International Journal of Innovative Computing, Information and Control
Volume **9**, Number **7**, July **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 2801–2819

A STABLE FUZZY LOGIC CONTROLLER USING THE LEAST PARAMETER FOR EXPLICIT TRAFFIC CONTROL

Jungang Liu and Oliver W. W. Yang

School of Electrical Engineering and Computer Science
University of Ottawa
800 King Edward Avenue, Ottawa, Ontario, K1N 6N5, Canada
jliu115@uottawa.ca; yang@eecs.uottawa.ca

Received May 2012; revised September 2012

Abstract. *The fuzzy-logic-based IntelRate controller is a newly proposed intelligent explicit traffic congestion controller that can avoid the evaluation of some critical network parameters such as bottleneck bandwidth or the number of flows in a router. It fundamentally eliminates the potential performance problems caused by parameter mis-estimations. Besides, it saves computation resources and improves the router efficiency. This paper theoretically and experimentally shows that the IntelRate control system is stable in whatever network traffic conditions while providing superior performance. It is the first time that an explicit rate-based congestion control system designed with the fuzzy logic control is proved globally asymptotically stable. Our simulations demonstrate that the IntelRate controller can have better performance than other exiting controllers upon network parameter changes.*
Keywords: Fuzzy logic control, Stability, Max-min fair, Congestion control

1. Introduction.

Network congestion control can prevent a network from severe degradation in throughput-delay performance. The IntelRate controller [1] is a fuzzy-logic-based explicit traffic congestion controller proposed to address the shortcomings in existing explicit congestion control protocols like XCP [2], RCP [3] and API-RCP [4-6]. In order to achieve good or optimum performance, the existing protocols require accurate network parameter values such as the link bandwidth and/or the number of flows in a router, which are often very hard to determine accurately. For example, the link bandwidth in the contention-based multi-access wired networks or IEEE 802.11 wireless networks is not deterministic due to link sharing, half-duplex or interference [7]. A wrong estimation can lead to poor performance.

By not relying on those network parameters, the IntelRate controller: (1) avoids large steady state error and instability issues potentially arising from the parameter mis-estimations; (2) saves precious router computation resources such as additional timers and mathematical operations; (3) does not require extra huge memory to store the per-flow address information in order to estimate the number of flows, which can number in millions every hour in a router [8]. Therefore, the IntelRate controller is more effective and superior when working for networks where network parameters are unpredictable or their computations consume much router resource or energy [9]. Our preliminary results (e.g., performance evaluation in [1]) have also shown that the IntelRate controller can achieve better performances in link utilization, throughput convergence and robustness. By incorporating max-min fairness and the TBO (Target Buffer Occupancy) in its design, the IntelRate controller has a very low queueing delay and zero packet loss performance.

Another performance aspect that needs to be investigated further is the stability of the IntelRate controller. Stability analysis of FLC (Fuzzy Logic Control) systems has been around since 1990s covering areas such as mass-spring-damper systems (e.g., [10,11]), electrical power systems (e.g., [12,13]), mechanical and motor control systems (e.g., [14,15]), robotic systems (e.g., [16,17]). These studies have used different models including the Takagi-Sugeno model (e.g., [18-20]), the sliding-mode control model (e.g., [21]) and the nonlinear strict-feedback model (e.g., [22]). These models can be applied to either the continuous-time or the discrete-time systems.

Different approaches have been used in the stability analysis of FLC systems. Lyapunov's stability criterion is the most frequently used approach (e.g., [10-12,15,20,21,23]) once the state equations of the plant are known. Other approaches include the describing function (e.g., [13]), circle criterion (e.g., [14]), Popov criterion and hyperstability criterion [24]. Each approach has its limitation and their applicability depends on the system structure and the type of information describing the plant. For example, the describing function technique can be applied to the Single (or Multiple)-Input- Single-Output fuzzy controllers, but is prohibitive for more than three inputs in the fuzzy controller [25] and for the systems that cannot be separated into a linear and a nonlinear part [24].

There are some FLC stability studies related to data network congestion control [26-28]. The RED (Radom Early Detection)-like controller design (e.g., [27]) is actually mixing FLC with the sliding mode control. Its proof of stability is based on linear control theory (i.e., Nyquist Criterion) by linearizing the plant model around the equilibrium point. LDM (Lyapunov's Direct Method) was used to obtain the system stability conditions of an AQM (Active Queue Management) control system (e.g., [26]), but the guarantee of stability was not proved. It is also not clear whether those conditions are eventually met with just three rules designed for their controller because a small number of rules can have many potential performance problems such as big overshoot, long settling time or instability due to the high dynamics of the Internet traffic. At least a 300% of overshoot and more than 20 seconds of settling time have been observed in their simulation results. Furthermore, the stability analysis in these existing works is only for window-based AQM congestion control protocols.

Unlike the above window-based protocols that provide implicit congestion information for TCP sources, the proposed IntelRate control system is rate-based and feeds the explicit congestion information back to sources. Some works using similar design rationale can be found in ATM (Asynchronous Transfer Mode) networks, e.g., [29-31], but unfortunately no stability analysis was provided.

Despite the advantages of the IntelRate controller, its stability study is of particular importance and interest. This is because the fuzzy-logic-based controller is inherently nonlinear, unlike other explicit congestion control protocols (such as XCP, RCP and API-RCP mentioned before) that are designed with the linear PI (Proportional-Integral) controller ideas based on established knowledge of classical control theory. The difficulty to prove the stability of an FLC system comes mainly from the rule base (which contains a logical quantification of heuristic information) that cannot be mathematically quantified. That is to say, it is very hard (if not impossible) to express a fuzzy-logic-based controller in an analytical form [24]. In addition, one cannot easily verify if the stability is guaranteed from the basic design process of a fuzzy logic controller.

In view of the above discussions, we are motivated to prove the stability for the fuzzy-logic-based IntelRate control system. Our objectives are to theoretically and experimentally investigate its global stability property. To achieve our objectives, we shall first use the LDM to analyze the stability of the nonlinear rate-based IntelRate control system in the time domain. Then we exhaustively show how the stability of the IntelRate control system is guaranteed in whatever network traffic conditions. Moreover, we shall use OPNET simulation to verify our theoretical analysis. The main contribution of this paper is, for the first time to our best knowledge, a fuzzy-logic-based control system is proved to be globally asymptotically stable for the rate-based explicit congestion control of data network traffics.

2. The IntelRate Controller.

We shall summarize the IntelRate controller operation below in order to review and introduce notations for our later analysis. The interested readers can refer to [1] for the design details, and to [25] for more fundamental FLC theory.

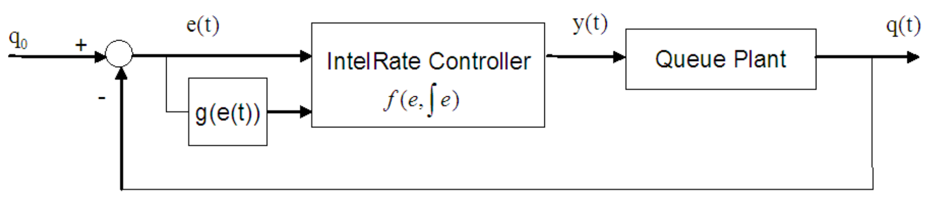


FIGURE 1. The closed-loop IntelRate control system for a single router network

Figure 1 depicts the closed-loop Single-Input-Single-Output IntelRate congestion control system. The variable $e(t) = q_0 - q(t)$ is the deviation of the router IQSize (Instantaneous Queue Size) $q(t)$ from the TBO q_0. In order to remove the steady state error, we choose $g(e(t)) = \int e(t)dt$ as the other input of the controller. Under heavy traffic situations, the IntelRate controller would compute an allowed sending rate $u_i(t)$ for each flow i according to $q(t)$ so that $q(t)$ is stabilized around q_0 and congestion is prevented. The aggregate output is $y(t) = \sum u_i(t - \tau_i)$, where τ_i is the round trip time. In this system, $q(t)$ is the only parameter we measure and feedback to the IntelRate control system as the congestion signal.

To model the queue plant, we consider the dynamics in both the aggregate controlled traffic $y(t)$ and the bottleneck bandwidth $c(t)$. Then the rate of change in $q(t)$ can be expressed as

$$\dot{q}(t) = \begin{cases} y(t) + v(t) - c(t) & q(t) > 0 \\ [y(t) + v(t) - c(t)]^+ & q(t) = 0 \end{cases} \tag{1}$$

where $[x]^+ = \max(0, x)$, and $v(t)$ is assumed to be the uncontrolled traffic.

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

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 [1]. In the following sections, we shall use the notation $(e(t), g(e(t)), u(t))$ to represent a rule in this table. The dashed box 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 input combinations of $e(t)$ and $g(e(t))$. The shaded part in Table 1 will be discussed in Section 3.3.

