notions as in Equation (1), and $\mu_{NS}(e(t))$, $\mu_{ZR}(e(t))$ and $\mu_{PS}(e(t))$ are respectively the certainty degrees $\mu_{P_1^4}(p_1)$, $\mu_{P_1^5}(p_1)$ and $\mu_{P_1^6}(p_1)$.

For an input *g(e(t))*, the MFs are:

$$\mu_{NV}(g(e(t))) = f_2(g(e(t)), NV_0, NVw_R); \quad (8)$$

$$\mu_{NL}(g(e(t))) = f_1(g(e(t)), NL_0, NLw_L, NLw_R); \quad (9)$$

$$\mu_{NM}(g(e(t))) = f_1(g(e(t)), NM_0, NMw_L, NMw_R); \quad (10)$$

$$\mu_{NS}(g(e(t))) = f_1(g(e(t)), NS_0, NSw_L, NSw_R); \quad (11)$$

$$\mu_{ZR}(g(e(t))) = f_1(g(e(t)), ZR_0, ZRw_L, ZRw_R); \quad (12)$$

$$\mu_{PS}(g(e(t))) = f_1(g(e(t)), PS_0, PSw_L, PSw_R); \quad (13)$$

$$\mu_{PM}(g(e(t))) = f_1(g(e(t)), PM_0, PMw_L, PMw_R); \quad (14)$$

$$\mu_{PL}(g(e(t))) = f_1(g(e(t)), PL_0, PLw_L, PLw_R); \quad (15)$$

$$\mu_{PV}(g(e(t))) = f_3(g(e(t)), PV_0, PVw_L), \quad (16)$$

where the parameters in $f_1(.)$, $f_2(.)$ and $f_3(.)$ have the same notions as in Equation (1), (3) and (4), $\mu_{NV}(g(e(t)))$, $\mu_{NL}(g(e(t)))$, $\mu_{NM}(g(e(t)))$, $\mu_{NS}(g(e(t)))$, $\mu_{ZR}(g(e(t)))$, $\mu_{PS}(g(e(t)))$, $\mu_{PM}(g(e(t)))$, $\mu_{PL}(g(e(t)))$ and $\mu_{PV}(g(e(t)))$ are the certainty degrees $\mu_{P_1^j}(p_2)$ *j=1,2...N*, respectively.

Similarly, for an output *u(t)*, the MFs are:

$$\mu_{ZR}(u(t)) = f_1(u(t), ZR_0, ZRw_L, ZRw_R); \quad (17)$$

$$\mu_{ES}(u(t)) = f_1(u(t), ES_0, ESw_L, ESw_R); \quad (18)$$

$$\mu_{VS}(u(t)) = f_1(u(t), VS_0, VSw_L, VSw_R); \quad (19)$$

$$\mu_{SM}(u(t)) = f_1(u(t), SM_0, SMw_L, SMw_R); \quad (20)$$

$$\mu_{MD}(u(t)) = f_1(u(t), MD_0, MDw_L, MDw_R); \quad (21)$$

$$\mu_{BG}(u(t)) = f_1(u(t), BG_0, BGw_L, BGw_R); \quad (22)$$

$$\mu_{VB}(u(t)) = f_1(u(t), VB_0, VBw_L, VBw_R); \quad (23)$$

$$\mu_{EB}(u(t)) = f_1(u(t), EB_0, EBw_L, EBw_R); \quad (24)$$

$$\mu_{MX}(u(t)) = f_1(u(t), MX_0, MXw_L, MXw_R), \quad (25)$$

where $\mu_{ZR}(u(t))$, $\mu_{ES}(u(t))$, $\mu_{VS}(u(t))$, $\mu_{SM}(u(t))$, $\mu_{MD}(u(t))$, $\mu_{BG}(u(t))$, $\mu_{VB}(u(t))$, $\mu_{EB}(u(t))$ and $\mu_{MX}(u(t))$ are the certainty degrees $\mu_{U^j}(z)$ $j=1,2...N$, respectively.

As an improvement to [13], the width of MFs in Figure 1(e) and Figure 1(f) for the inputs $e(t)$ and $g(e(t))$ is designed by introducing a parameter m ($m \geq 1$). The purpose is to have a smaller TBO (*i.e.*, q_0) design such that the queueing delay will not degrade the network performance under heavy traffic. The dashed lines in Figure 1 denote the boundaries of the inputs or output, *e.g.*, the boundary of $e(t)$ satisfies $q_0\text{-}B \leq e(t) \leq q_0$ imposed by the physical limitations of a queue. The boundaries $\pm mq_0$ for $g(e(t))$ indicate its upper and lower limits in order to prevent $g(e(t))$ from increasing or decreasing infinitely towards positive and negative. The boundary D of the output is the maximum *Req_rate* (recorded in the congestion header [13]) among the incoming flows to the router.