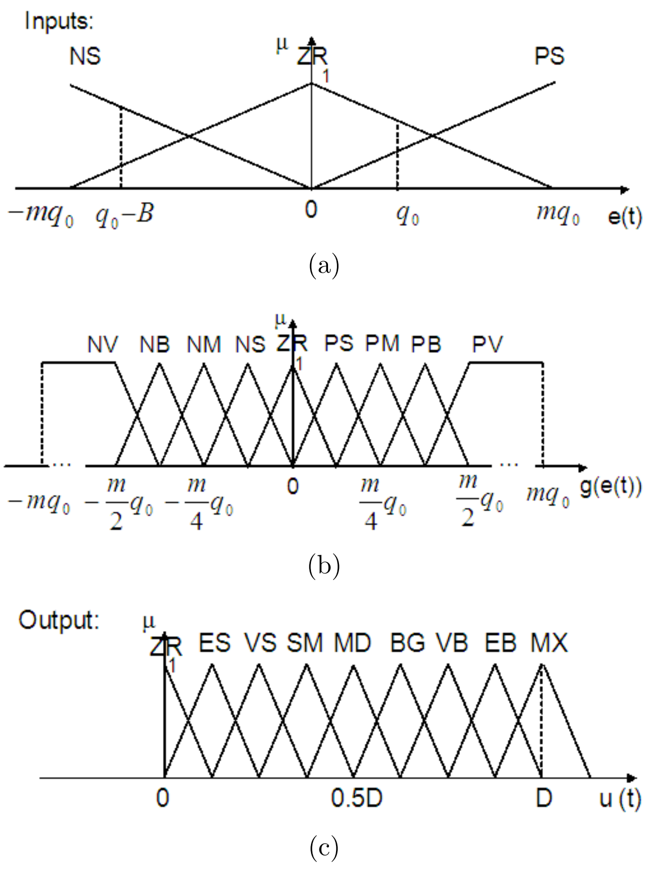


FIGURE 2. Membership functions

The IntelRate controller employs the isosceles triangular and trapezoid-like functions as MFs (Membership Functions), which are depicted in Figure 2. As an improvement over the work of [1], the width of MFs in Figure 2 for the inputs $e(t)$ and $g(e(t))$ has been redesigned by introducing a new 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 2 denote the boundaries of the inputs or output, e.g., the boundary of $e(t)$ satisfies $q_0 - B \leq e(t) \leq q_0$ 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 [1]) among the incoming flows to the router.

Below is a summary of the traffic-handling procedure of the IntelRate controller in a router [1] which has incorporated a max-min fairness design as demonstrated later.

(1) Upon the arrival of a packet, extract *Req_rate* from the congestion header of the packet.

(2) Sample $q(t)$ of the router and update $e(t)$ and $g(e(t))$.
(3) Compare the crisp output $u(t)$ with *Req_rate*.
(3a) If $u(t) <$ *Req_rate*, the link cannot accommodate the *Req_rate* amount of sending rate. So update *Req_rate* in the congestion header by $u(t)$.
(3b) Otherwise, leave the *Req_rate* filed unchanged.
(4) Whenever an operation cycle is over, update D and compute $u(t)$.

When the packet arrives at the destination, the receiver extracts *Req_rate* from the header and sends it back to the source via the acknowledgement packet.

3. Stability Analysis.

The IntelRate control system can be shown to be globally asymptotically stable. By reversely applying the LDM [32], we first derive what the controller is supposed to behave in order to meet the asymptotical stability conditions, and then show that the controller is behaving as expected. Before we proceed, we state the following assumption and proposition. Readers are referred to references like [24,32] for the concept and conditions of LDM.

3.1. The transformed system model.

We let $(y(t), q(t))$ be the state vector of the IntelRate control system described in Figure 1. The following assumption and proposition would allow us to establish the transformed system model and stability theorems later on.

***Assumption* 1**: *For a router using the IntelRate controller, the uncontrolled traffic $v(t)$ is far less than the dominant controlled traffic $y(t)$ and can be neglected by letting $v(t) = 0$.*

Comment: This assumption is practical because in the networks nowadays, the controlled TCP generates 90-95% of the Internet traffics [33], while all other uncontrolled traffics such as UDP make up the rest which is only a small portion of the Internet traffic.

***Proposition* 1**: *Consider the closed-loop unity feedback system in Figure 1 with a non-zero reference value q_0 (i.e., $q_0 \neq 0$). Let y_s be the steady state of the aggregate controller output (also the aggregate incoming traffic to a router), and q_s be the steady state of the IQSize. If the system is stable under heavy traffic conditions, then (y_s, q_s) would be a non-zero vector at steady state. Further, $q_s = q_0$, and $y_s = c(t)$ if the system steady-state error is negligible.*

Proof: This proposition is straightforward from the perspective of control theory. For the closed-loop system with unity feedback in Figure 1 to be stable as $t \to \infty$, its steady state output approaches the reference input if the steady state error can become negligible [34], i.e., $q(t) = q_s = q_0 \neq 0$. Since the inbound and the outbound traffic in the queue must strike a balance to achieve stability at steady state, we have y_s equals the link capacity $c(t)$, i.e., $y_s = c(t) \neq 0$. The proof is complete.

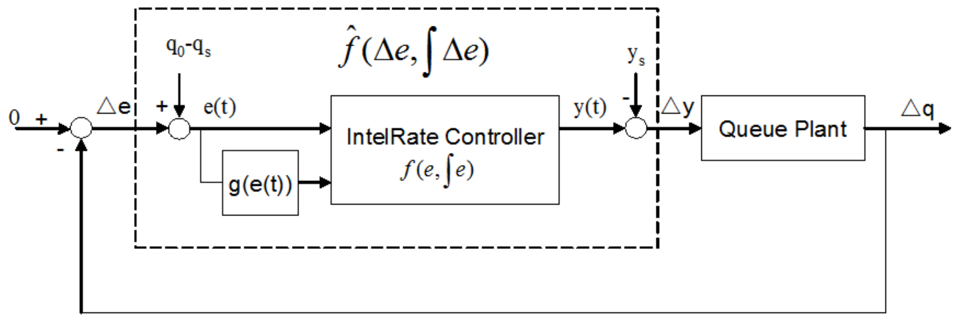


FIGURE 3. Transformed system model with steady state at zero

Since a non-zero steady state vector (y_s, q_s) does not meet the prerequisite of LDM for the stability analysis [32], we use the approach described in [24] to convert (y_s, q_s) into a zero vector. This is done by defining a new state vector $(\Delta y, \Delta q)$ which is the deviation of $(y(t), q(t))$ from (y_s, q_s). Figure 3 shows the transformed equivalent system of Figure 1.

3.2. **The stability conditions.** With the transformed system model in Figure 3, we can now find the conditions that guarantee the stability of the IntelRate control system.

Since $(\Delta y, \Delta q)$ is the deviation of the state vector $(y(t), q(t))$ from (y_s, q_s), we have

$$\begin{cases} \Delta y = y(t) - y_s \\ \Delta q = q(t) - q_s \end{cases} \tag{2}$$

and their derivatives are

$$\begin{cases} \dot{\Delta y} = \dot{y}(t) \\ \dot{\Delta q} = \dot{q}(t) \end{cases} \tag{3}$$

From Figure 3, the aggregate output equation of the IntelRate controller can be represented by $y(t) = f(e(t), g(e(t)))$. The change rate of $y(t)$ is thus $\dot{y}(t) = \frac{\partial f}{\partial e} \cdot \frac{de}{dt} + \frac{\partial f}{\partial g} \cdot \frac{dg}{dt}$. Since $g(e(t)) = \int e(t)dt$, $\dot{y}(t)$ can be rewritten as $\dot{y}(t) = \frac{\partial f}{\partial e} \cdot \dot{e}(t) + \frac{\partial f}{\partial g} \cdot e(t)$.

Since the queue deviation $e(t) = q_0 - q(t) = q_0 - (\Delta q + q_s)$, $e(t)$ is a function of Δq. Consequently,

$$\dot{e}(t) = -\Delta \dot{q} \tag{4}$$

(4) means $\dot{e}(t)$ is a function of $\Delta\dot{q}$. The right-hand side of $\dot{y}(t)$ is thus a function of Δq and $\Delta\dot{q}$. To represent the IntelRate controller state equation $\dot{y}(t)$ with Δq and $\Delta\dot{q}$, we define the right-hand side of $\dot{y}(t)$ to be a new function $h(\Delta q, \Delta\dot{q})$. Hence, $\dot{y}(t) = h(\Delta q, \Delta\dot{q})$.

From (1), the lower equation of (3) becomes $\dot{\Delta q} = \begin{cases} \Delta y + y_s + v(t) - c(t) & q(t) > 0 \\ [\Delta y + y_s + v(t) - c(t)]^+ & q(t) = 0 \end{cases}$. One can see that $\Delta\dot{q}$ is a function of Δy. Therefore, we have

$$\dot{y}(t) = h(\Delta q, \Delta y) \tag{5}$$

By carefully checking (5), the function $h(\Delta q, \Delta y)$ represents the action of the controller in response to the change of the aggregate arrival rate, i.e., $\dot{y}(t)$.

Till now, the system Equation (3) can be rewritten as

$$\begin{cases} \dot{\Delta y} = h(\Delta q, \Delta y) \\ \dot{\Delta q} = \begin{cases} \Delta y + y_s + v(t) - c(t) & q(t) > 0 \\ [\Delta y + y_s + v(t) - c(t)]^+ & q(t) = 0 \end{cases} \end{cases} \tag{6}$$

Using $\boldsymbol{x} = (x_1(t), x_2(t)) = (\Delta y, \Delta q)$, we obtain a more general form,

$$\begin{cases} \dot{x}_1 = h(x_1, x_2)) \\ \dot{x}_2 = \begin{cases} x_1 + y_s + v(t) - c(t) & q(t) > 0 \\ [x_1 + y_s + v(t) - c(t)]^+ & q(t) = 0 \end{cases} \end{cases} \tag{7}$$

Without loss of generality, we choose the positive scalar $V(\boldsymbol{x}) = (x_1^2(t) + x_2^2(t))/2$ as the Lyapunov function. To guarantee the system's stability, we must have $\dot{V}(\boldsymbol{x}) = x_1(t)\dot{x}_1(t) + x_2(t)\dot{x}_2(t) < 0$. Specifically, $x_1(t)h(x_1, x_2) + x_2(t)(x_1 + y_s + v(t) - c(t)) < 0$. Note that we have reasonably neglected the boundary conditions of $\dot{x}_2(t)$ in (7). Later on we shall show how the IntelRate control system designed with this inequality can maintain stability despite these boundary conditions. According to the LDM [32], the system would be asymptotically stable if

$$x_1(t)h(x_1, x_2) + x_2(t)(x_1(t) + y_s + v(t) - c(t)) < -\varepsilon \tag{8}$$

for any small $\varepsilon > 0$. Therefore, for $x_1(t) \neq 0$, $h(x_1, x_2)$ should satisfy

$$\begin{cases} h(x_1,x_2) > \frac{1}{x_1(t)}[x_2(t)(c(t) - x_1(t) - y_s - v(t)) - \varepsilon] & \text{if } x_1(t) < 0 \\ h(x_1,x_2) < \frac{1}{x_1(t)}[x_2(t)(c(t) - x_1(t) - y_s - v(t)) - \varepsilon] & \text{if } x_1(t) > 0 \end{cases} \tag{9}$$

i.e.,

$$\begin{cases} \dot{y}(t) > \frac{1}{x_1(t)}[x_2(t)(c(t) - x_1(t) - y_s - v(t)) - \varepsilon] & \text{if } x_1(t) < 0 \\ \dot{y}(t) < \frac{1}{x_1(t)}[x_2(t)(c(t) - x_1(t) - y_s - v(t)) - \varepsilon] & \text{if } x_1(t) > 0 \end{cases} \tag{10}$$

The two inequalities in (10) depict how the IntelRate controller $y(t) = f(e(t), g(e(t)))$ should behave in order to guarantee the asymptotic stability of the system upon different traffic conditions. These behaviors are established by the following two theorems.

3.3. **Stability analysis under light traffic ($x_1(t) < 0$).** Theorem 3.1 below establishes the behaviors of the IntelRate controller under light traffic.

Theorem 3.1. *The inequality $\dot{y}(t) > \frac{1}{x_1(t)}[x_2(t)(c(t) - x_1(t) - y_s - v(t)) - \varepsilon]$ if $x_1(t) < 0$ in (10) corresponds to the light traffic scenario in a router where the IntelRate controller allows each source to increase its sending rate until the desirable rate is reached while maintaining the system stable.*

Proof: According to the upper inequality in (10) and to the definition of $x_1(t)$, the case of $x_1(t) < 0$ implies the incoming traffic $y(t)$ is arriving at a rate less than y_s. From Proposition 1, it means that $y(t) < c(t)$, or $c(t) - x_1(t) - y_s - v(t) > 0$ (where $v(t) = 0$, as assumed in Assumption 1). Therefore, the aggregate incoming rate $y(t)$ is less than the router service rate $c(t)$, and the system is working in a light traffic scenario. In this context, if $x_1(t)$ remains less than zero, $q(t)$ is working in an empty state, i.e., $x_2(t) < 0$.

Note that ε in (10) can be any small positive number to ascertain the stability condition in (8). As ε approaches zero from the positive side, the right side of the upper inequality in (10) would be positive, which in turn means the output change $\dot{y}(t)$ has to be positive. This requires the controller to increase its output $y(t)$ to make $q(t)$ reach q_s, provided every source has enough data to send. The scenarios of this for the controller to achieve stability are enumerated and exemplified below. The rule region that is shaded gray in Table 1 regulates the controller behaviors under the light traffic scenario where $e(t) = q_0 - q(t) > 0$ (since $x_2(t) < 0$).

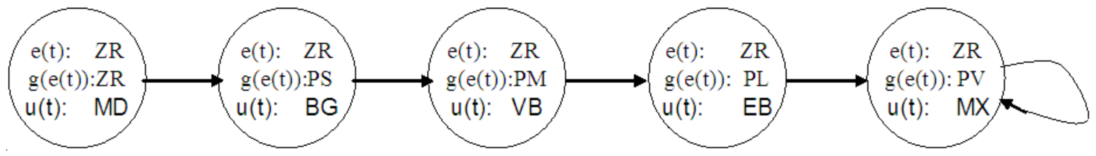


FIGURE 4. The IntelRate controller behavior when $e(t)$ = "ZR" (light traffic case)

Figure 4 shows the controller behavior under one of the light traffic scenarios, i.e., when $e(t)$ = "ZR" (which corresponds to the left column of the shaded area in Table 1). Since $g(e(t)) = \int e(t)dt$ and $e(t) > 0$, $g(e(t))$ would keep on increasing from ZR to PS, PM, PL until PV, and eventually the controller output reaches the linguistic value "MX", as seen in Figure 4, where all the sources can send data with their maximum desired rates.

Figure 5 shows the controller behavior under the other light traffic scenario, i.e., when $e(t)$ = "PS" (which corresponds to the right column of the shaded area in Table 1). Like the $e(t)$ = "ZR" case above, the controller also increases its output continuously until the "MX" is reached.

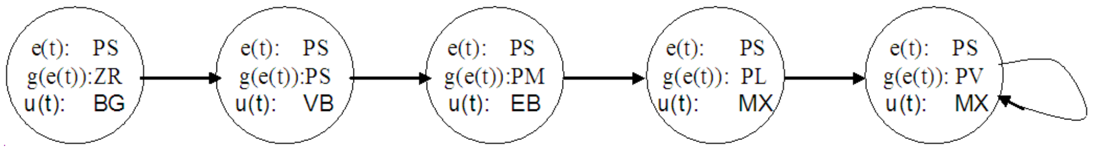


FIGURE 5. The IntelRate controller behavior when $e(t)$ = "PS" (light traffic case)

Note that in the light traffic condition, the bandwidth of a link is enough to deal with all the incoming traffic, and there is no congestion. The queue is often close to empty (Note this situation corresponds to the system model with the boundary condition $q(t) = 0$ in (7)), or occasionally there can be a low buffer occupancy due to some arrival bursts. In the situation where there is an occasional large data burst (which causes the queue size $q(t)$ to exceed q_0 resulting in $e(t) < 0$), the variable $g(e(t))$ would decrease its value for a while until the burst has subsided and $q(t)$ drops below q_0. During the decrease, the controller output would return to the previous states for a while. For example, the LVs may go from "MX" back to "EB" or even "VB". After the burst dissipates, $g(e(t))$ will increase again. Therefore, the general practice that $e(t) > 0$ under light traffic scenario is justifiable.

In summary, the controller behaviors described in the above two cases required by the upper inequality in (10) is met for the light traffic condition, and the proof is complete.

Theorem 3.1 has established that the IntelRate controller can follow the prescribed behaviors to guarantee the system stability in the presence of light traffic. It is noteworthy that the queue size $q(t)$ mostly operates below q_0 under light traffic scenarios. However, this does not affect the system stability because the router is not congested at all in such scenarios. This will also be verified by our simulations in Section 4.1. Note that the evolutions among NV, NL, NM and NS for $g(e(t))$ are also possible, but they correspond to the cases of $g(e(t)) < 0$ and $e(t) < 0$ which will be discussed in the next section.

3.4. **Stability analysis under heavy traffic ($x_1(t) > 0$).** Theorem 3.2 below establishes the behaviors of the controller under heavy traffic.