As can be seen from the above summary, both the width parameter m of the MF and the number parameter N of the LVs can be important in their choices as they would affect the performance of the controller such as convergence speed or transient response. Fixed values (N=9 and m=8) were used in various investigations before to highlight the advantages of the IntelRate controller. Below, we shall conduct in-depth investigations about the choices of N and m.

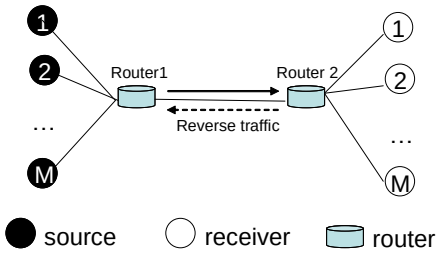


Figure 2. Simulation Network

3. Network Simulation Setup

With OPNET modeler [16], our experiments are conducted to investigate the controller performance in the most congested router, as shown in Figure 2 above. 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 marked on the nodes designate the subnets attached to each router.

Table 2. Sources Characteristics in Single Bottleneck Network

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

Table 2 summarizes the configuration of the M =11 subnet pairs which form the source-destination data flows in the network. They run various Internet applications such as the long-lived ftp, short-lived http, or the unresponsive UDP-like flows (also called uncontrolled ftp flows [8]). Since the link bandwidth we want to simulate has a magnitude of Giga bits per second, we use 20 flows in each subnet to generate enough traffic to produce congestion. All flows within each group are identical with the same RTPD and behavior, but different from 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 in the experiments. The reverse traffic is generated by the destinations when they piggyback the ACK (ACKnowledgment) information back to the sources.

The TBO and the buffer capacity B in Router 1 in each experiment are set according to the reasonable queueing delay as done in each experiment below. We also adopt some typical values from the experiments of existing works. In particular, all the ftp packets have the same size of 1024 bytes [8] while the http packet size is uniformly distributed in the range of [800, 1300] bytes [9].

In order to demonstrate and discuss the robustness of our IntelRate controller, our experiments would focus on the testing of the 100 long-lived ftp flows, unless otherwise stated. The 100 sporadic short-lived http flows will just act as the disturbance to the ftp traffic and their transfer size follows the real web traffic scenario; it has a Pareto distribution [17] with a mean transfer size of 30 packets [6]. The arrivals of http flows follow a think-time [17] uniformly distributed in [0.1s, 30s].

The experiments were conducted using OPNET Modeler 11.5 [16] on an Intel Core TM2 Quad platform with 2.40GHz processor. Typical simulated time is set according to the specific experiment scenario. The simulation time depends on bottleneck bandwidth and the simulated time. A typical simulation run usually takes hours or days. The number of packets generated in an experiment is related to the TBO value, the bandwidth, the simulated time and the traffic intensity. 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 [16]. Here, a bit is considered to be successfully sent out if it is part of a packet that has been successfully sent [16].

2) IQSize is the length of the bottleneck buffer queue (measured in packets) seen by a departing packet [18].

3) 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.

Since the behavior and performance of the sources within each group are quite similar, in the following experiments we shall show the results of one source from each group for brevity reason.

4. The Width Parameter *m*

We have conducted various experiments to study the effect of the width parameter *m*. In the first experiment, we use a link bandwidth of 4 Gbps, and set TBO=4000 packets (which will mean a queueing delay of about 8.19ms upon heavy traffic). The desired rates of the sources from ftp group 1 to group 5 are 13.11Mbps, 22.94Mbps, 32.77Mbps, 65.44Mbps and 72.09Mbps, respectively.