Theorem 3.2. *The inequality $\dot{y}(t) < \frac{1}{x_1(t)}[x_2(t)(c(t) - x_1(t) - y_s - v(t)) - \varepsilon]$ if $x_1(t) > 0$ in (10) corresponds to the heavy traffic scenario in a router where the IntelRate controller decreases the source sending rate according to max-min fairness until the queue size reaches the TBO q_0 so that the asymptotical stability of the IntelRate control system is guaranteed.*

Proof: The condition $x_1(t) > 0$ in the lower inequality of (10) implies the incoming traffic $y(t)$ is arriving at a rate higher than y_s. From Proposition 1, it means $y(t) > c(t)$, or $c(t) - x_1(t) - y_s - v(t) < 0$. Since the aggregate incoming rate $y(t)$ to the queue is higher than the router service rate $c(t)$, and when $x_1(t)$ is to remain bigger than zero, $q(t)$ would build up quickly and pass over q_0. The system is then working under heavy traffic with $x_2(t) > 0$. Thus, the right side of the lower inequality in (10) becomes negative. Note that ε was approximated as zero. Now $\dot{y}(t) < 0$ means the controller should decrease its output $y(t)$ so that the IQSize $q(t)$ can be reduced back to q_0 (which is the steady state q_s) and then be stabilized there. To analyze whether the controller can behave this way, below we do the analysis based on the shaded rule region of Table 2 which regulates the controller behaviors in the presence of heavy traffic.

For simplicity, assume the router is originally working in a non-congested condition where the controller allows the flows to send data with the LV "MX" (e.g., the scenario described in the proof of Theorem 3.1). We then assume an infinite number of ftp flows swarm into the router, each greedily demanding a desired sending rate. Obviously, this

Table 2. Rules for heavy traffic (shadowed gray)

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

is the worst case assumption for a link to become severely congested. In such a scenario, when the link bandwidth is fully utilized and the congestion becomes imminent, the queue begins to build up (Note this situation corresponds to the system model under the boundary condition $q(t) > 0$ in (7)), and that $e(t) = q_0 - q(t)$ would become smaller and smaller. Eventually, $e(t)$ would become negative after $q(t) > q_0$. With reference to the shaded area of Table 2, the negative $e(t)$, i.e., $e(t) < 0$, corresponds to the state $e(t) =$ "ZR" and $e(t) =$ "NS". As a result, $g(e(t))$ starts to decrease along the direction from "PV" to "NV" until $q(t)$ is working around the TBO q_0. When the controller input $g(e(t))$ decreases to "NV", the output would reach the linguistic value of "ZR" independent of the input $e(t)$, which means the controller is trying to assign an infinitesimal portion of bandwidth to each of the flows. As such, there are the following possible cases of the IntelRate controller behavior.

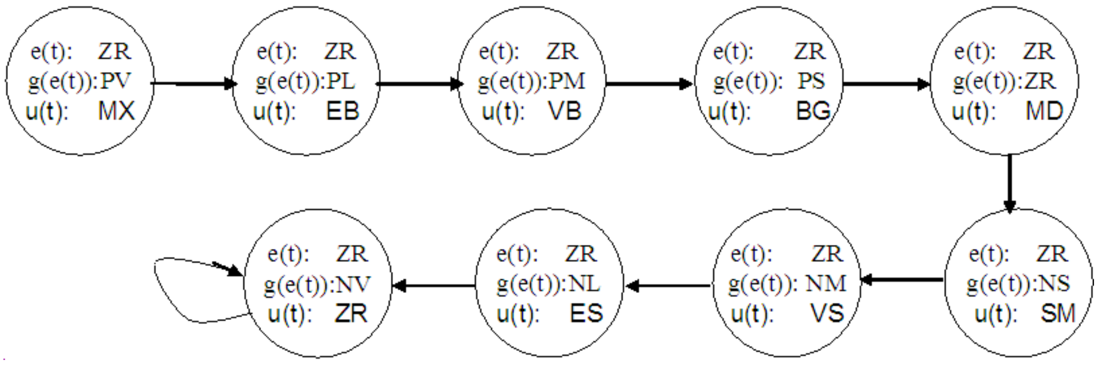


Figure 6. The IntelRate controller behavior when $e(t) =$ "ZR" (worst traffic case)

a) The case of $e(t) =$ "ZR" as shown in Figure 6 (which corresponds to the right column of the shaded area in Table 2).

b) The case of $e(t) =$ "NS" as shown in Figure 7 (which corresponds to the left column of the shaded area in Table 2).

In both cases, one can see the controller output eventually decreases to "ZR" when $g(e(t))$ remains negative in order to maintain $q(t)$ to q_0.

Beside the above worst cases, the IntelRate controller output need not decrease all the way to "ZR". This leads to the other two cases:

c) The case of $e(t) =$ "ZR" with bi-directional transition as shown in Figure 8.

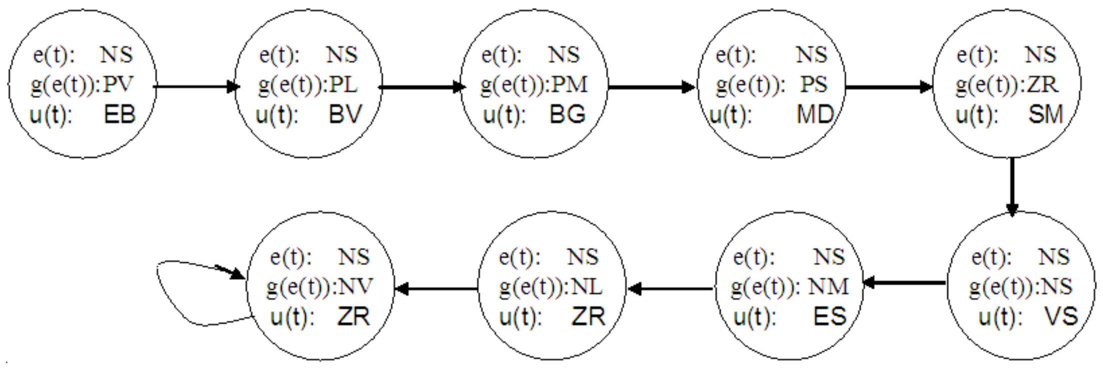


FIGURE 7. The IntelRate controller behavior when $e(t)$ = "NS" (worst traffic case)

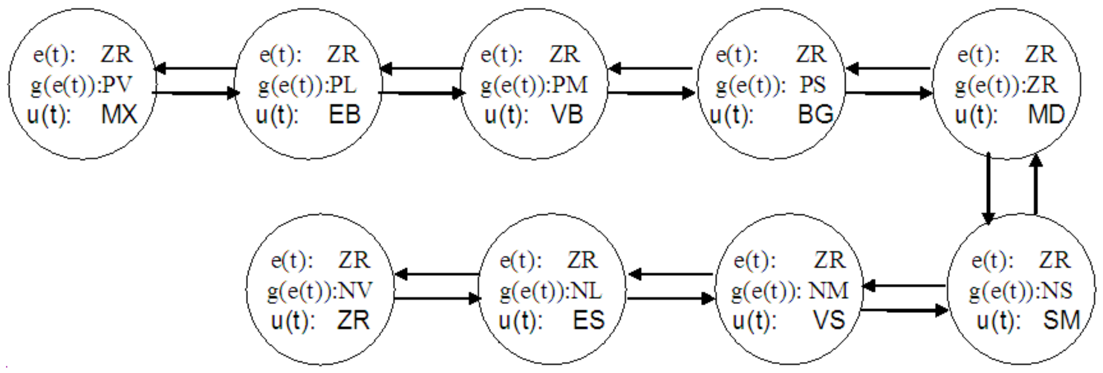


FIGURE 8. The IntelRate controller behavior when $e(t)$ = "ZR" (heavy traffic case)

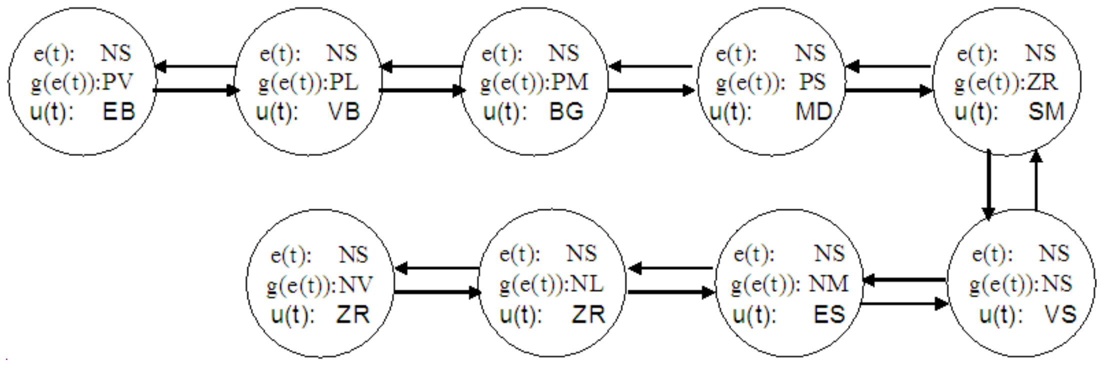


FIGURE 9. The IntelRate controller behavior when $e(t)$ = "NS" (heavy traffic case)

d) The case of $e(t)$ = "NS" with bi-directional transition as shown in Figure 9.