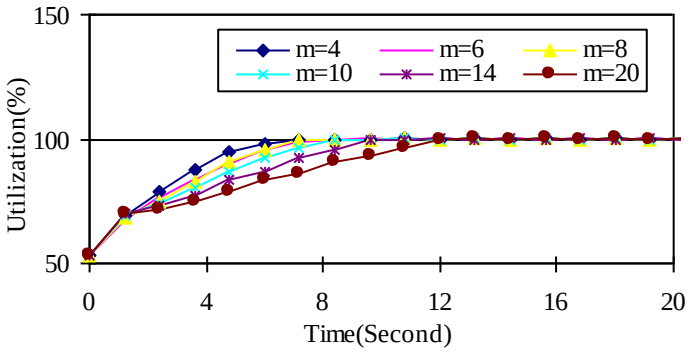


Figure 3. Link Utilization

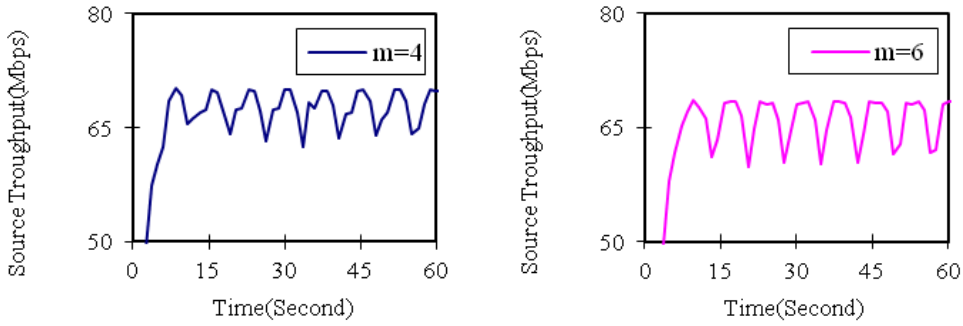
Figure 3 shows the time evolution of the link utilizations in the router for different *m* values. For example, when *m*=4, the utilization appears to approach the steady state value of 100% exponentially. The 2% settling time[1] *Ts* is measured to be 6.0s while the rise time[2] *Tp* is measured to be 4.3s for *m*=4. The behavior is similar as *m* increases but both the settling time and rise time become bigger. This is because too big of an *m* value leads to small partitions along $e(t)$ or $g(e(t))$ in the IntelRate controller, which brings about the long inference process of the fuzzy logic. Specifically, the settling time *Ts* with *m*=6, 8, 10, 14 and 20 are separately about 6.5s, 6.9s, 7.4s, 9.0s and 11.4s, respectively. In this regard, the relationship between *Ts* and *m* can be linearly approximated as $Ts=0.25m+5.3$. The rise time with *m*=6, 8, 10, 14 and 20 are about 4.8s, 5.2s, 5.6s, 6.5s and 8.2s. Similarly, the relationship between *Tp* and *m* can be linearly approximated as $Tp=0.2m+3.5$. Therefore, from the perspective of having a short rise time or settling time, smaller *m* is preferred.

Figure 4 shows the source throughput dynamics under different *m*. From Figure 4(a), the source rate increases approximately in exponential from its initial value, overshoots the steady state value and oscillates with an amplitude (trough or crest to the average) of about

[1] Settling time is the time required for the transient to reach and stay within ±2% of the steady state value [19].
[2] Rise time is the time required for the waveform to go from 10% to 90% of the final value [19].

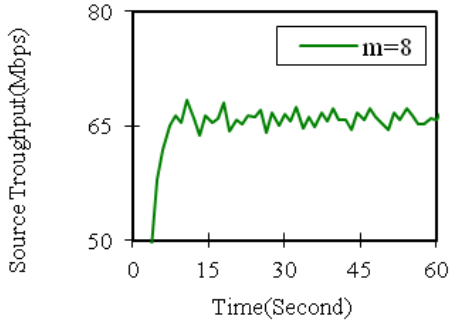
3.4Mbps after the transients. Figure 4(b) shows that the similar increase trend to Figure 4(a), and oscillates with an amplitude of about 3.8Mbps. From Figures 4(c) to 4f (*i.e.*, *m*=8, 10, 14, 20), the oscillations are about 1.8Mbps, 1.2Mbps, 1.6Mbps and 0.72Mbps, respectively (Note in Figure 4(f), big oscillation just happen occasionally). One can see the general trend is the oscillations become smaller with increasing *m*, but the oscillations does not decrease monotonically or linearly, instead the oscillations can become bigger when *m* increases, *e.g.*, the oscillations when *m*=14 is bigger than some of those when *m*=10. Therefore, from the perspective of the smoothness/oscillations of the source sending rate, a big *m* is preferred.

As for the transient response in Figure 4(a), the settling time *Ts* and the rise time *Tp* of the source are 6.6s and 4.8s respectively for *m*=4. Figure 4(b)-4(f) show that the settling time and the rising time, like the link utilization in Figure 3, becomes longer when *m* increases. The settling time we have determined from Figure 4(b) to 4(f) are about 7.8s, 8.2s, 8,4s, 9.6s and 12.0s, respectively, which can be roughly approximated as $Ts = 0.3m+5.4$. On the other hand, the rise times from Figures 4(b) to 4(f) are around 5.8s, 6.4s, 7.2s, 8.0s and 9.6s, respectively, which can be approximated as $Tp = 0.35m+3.5$. Therefore, like the utilization in Figure 3, from the perspective of a shorter settling time and rise time, a smaller *m* is preferred.

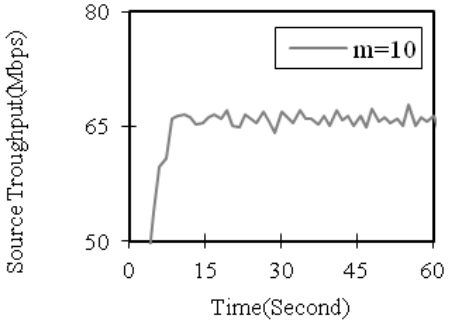


(a)

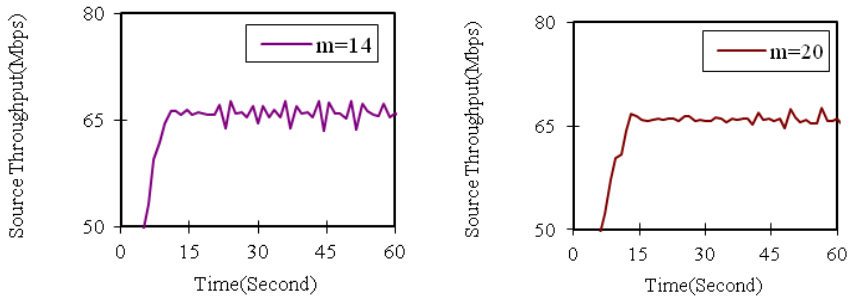
(b)



(c)



(d)

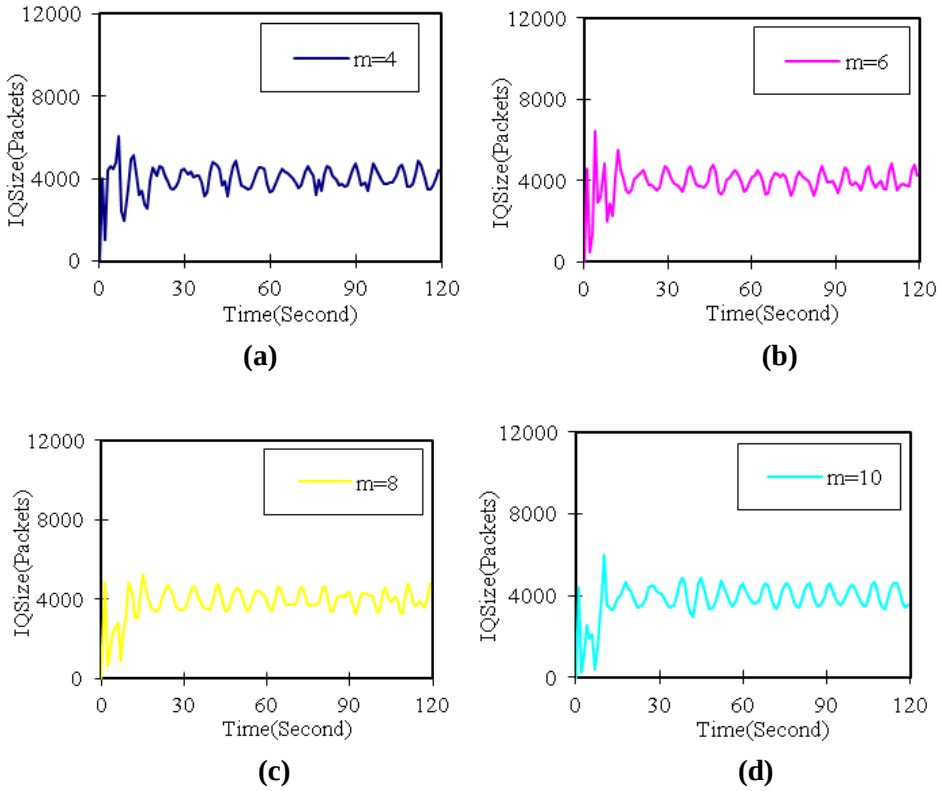


(e) **(f)**

Figure 4. The Source Throughput Dynamics

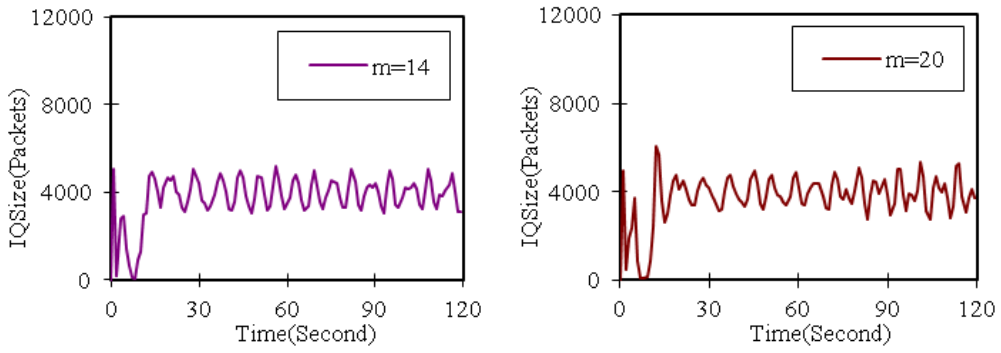
As a summary from above observations, either a too small *m* or a too big *m* would not make the IntelRate controller work well if we would like to have both good transient response (*i.e.*, setting time or rise time) and smooth source throughput. Instead, it should be a trade-off of the two performances. Considering such a trade-off, we would like to choose the middle, *i.e.*, *m*=8.