System stability can still be accomplished in cases c) and d) above. This is because, as shown in Figure 8 or Figure 9, the IntelRate controller can transit bi-directionally among the neighboring rules. Take the controller rule (ZR, NS, SM) in Figure 8 as an example. If this controller rule cannot decrease the queue size $q(t)$ to q_0, $g(e(t))$ will become more negative (because $e(t)$ is still less than 0), and consequently the controller will make a transition to the next rule (ZR, NM,VS) to further decrease its output. If $q(t)$ is still greater than q_0, the controller will continue to decrease the output by going to the next rule (ZR, NL, ES) for the same reason. This process is repeated until one output can decease the queue size $q(t)$ to q_0 and stabilize it there. Simulations in Section 4.2 will further show how the IQSize is stabilized around q_0. In fact, under such a process, the system is working in max-min fairness. Similarly, for any given rule that the controller initially resides in, it can be shown that the IntelRate controller can always adjust the sending rates to shift $q(t)$ back to q_0 and make the system stable. In summary, the controller behavior in the heavy traffic conditions required by the lower inequality in (10) is met.

Finally, the only special case to analyze is when $x_1(t) = 0$, i.e., the incoming rate $y(t)$ to the router remains constant at y_s. We consider this as a heavy traffic scenario because the bandwidth has been fully utilized, i.e., $y(t) = c(t)$. From $\dot{V}(\boldsymbol{x}) = x_1(t)h(x_2, \dot{x}_2) + x_2(t)(x_1(t) + y_s + v(t) - c(t)))$, we note that $\dot{V}(\boldsymbol{x}) = 0$ only when $\boldsymbol{x} = 0$. Furthermore, as $|\boldsymbol{x}| \to \infty$, $V(\boldsymbol{x}) = (x_1^2(t) + x_2^2(t))/2 \to \infty$. Therefore, from the stability principle of LDM, along with the established Theorem 3.1 and Theorem 3.2, we can now conclude that the IntelRate control system is globally asymptotically stable given any initial state of $(y(t), q(t))$. Please note that the initial state $(y(t), q(t))$ should be compliant with the system physical limitations, e.g., $q(t)$ should neither be a negative value nor be greater than the buffer capacity B.

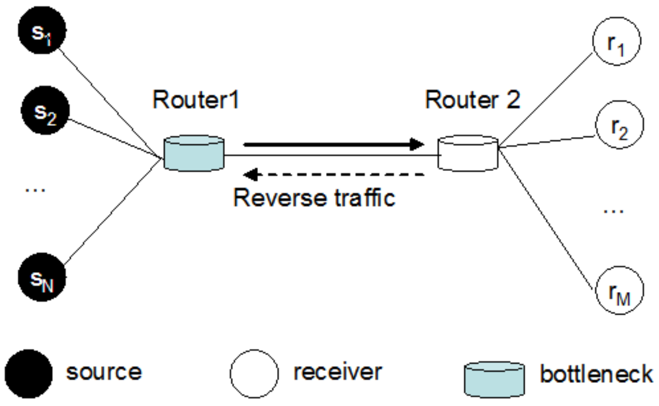


FIGURE 10. A wired network

4. **Congestion Control of a High-Speed Wired Network.**

The capability of our IntelRate controller can be demonstrated in wired networks using OPNET Modeler [35]. Figure 10 depicts one common congestion scenario encountered in Internet [3,4,26-28], in which Router 1 is the bottleneck. The numbers marked on the nodes designate the subnets/groups attached to each router. There are $M = N = 11$ subnet pairs to form the source-destination data flows in the network, and they run various Internet applications such as long-lived ftp, short-lived http, or unresponsive UDP-like flows (the uncontrolled ftp flows [4]).

Table 3 summarizes the RTPD (Round Trip Propagation Delay) used in each flow. 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 send the ACK packets to the sources.

The bandwidth of Router 1 is 5Gbps. Since we are going to investigate the controller performance in the bottleneck Router 1, Router 2 is configured to have sufficient link bandwidth $c(t)$ and buffer B so that congestion never happens there. The TBO of the bottleneck in this application example is 6000 packets, which can cause a queueing delay of 9.83ms (from 6000 packets*1024bytes*8bits/5Gbps). The buffer capacity $B = 10$*TBO. We also adopt some typical values from the experiments of existing works so that we can

TABLE 3. Source 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

make meaningful comparisons later on. In particular, all the ftp packets have the same size of 1024 bytes [4] while the http packet size is uniformly distributed in $[800, 1300]$ bytes [36].

In order to demonstrate and discuss the stability and robustness of our IntelRate controller, our study would focus on the testing of the 100 long-lived ftp sources (because they tend to cause severe congestion in the network), unless otherwise stated. For brevity reason we shall show the results of one flow from each group. The 100 sporadic short-lived http flows just act as the disturbance of the ftp traffic. Their transfer size follows the real web traffic scenario, i.e., it has a Pareto distribution [36] with a mean transfer size of 30 packets [2]. The arrivals of http flows follow a think-time [36] uniformly distributed in $[0.1\text{s}, 30\text{s}]$. One of the http session examples is shown in Figure 11. The uncontrolled ftp flow keeps its window size at 100 packets and operates at an almost constant rate as shown in Figure 12. These http and UDP-like flows generate an aggregate traffic $v(t)$ mentioned in Assumption 1 of Section 3.1. The IntelRate control system 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 [35]. Here, a bit is considered successfully sent out if it is part of a packet that is successfully sent out [35].

2) IQSize is the length of the bottleneck buffer (measured in packets) seen by a packet departure [37].

3) Utilization is the ratio between the current actual throughput in the bottleneck and the maximum service rate of the bottleneck. It is expressed as a percentage.

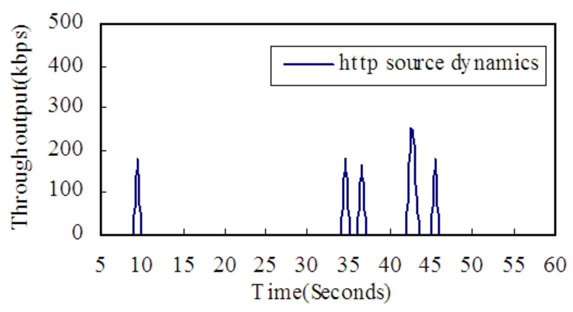


FIGURE 11. Http sessions example

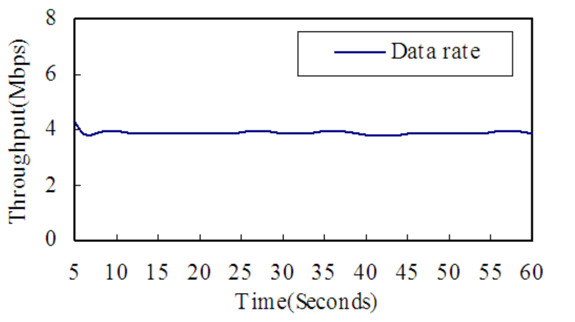


FIGURE 12. Uncontrolled ftp flow

4) Max-min fairness: A feasible allocation of rates is 'max-min 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 [38].

4.1. **Stability in light traffic scenario.** The light traffic scenario is used to demonstrate that the IntelRate controller can maintain system stability by allowing all flows to send data with their desired rates, as analyzed in Section 3.3. The desired sending rates from ftp Group 1 to Group 5 are set 6.14Mbps, 10.08Mbps, 20.15Mbps, 41.78Mbps and 76.19Mbps, respectively. Note that the total desired rate is less than the bottleneck bandwidth, i.e., 5Gbps, in order not to cause congestion in the bottleneck.