Finally, recall that the IQSize is the only variable that the IntelRate controller relies on to do its control. It would be of interest to find out if IQSize performance gives any information about the determination of *m*. Note that we have shown from numerous previous experiments that the IQSize can always be controlled around the TBO. So the interests would be to see how the changes of *m* may affect the transient response and steady state of the queue size as shown below.



(a) **(b)**

(c) **(d)**



(e) **(f)**

Figure 5. The IQSize Dynamics

Figure 5(a)-(f) show the IQSize performances under the different *m* values. All of them show that after the transients, the IntelRate controller is able to well stabilize the queue size around TBO=4000 packets despite the value of *m*. However, the steady state error (*i.e.*, the variations of queue size around the TBO) may vary. For example, among Figure 5(a)-(f), the queue size with *m*=4 has the smallest steady state error (*i.e.*, the oscillations above or below the TBO), which are averagely about 400 packets. While the ones with *m*=20 have the biggest steady state errors, which are averagely 960 packets. The steady state errors of the queue size with *m*=6, *m*=8, *m*=10 and *m*=14 are in between, *i.e.*, 430 packets, 500 packets, 620 packets and 780 packets. Similar to the transients (*i.e.*, the settling time or rise time), the relationship between the steady state error and *m* may be approximated with 30*m*+280. With such an observation, it may allow us to draw a general conclusion that small *m* tends to give small steady state error.

Furthermore, qualitatively, the settling time or the rise time is also shorter with a small *m* than a big *m* in that the IntelRate controller spends more time in transient before the queue size reaches the TBO, as seen in Figure 5(a)-(f).

We have also conducted similar experiments under different link bandwidth scenarios (*e.g.*, 1Gbps, 1.62Gbps, 3Gbps and 4Gbps), and obtained very similar trends and therefore the same conclusion. They are omitted to present here for brevity.

To summarize, a good queue size performance requires a small *m*. However, as discussed above, such a requirement also has to be compromised with the performance requirement of the source sending rate (which requires a big *m* for smoothness). The above decision that *m*=8 is also acceptable to the queue size performance, as shown in Figure 5(c).

5. The Number Parameter *N*

The parameter *N* is the number of LVs employed by the IntelRate controller. To determine this parameter, couple experiments are conducted under different link bandwidth. In the first experiment, with the 1Gbps link, TBO is set to be 1000 packets. The desired rates of the sources from ftp group 1 to group 5 are 3.28Mbps, 5.74Mbps, 8.19Mbps, 16.36Mbps and 18.02Mbps, respectively. We test the IntelRate controller with *N*=5, 7, 9, 11, 13 and 15 and compare their transient response and steady state performances.

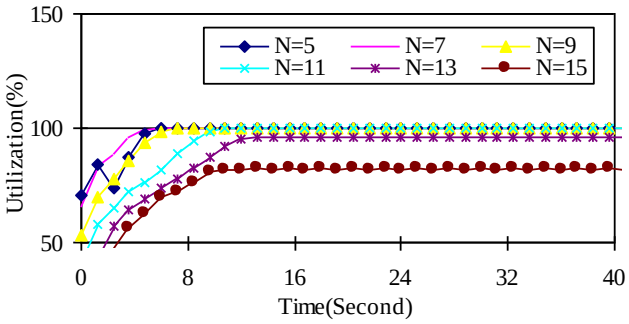
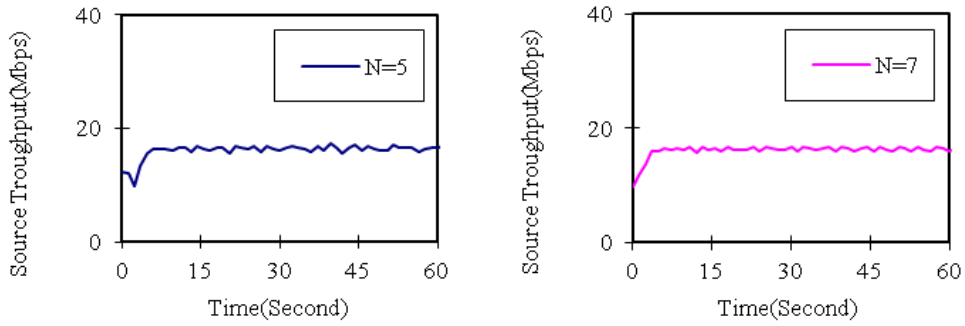