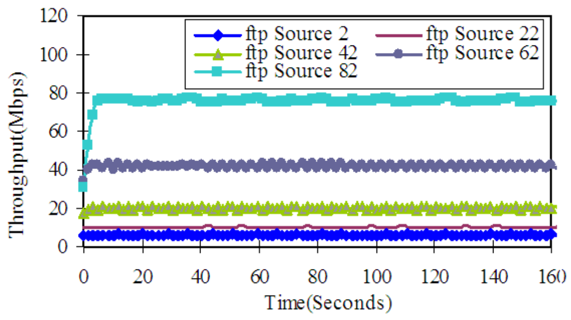


FIGURE 13. Sources throughput dynamics

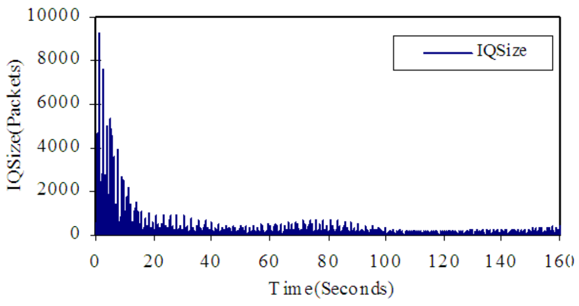


FIGURE 14. Router IQSize

Figure 13 shows the throughput of 5 ftp sources, one from each ftp group. In comparison to their desired rate we set above, all the flows can stably send data in a rate they desired. This verifies our system stability principle stated in Theorem 3.1 of Section 3.3. Further discussion will be provided shortly.

Figure 14 shows the router IQSize dynamics. Compared with the TBO, i.e., 6000 packets, the IQSize operates in a very low buffer level most of the time. After the router is started, the controller is trying to increase the queue size to the TBO. However, because of the total incoming traffic less than the bottleneck bandwidth, the queue size has to settle at a lower level eventually. This indicates that, in a light traffic condition, the IQSize may not reach the TBO. Since the router is not congested, this does not cause the system instability. As a matter of fact, it is queue overflow that would cause congestion so the system instability, which will be investigated in the heavy traffic scenario later on.

Discussion: It is interesting to review once more the fuzzy logic mechanism that allows the IntelRate sources to send data with their desired rate in light traffic scenario. As shown in Figure 14, when the IQSize $q(t)$ is less than TBO, $e(t)$ is always positive (corresponding to the LV "ZR" or "PS" of $e(t)$ in Figure 2). Consequently, $g(e(t))$ is increasing until it reaches the maximum edge value mq_0 (corresponding to the "PV" of $g(e(t))$). According to Table 1, the system is now working under the rule (ZR, PV, MX) or (PS, PV, MX). Either one means the IntelRate controller allows the "MX" sending rate, where the value D resides. Therefore, every source can send data with its desired rate.

Unlike the heavy traffic scenario to be discussed below, we have no max-min fairness issue here. This is because the max-min fairness is a kind of resource allocation mechanism for many greedy consumers sharing limited resources. This fairness works by first satisfying the small users, and then evenly allocating the remaining resources among the large users (as will be illustrated in the next section).

4.2. **Stability in heavy traffic scenario.** In this section, we want to demonstrate that upon heavy traffic the IntelRate controller can maintain system stability by allocating the bottleneck bandwidth according to max-min fairness, as analyzed in Theorem 3.2 of Section 3.4. The ftp flows in Group 1 to Group 5 would desire higher sending rates, i.e., 24.48Mbps, 33.18Mbps, 49.15Mbps, 90.11Mbps and 114.69Mbps respectively, which are the rates we use to produce congestion.

Figure 15 shows the stable sending rates of 5 sources, one from each ftp group. The sources in groups 1, 2 and 3 (e.g., sources 2, 22 and 42) obtain a throughput they desire, i.e., 24.48Mbps, 33.18Mbps and 49.15Mbps, respectively. However, the sources in groups 4 and 5 (e.g., sources 62 and 82) cannot obtain their desired rates 90.11Mbps and 114.69Mbps, respectively. Instead they have to share the same portion of bandwidth, i.e., 82.45Mbps. This illustrates and verifies the max-min fairness capability we implemented for the IntelRate controller.

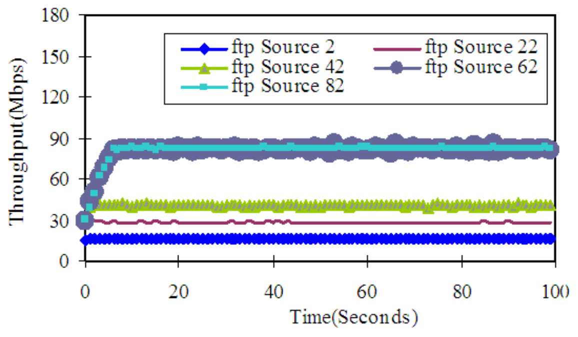


FIGURE 15. Source throughput dynamics

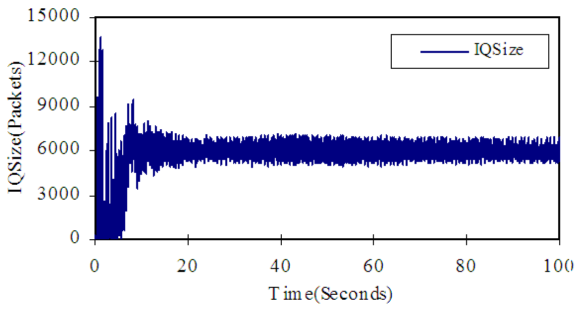


FIGURE 16. Router IQSize

Figure 16 shows the router IQSize is well controlled around the TBO, i.e., 6000 packets. Even though the queue size may rise sharply at the starting stage, but its level is much smaller than the buffer size of 60000 packets. This demonstrates that the IntelRate controller is capable of restricting the IQSize around the TBO so that the buffer is not overflowed and the system is stable. This also leads to another advantage observed in our controller, i.e., there is no packet loss upon heavy traffic.

Discussion: Now we analyze how the IntelRate controller imposes max-min fairness under heavy traffic to maintain the system stable. As mentioned before, the system is stable because the controller is capable of controlling the IQSize around the TBO. This means that $e(t)$ is hovering around zero depending on the relative values of $q(t)$ and q_0. Consequently, the input $g(e(t))$ decreases or increases its crisp value. According to Table 1, the router outputs an allowed sending rate $u(t)$ for each flow according to the crisp values of $e(t)$ and $g(e(t))$. The flows whose desired rates larger than $u(t)$ (e.g., the above flow 62 and 82) will be limited to $u(t)$, while those with smaller desired rates are allowed to send data with the rates they desire (e.g., flows 2, 22 and 42 above). This is exactly the max-min fairness the IntelRate controller was designed for.

5. **Congestion Control of a Wireless Network.** In IEEE 802.11 wireless LAN (Local Area Network), the wired and the wireless interfaces of the AP (Access Point) are characterized by the disparity in channel capacity. This presents a significant bottleneck for traffic flowing from the wired network to the wireless network [39]. With this practical example, we aim to demonstrate that the IntelRate controller can stably operate in the presence of traffic changes under heavy traffic situations.

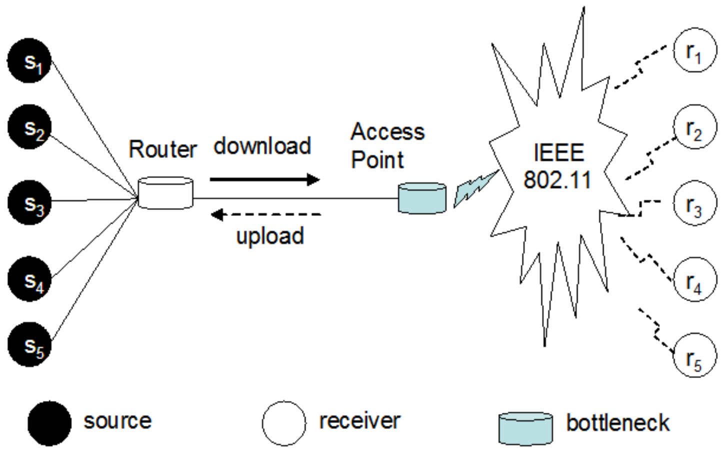


Figure 17. A wireless network

The wireless LAN topology is shown in Figure 17, which consists of 5 source-destination pairs (i.e., $s_i - r_i$, $i = 1, 2, \ldots, 5$). Since wireless LAN usually is a small scale network, not like the above wired network, we do not duplicate flows for each subnet. The backhaul has a propagation delay of 100ms. The bandwidth between s_i and the router is 100Mbps, and the backhaul bandwidth between the router and the AP is 1Gbps. The data rate of the IEEE 802.11 wireless LAN is 11Mbps. The IntelRate controller resides in the AP to prevent the congestion.

We use the Application and Profile module of the OPNET to generate traffic in the sources s_i. In the Application module, we choose video (this module also has other types of traffic available such as ftp, http, audio or email) as our network traffic because like ftp it is another giant network traffic maker nowadays. The starting time, duration and the number of repetitions of the video can be set in the Profile module. Network traffic change can happen due to video flow joining or transmission ending (including connection broken) in wireless networks. To produce traffic changes, we only allow video flows from the sources s_1, s_2 and s_5 to operate in the first 50 seconds. The flows from s_3 and s_4 will join in at $t = 50$s and $t = 100$s, respectively. Then the flow from s_2 will finish its transmission at $t = 200$s while other flows remaining operation. We will show that the IntelRate controller is able to adjust the video generation rate following the explicit congestion signal so that the congestion is prevented in AP. To make the experiment more rigid, we let all the sources have a greedy video sending rate.