Figure 6. Link Utilization

The link utilization under different *N* values is shown in Figure 6. When *N*=5, the settling time *Ts* and the rise time *Tp* are 7.2s and 5.7s, respectively. We also notice that there is a "kink" during the rise time, which we suspect to be the queue size increasing too fast (also observed in Figure 8(a) to be discussed later) at the beginning and the controller therefore decreases the source sending rate a bit aggressively. Figure 6 shows that when *N* increases from 5 to 15, the rise time of the link utilization becomes longer because the controller spends more time before reaching the steady state. Specifically, the settling time from *N*=7 to 15 is separately 6.8s, 7.2s, 9.6s, 11.0s and 9.6s, Furthermore, when *N* increases to 13 or 15, the link can only reach a utilization of 95% or 82%, instead of 100%, which means the big *N* can cause link under-utilization problem. Therefore, the settling time for the last one may not be good result since it is not based on 100% utilization or close to 100% utilization. Besides, as one can see from the settling times, the relationship between *Ts* and *N* cannot simply be linearly approximated. As for the rise time, when *N* increases from 7 to 15, they are separately 5.6s, 5.8s, 8.0s, 9.3s and 7.6s. The above comment for *Ts* can be similarly applied to *Tp* here, too, *i.e.*, it seems a linear relationship cannot be used to describe *Tp* and *N*. However, the general trend of the transients shows that in order to have fast transient response and full link utilization, one would like to choose a small *N*.



(a) **(b)**

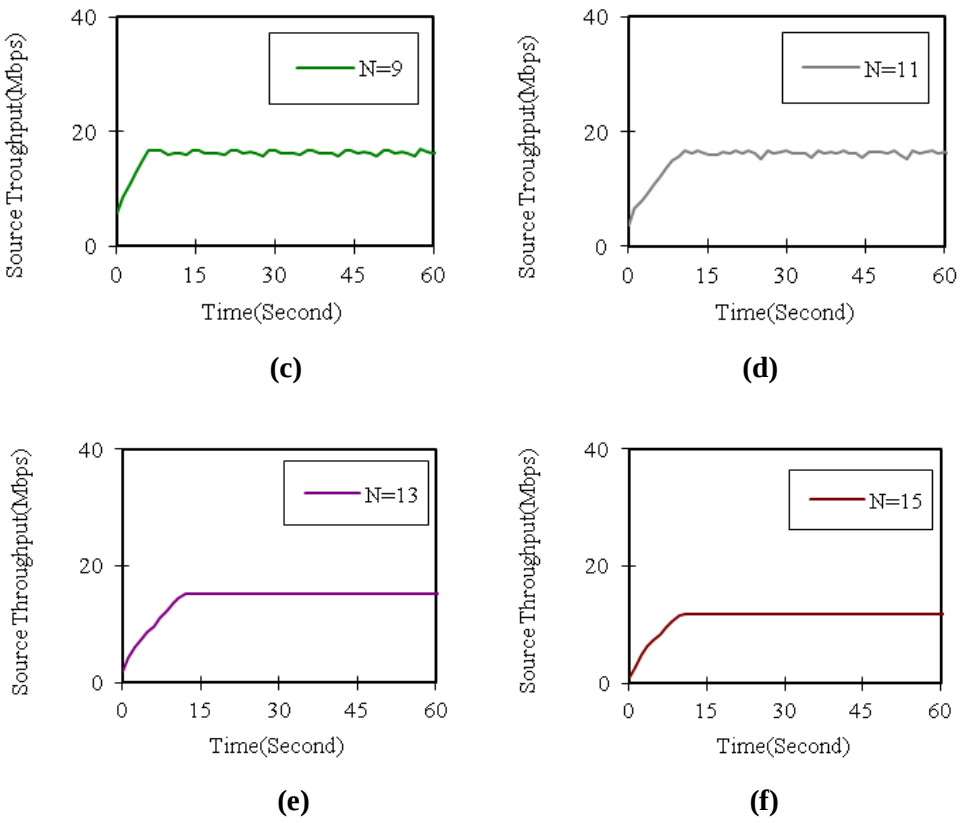


Figure 7. The Source Throughput Dynamics

The source throughput is depicted in Figure 7 with the different *N*. On one hand, like the link utilization performance in Figure 6, the bigger *N* leads to longer settling time or rise time, as seen from Figure 7(a)-(f). Specifically, the setting times from *N*=5 to 15 are 7.2s, 8.4s, 9.0s, 10.2s, 11.5s and 9.6s respectively. For Figure 7(f), since the link is under-utilized (as shown in Figure 6(f)), the sending rate is lower, too, as expected than others in Figure 7(a)-(e), On the other hand, although the smoothness shown from Figure 7(a)-(f) is all acceptable in terms of the slight difference on their oscillations, it still indicates a big *N* gives better smoothness, which is evident when we compare Figure 7(e) or Figure 7(f) with Figure 7(a)-(d). However, the caution must be exercised when we prefer big *N* because big *N* complicates the controller in the sense that the controller has to do more logic computations to produce a control command. Therefore, from the viewpoint of the source performance shown in Figure 7, *N* can be a value to trade off the transients and the smoothness. In this regards, we would like to choose *N*=9. Furthermore, the following queue size performance also renders such a requirement for the choice of *N*.