Figure 18 shows the sending rate dynamics of each flow during the traffic change processes. The figure demonstrates that, whenever a new flow starts, the IntelRate controller can reallocate the bandwidth in max-min fairness so that the system stability is guaranteed. Since every flow wants to send data into the network as much as it can, the max-min fairness in this case is imposed to evenly re-allocate the bandwidth among the flows upon each traffic change. For example, in the first 50s, flows 1, 2 and 5 evenly share the 11Mbps bandwidth, each with about 3.65Mbps. At $t = 50$s, after the flow 3 joins in, the sending rate of the flow 1, 2 and 5 decreases and the 4 flows now each share 2.75Mbps. Likewise, at $t = 100$s, after the flow 4 joins in, each flow shares 2.2Mbps. After the flow 2 withdraws at $t = 200$s, the remaining 4 flows each increases to share 2.75Mbps again. In summary, the sources are stable in readjusting their sending rates upon each traffic change and provide smooth performance.

Figure 19 shows that the IQSize is well controlled and stably operating around the TBO of 60 packets. The fluctuations in IQSize are small when the traffic changes, and are even

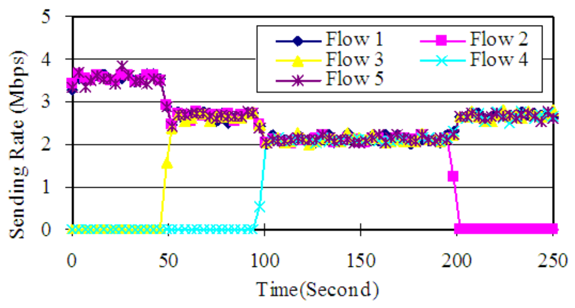


FIGURE 18. Source throughput dynamics

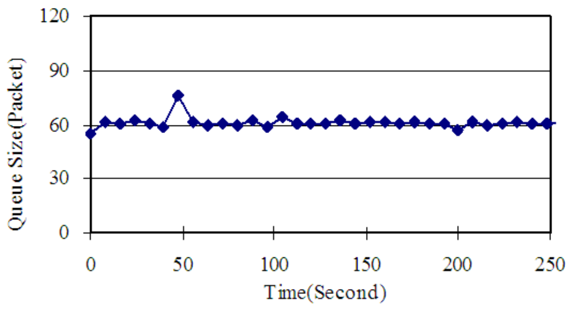


FIGURE 19. IQSize in AP

negligible at $t = 100$s and 200s. Furthermore, all these fluctuations quickly settled down to the TBO just after one oscillation.

In conclusion, Figure 18 and Figure 19 demonstrate good stability as well as fast response of the IntelRate controller upon traffic changes in the wireless network.

6. Comparison with Other Controllers.

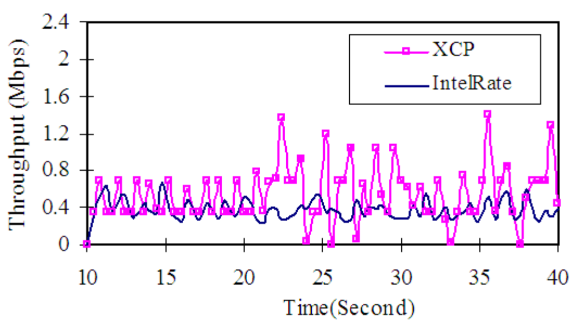
The existing explicit congestion control protocols such as XCP and RCP all need to estimate the link price or the bottleneck bandwidth in order to compute the source sending rates. As pointed out in the investigation of XCP in [7], the potential mis-estimation of bottleneck bandwidth can result in non-converging throughput and thus instability. Our IntelRate controller does not have this problem which will be demonstrated in Section 6.1. Section 6.2 shows that the IntelRate controller, without estimating the number of flows, can even achieve better stable performance than the API-RCP upon the traffic changes while saving the precious router computation resources.

6.1. Comparison with XCP.

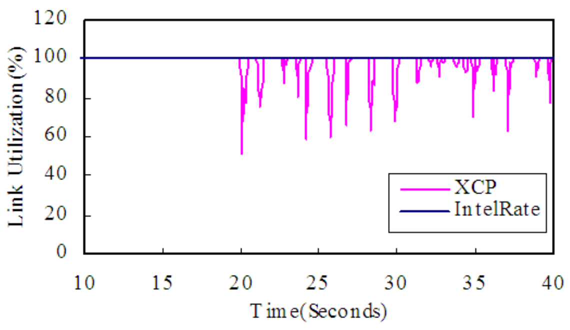
We set up a 45Mbps bottleneck (i.e., Router 1) in Figure 10, and allow all flows have a common round trip time of 40ms, as set in [2]. The XCP design parameters are the same as the values adopted in [2], i.e., $\alpha = 0.4$ and $\beta = 0.226$. The 100 long-lived ftp flows share the bottleneck and greedily consume all the bandwidth. To allow comparison, we change the bottleneck bandwidth from 45Mbps to 40Mbps at $t = 20$s in order to mimic the error in bandwidth estimation. Such a bandwidth change can happen in wired networks due to the sudden joining of some uncontrolled traffic such as UDP so that a portion of bandwidth becomes unavailable to the controlled traffic. Then we test the IntelRate controller with the same bandwidth change.

As shown in Figure 20(a), the throughput of XCP flow becomes unstable after the bandwidth changes from 45Mbps to 40Mbps at $t = 20$s, and cannot converge to a relatively smooth state. The reason can be seen from its efficiency equation $\phi = \alpha \cdot (c - y(t)) - \beta \cdot Q/d$, where $y(t)$ is the aggregate incoming rate as in our IntelRate controller and c is the estimated bandwidth. Without an accurate estimation of c (as emulated in this experiment), the stability performance of XCP controller is greatly degraded. In contrast, the IntelRate controller is based on the heuristic expert knowledge so that the source is able to find a new rule according to the queue size variation, and thus adapting itself to the new bandwidth condition. This is why the IntelRate controller can maintain the flow throughput relatively smooth and the system stable upon bandwidth change.

Figure 20(b) shows how the link utilization of XCP degrades accordingly when its flow rate becomes unstable, while the IntelRate controller maintains full utilization all the time due to its relatively stable sending rate.



(a)



(b)

FIGURE 20. Source throughput dynamics and link utilization

6.2. **Comparison with API-RCP.** API-RCP depends on an additional mechanism, called self-adaptation, to adapt its PI gains to the new traffic conditions. With such a mechanism, API-RCP can detect and deal with the changes in the number of flows so that the system can maintain stable [4]. However, the IntelRate controller does not need to estimate the number of flows at all and thus saving much router computation and memory resource. In the following, with the wired network Figure 10, we will demonstrate that the IntelRate controller can achieve better response performance upon the traffic change than API-RCP while maintaining the system stable.

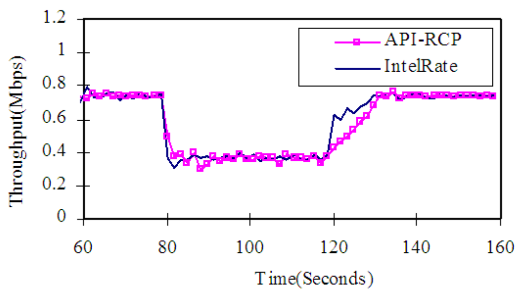


FIGURE 21. Source performance

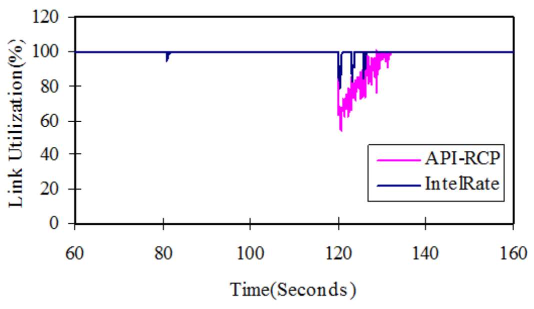


FIGURE 22. Link utilization

We set up a 45Mbps bottleneck bandwidth with $B = 1000$ packets and TBO $= 300$ packets, as described in [4]. We use a phase margin of 45 degrees for the API-RCP which is considered to be within an optimum range of the settings [4]. We set $M = N = 100$ ftp sources. In this experiment, in the first 80 seconds, we only have 60 ftp flows operating in Group 2, Group 4 and Group 5. Then at $t = 80$s, 40 ftp flows in Group 1 and Group 3 join in the bottleneck traffic and stay for 40 seconds before ending their transmission at $t = 120$s. Figure 21 compares the throughput performance of the two controllers when additional 40 ftp flows from Group 1 and Group 3 join in or leave the bottleneck. The two controllers have similar response behavior in the first 120 seconds but differs noticeably when they try to recover their sending rates after $t = 120$s (referred to as "recovery stage"). One sees that the IntelRate controller can recover its throughput faster than the API-RCP while maintaining good stability. This suggests that the IntelRate controller can utilize the spare bandwidth more quickly and efficiently upon traffic change. Figure 22 has verified such an allegation. As shown, during the recovery stage, the API-RCP shows a dramatic and longer utilization drop, whereas the IntelRate controller just shows three small and short drops.