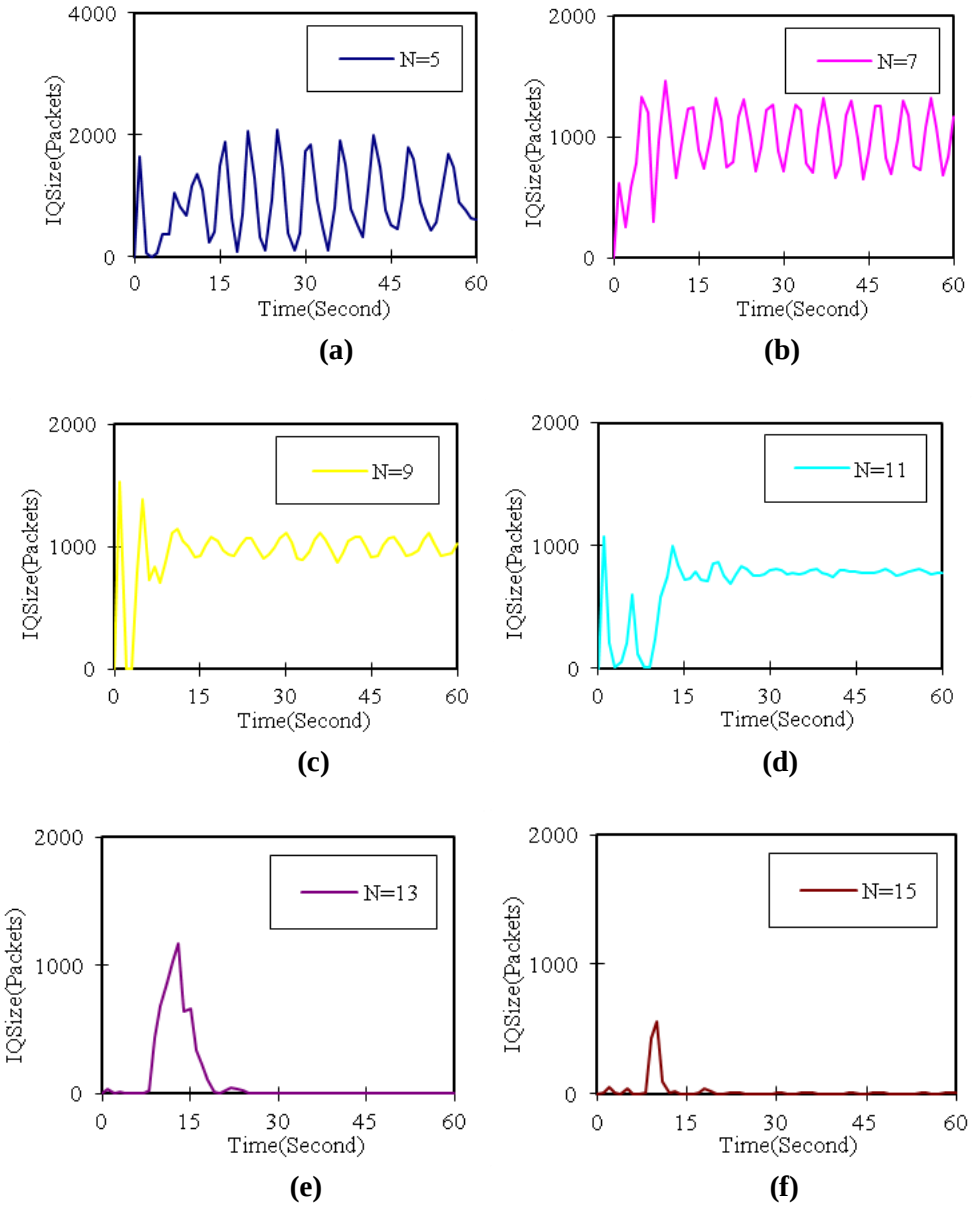


Figure 8. The IQSize Dynamics

The queue size performance under different *N* is depicted in Figure 8(a)-(f). In Figure 8(a), in the first 3.6s the queue size oscillates as high as 1600 packets in the transient stage. When the controller reaches the steady state, the steady state error is 980 packets in average. When *N* becomes bigger, the oscillations become smaller from Figures 8(b) to 8(f). In detail, the steady state error from Figure 8b-d (*i.e.*, *N*=7 to *N*=11) is separately 320 packets, 78 packets and 43 packets. For Figure 8(e) (*N*=13) and Figure 8f (*N*=15), the queue does not reach TBO (*i.e.*, 1000 packets), instead it can only stay close to the empty position, which means a big *N* can affect the queue performance in terms of the ability to control IQSize to TBO. The reason is that the source sending rate decreases (*e.g.*, as seen in Figure 7(f)) with a big *N*, so there is not enough traffic to support the queue level. In a nutshell, as chosen for *N* above, *N*=9 can

also a good trade-off between the queue steady state error and the ability to control queue size to TBO.