As a summary, the IntelRate controller, without estimating the number of flows, not only saves router computation and memory resources, but also is able to reutilize the spare bandwidth faster during the recovery stage while remaining the system stable.

7. 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. In the meanwhile, it saves the computational resources in a router. To theoretically prove the stability of the IntelRate control system, we have reversely employed the Lyapunov's stability criteria and showed that the IntleRate control system is able to achieve global asymptotical stability. The simulations have verified the effectiveness and the superiority of the fuzzy-logic-based IntelRate controller, and demonstrated that the IntelRate control system can remain stable by relying on the IQSize alone (the least number of parameter we can have) to effectively prevent the network from congestion.

Acknowledgment. This work is partially supported by Research Discovery Grant (#R-GPIN42878) and an Accelerated Grant from NSERC. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the quality and presentation of our paper.

REFERENCES

[1] J. Liu and O. Yang, Design and evaluation of an intelligent controller for heterogeneous networks, *Proc. of IEEE GLOBCOM*, Miami, USA, pp.1-5, 2010.

[2] D. Katabi, M. Handley and C. Rohrs, Congestion control for high bandwidth-delay product networks, *Proc. of SIGCOMM*, Pittsburgh, Pennsylvania, USA, pp.89-102, 2002.

[3] N. Dukkipati, N. McKeown and A. G. Fraser, RCP-AC congestion control to make flows complete quickly in any environment, *Proc. of IEEE INFOCOM*, Barcelona, Spain, pp.1-5, 2006.

[4] Y. Hong and O. Yang, Design of adaptive PI rate controller for best-effort traffic in the Internet based on phase margin, *IEEE Trans. Parallel & Distr. Syst.*, vol.18, no.4, pp.550-561, 2007.

[5] Y. Hong and O. Yang, An API-RCP design using pole placement technique, *Proc. of IEEE ICC*, pp.1-5, 2010.

[6] Y. Hong and O. Yang, Can API-RCP achieve max-min fair bandwidth allocation in a multiple-bottleneck network, *Proc. of the 43rd CISS*, pp.723-728, 2009.

[7] Y. Zhang and T. R. Henderson, An implementation and experimental study of the explicit control protocol (XCP), *Proc. of IEEE INFOCOM*, Miami, USA, vol.2, pp.1037-1048, 2005.

[8] B. Ribeiro, T. Ye and D. Towsley, Resource-minimalist flow size histogram estimator, *Proc. of the 8th ACM SIGCOMM Conference on Internet Measurement*, Greece, pp.285-290, 2008.

[9] S. Guo and O. Yang, Energy-aware multicasting in wireless ad hoc networks: A survey and discussion, *Computer Communication*, vol.30, no.9, pp.2129-2148, 2007.

[10] H. K. Lam, F. H. Leung and P. K. S. Tam, An improved stability analysis and design of fuzzy control systems, *Proc. of IEEE Int'l Fuzzy Systems Conference*, vol.1, pp.430-433, 1999.

[11] H. Li, H. Liu, H. Gao and P. Shi, Reliable fuzzy control for active suspension systems with actuator delay and fault, *IEEE Trans. on Fuzzy Systems*, vol.20, no.2, pp.342-357, 2012.

[12] T. Gusmi, H. H. Adballa and A. Toumi, Transient stability fuzzy control approach for power systems, *Proc. of IEEE Int'l conf. on Industrial Technology*, vol.3, pp.1676-1681, 2004.

[13] T. Sijak, S. Tesnjak and O. Kuljaca, Stability analysis of fuzzy control systems using describing function method, *Proc. of the 9th Mediterranean Conference on Control and Automation*, 2001.

[14] R. E. Haber, G. Schmitt-Braess, R. H. Haber, A. Alique and J. R. Alique, Using circle criteria for verifying asymptotic stability in Pl-like fuzzy control systems: Application to the milling process, *IEE Proceedings of Control Theory and Applications*, vol.150, no.6, pp.619-627, 2003.

[15] G. R. Yu, Robust fuzzy control of piezoelectric systems with input delays and disturbances based on piecewise Lyapunov functions, *International Journal of Innovative Computing, Information and Control*, vol.4, no.10, pp.2721-2730, 2008.

[16] J. L. Meza, V. Santibanez, R. Soto and M. A. Llama, Stable fuzzy self-tuning pid controller of robot manipulators, *Proc. of IEEE Int'l Conf. on Sys., Man,and Cybernetics*, pp.2624-2629, 2009.

[17] V. Santibanez, R. Kelly and M. A. Llama, Global asymptotic stability of a tracking sectorial fuzzy controller for robot manipulators, *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.34, no.1, pp.710-718, 2004.

[18] H. Han, Fuzzy controller design with input saturation, *International Journal of Innovative Computing, Information and Control*, vol.4, no.10, pp.2507-2521, 2008.

[19] L. Wang and X. Liu, New relaxed stabilization conditions for fuzzy control systems, *International Journal of Innovative Computing, Information and Control*, vol.5, no.5, pp.1451-1460, 2009.

[20] Z. G. Wu, P. Shi, H. Su and J. Chu, Reliable H_∞ control for discrete-time fuzzy systems with infinite-distributed delay, *IEEE Trans. on Fuzzy Systems*, vol.20, no.1, pp.22-31, 2012.

[21] J. Zhang, P. Shi and Y. Xia, Robust adaptive sliding-mode control for fuzzy systems with mismatched uncertainties, *IEEE Trans. on Fuzzy Systems*, vol.18, no.4, pp.700-711, 2010.

[22] Q. Zhou, P. Shi, J. Lu and S. Xu, Adaptive output-feedback fuzzy tracking control for a class of nonlinear systems, *IEEE Trans. on Fuzzy Systems*, vol.19, no.5, pp.972-982, 2011.

[23] L. Wu, X. Su, P. Shi and J. Qiu, A new approach to stability analysis and stabilization of discrete-time t-s fuzzy time-varying delay systems, *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.41, no.1, pp.273-286, 2011.

[24] K. Michels, F. Klawonn, R. Kruse et al., *Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers*, 1st Edition, Springer, 2006.

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

[26] H. Aoul, A. Nafaa, D. Negru and A. Mehaoua, FAFC: Fast adaptive fuzzy AQM controller for TCP/IP networks, *Proc. of IEEE GLOBECOM*, vol.3, pp.1319-1323, 2004.

[27] X. Guan, B. Yang, B. Zhao et al., Adaptive fuzzy sliding mode active queue management algorithms, *Telecommunication Systems*, vol.35, no.1-2, 2007.

[28] Y. Jing, Z. Chen and G. M. Dimirovski, Robust fuzzy observer-based control for tcp/aqm network systems with state delay, *Proc. of American Control Conference*, pp.1350-1355, 2010.

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

[30] J. Harju and K. Pulakka, Optimisation of the performance of a rate-based congestion control system by using fuzzy controllers, *Proc. of IEEE International Performance, Computing and Communications Conference*, pp.192-198, 1999.

[31] W. T. Lau, K. K. Phang, Y. Mashkuri et al., Fuzzy logic control in ATM network, *Malaysian Journal of Computer Science*, vol.12, no.2, pp.47-56, 1999.

[32] N. Rouche, P. Habets and M. Laloy, *Stability Theory by Liapunov's Direct Method*, Springer-Verlag New York Inc., 1977.

[33] S. Floyd, *Measurement Studies of End-to-End Congestion Control in the Internet*, http://www.icir.org/floyd/ccmeasure.html, 2008.

[34] R. C. Dorf and R. H. Bishop, *Modern Control System*, 11th Edition, Pearson Prentice Hall, 2008.

[35] *Opnet Modeler Manuals*, Opnet Version 11.5, Opnet Technologies Inc., 2005.

[36] M. E. Crovella and A. Bestavros, Self-similarity in world wide web traffic: Evidence and possible causes, *IEEE/ACM Transactions on Networking*, vol.5, no.6, pp.835-846, 1997.

[37] D. Gross, J. Shortle et al., *Fundmentals of Queueing Theory*, John Wiley&Sons, 2008.

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

[39] C. N. Nyirenda and D. S. Dawoud, Fuzzy logic congestion control in IEEE 802.11 wireless local area networks: A performance evaluation, *Proc. of AFRICON*, 2007.