The similar experiments under different link bandwidth scenarios (*e.g.*, 5Gbps) shows the same trend in their results as shown in Figure 6, Figure 7 and Figure 8. For brevity, they are not shown here.

As a summary to the controller performances, the choice of N (*i.e.*, we choose N=9) is made as a trade-off by considering the link utilization performance, the source throughput performance, the queue size performance as well as the computation complexity.

6. Conclusion

The IntelRate controller need not estimate the network parameters and fundamentally overcomes the instability issues caused by parameter mis-estimations in other explicit congestion control protocols while saving the computational resources in a router. In this paper, we have investigated further the effects of two important design parameters (*i.e.*, the width of MFs and the number of LVs) on the IntelRate controller performance with the extensive experiments. We concluded that the choices of these design parameter should be trade-offs of the related control performances.

The IntelRate controller is so far tested via software simulations only. One urgent future work is to set up real networks from small scales to large scales in our lab to practically test the controller and figure out the deployment issue.

Acknowledgement

This work is supported by the Research Discovery Grant (#RGPIN42878) as well as the Accelerated Grant from NSERC.

References

[1] V. Jacobson, “Modified TCP congestion avoidance algorithm”, mailing list, end-to-end-interest, **(1990)** April 30.
[2] K. K. Ramakrishnan and S. Floyd, “Proposals to add explicit congestion Notification(ECN) to IP”, RFC 2481, **(1999)** January.
[3] V. Jacobson and M. J. Karels, “Congestion avoidance and control”, Proc. of ACM SIGCOMM Computer Communication, **(1988)** August, pp. 314-329.
[4] W. Feng and S. Vanichpun, “Enabling compatibility between TCP Reno and TCP Vegas”, Proceed. Symposium on Applications and the Internet, **(2003)**, pp. 301-308.
[5] M. M. Hassani and R. Berangi, “An analytical model for evaluating utilization of TCP Reno”, Proc.of International conference on computer systems and technologies, Bulgaria, **(2007)**, pp. IIIB 14-17.
[6] D. Katabi, M. Handley and C. Rohrs, “Congestion control for high bandwidth-delay product networks”, Proc. of SIGCOMM 2002, **(2002)**, pp. 89-102.
[7] N. Dukkipati, N. McKeown and A. G. Fraser, “RCP-AC congestion control to make flows complete quickly in any environment”, Proc. of INFOCOM 2006, **(2006)** April, pp. 1-5.
[8] Y. Hong and O. W. W. Yang, “Design of adaptive PI rate controller for best-effort traffic in the Internet based on phase margin”, IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 4, **(2007)** April, pp. 550-561.
[9] Y. Zhang, D. Leonard and D. Loguinov, “JetMax: scalable max-min congestion control for high-Speed heterogeneous networks”, Proc. of INFOCOM, **(2006)** April, pp. 1-13.
[10] B. Wydrowski, L. L. H. Andrew and M. Zukerman, "MaxNet: a congestion control architecture for scalable networks", IEEE Communications Letters, vol. 7, no. 10, **(2003)** October, pp. 511–513.
[11] F. P. Kelly and G. Raina, “Explicit congestion control: charging, fairness and admission management”, Next-Generation Internet Architectures and Protocols, Cambridge University Press, **(2010)**.
[12] Y. Zhang and T. R. Henderson, “An implementation and experimental study of the explicit control protocol

(XCP)", Proc.of INFOCOM 2005, vol. 2, **(2005)** March, pp. 1037- 1048.
[13] J. Liu and O. Yang, "Design and evaluation of an intelligent controller for heterogeneous networks", IEEE Globecom 2010, **(2010)** December 6-10, pp. 1-5.
[14] J. Liu and O. Yang, "Stability Analysis and Evaluation of the IntelRate Controller for High-Speed Heterogeneous Networks", IEEE ICC 2010, **(2011)** June, pp. 1-5.
[15] K. M. Passino and S. Yurkovich, "Fuzzy Control", Addison Wesley Longman, Inc., **(1998)**.
[16] Opnet Modeler Manuals, Opnet version 11.5, Opnet Technologies Inc., **(2005)**.
[17] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence an possible causes", IEEE/ACM Transactions on Networking, vol. 5, no. 6, **(1997)** December, pp. 835-846.
[18] D. Gross, J. Shortle, J. Thompson, *et al.*, "Fundamentals of Queueing Theory", John Wiley & Sons Inc., 4th Edition, **(2008)**.
[19] R. C. Dorf and R. H. Bishop, "Modern Control System", Pearson Prentice Hall, 11th edition, **(2008)**.