

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

Kadanti Theja

(CNIS)

Sree Vidyanikethan Engineering College, Andhra Pradesh

ABSTRACT

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

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

1. INTRODUCTION

Network traffic management can prevent a network from severe congestion and degradation in throughput-delay performance. Traffic congestion control is one of the effective approaches to manage the network traffic [1], [2]. Historically, Transmission Control Protocol Reno [3], [4] is a widely deployed congestion control protocol that handle the Internet traffic. It has the important feature that the network is treated as a

black box and the source alter its window size based on packet loss signal [5]. However, as an implicit control protocol, TCP confronts various performance problems (e. g., utilization, fairness and stability) when the Internet BDP (Bandwidth-Delay Product) continues to increase. These have been widely investigated with various proposed solutions such as the AQM (Active Queue Management) schemes [6]-[10] whose control protocols are also implicit in nature. As an alternative, a class of explicit congestion control protocols has been proposed to signal network traffic level more precisely by using multiple bits. Examples are the XCP [6], RCP [11], JetMax [12] and MaxNet [13]. These protocols have their controllers reside in routers and directly feed link information back to sources so that the link bandwidth could be efficiently utilized with good scalability and stability in high BDP networks.

Fuzzy Logic Control [16] has been considered for Intelligence Control. It is a methodology used to design robust systems that can contend with the common adverse synthesizing factors such as system nonlinearity, parameter uncertainty, measurement and modeling imprecision. In addition, fuzzy logic theory gives a convenient controller design approach based on expert knowledge which is close to human decision making, and readily helps engineers to model a complicated non-linear system. In fact, fuzzy logic control has been applied in industrial process control and showed extraordinary and mature control performance in accuracy, transient response, robustness and stability.

Specifically, the objectives of this paper are: 1) to design a new rate-based explicit congestion controller based on FLC to avoid estimating link parameters such as link bandwidth, the number of flows, packet loss and network latency, while remaining stable and robust to network dynamics (Hence, we make this controller “intelligent”); 2) to provide max-min fairness to achieve an effective bandwidth allocation and utilization; 3) to generate relatively smooth source throughput, maintain a reasonable network delay and achieve stable jitter performance by controlling the queue size; 4) to demonstrate our controller has a better QoS performance through case study.

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

For the remainder of the paper, the following notations and symbols pertain.

A	Edge value of MFs (Membership Functions) of $e(t)$, beyond which the MFs of $e(t)$ saturate
B	Buffer capacity
$c(t)$	Service rate (output link capacity) of a router
C	Edge value of MFs of $g(e(t))$, beyond which the MFs of $g(e(t))$ saturate
D	Outermost edge value of MFs of $u(t)$
$e(t)$	Queue error which is one input of the IntelRate controller
$g(e(t))$	Integration of $e(t)$ which is the other input of the IntelRate controller
m	Multiple of TBO to design the width limit for the MFs of input $e(t)$ and g

	(e (t))
N	Number of LVs (Linguistic Values)
q0	TBO of a router
q (t)	IQSize (Instantaneous Queue Size) of a router
u (t)	The controller crisp output for each flow
u_ (t)	Current source sending rate
v (t)	Aggregate uncontrolled incoming traffic rate to a router
y (t)	Aggregate controlled incoming traffic rate to a router (also aggregate controller output)
μP_j	Input fuzzy set of the IntelRate controller
μU_j	Output fuzzy set of the IntelRate controller
τ_{fi1}	Time delay of a packet from source i to a router
τ_{fi2}	Time delay of a packet from a router to its destination i
τ_{bi}	Feedback delay of a packet from destination i back to source i
τ_{pi}	RTPD (Round Trip Propagation Delay)
τ_i	RTT (Round Trip Time)

2. TRAFFIC MANAGEMENT PRINCIPLE AND MODELING

We consider a backbone network interconnected by a number of geographically distributed routers, in which hosts are attached to the access routers which cooperate with the core routers to enable end-to-end communications. Congestion occurs when many flows traverse a router and cause its IQSize (Instantaneous Queue Size) to exceed the buffer capacity, thus making it a bottleneck in the Internet. Since any router may become bottleneck along an end-to-end data path, we would like each router to be able to manage its traffic. Below is the general operation principle of our new traffic management/control algorithm.

Inside each router, our distributed traffic controller acts as a data rate regulator by measuring and monitoring the IQSize. As per its application, every host (source) requests a sending rate it desires by depositing a value into a dedicated field *Req_rate* inside the packet header. This field can be updated by any router en route. Specifically, each router along the data path will compute an allowed source transmission rate according to the IQSize and then compare it with the rate already recorded in *Req_rate* field. If the former is smaller than the latter, the *Req_rate* field in the packet header will be updated; otherwise it remains unchanged. After the packet arrives at the destination, the value of the *Req_rate* field reflects the allowed data rate from the most congested router along the path if the value is not more than the desired rate of the source. The receiver then sends this value back to the source via an ACK (ACKnowledgment) packet, and the source would update its current sending rate accordingly. If no router modifies *Req_rate* field, it means that all routers en route allow the source to send its data with the requested desired rate.

In order to implement our new controller in each router, we model a typical AQM router in Fig. 1 with M sources sending their Internet traffic to their respective destinations. For $i = 1, 2, \dots, M$, $u_{-i}(t)$ is the current sending rate of source i; $u_i(t)$ is the sending rate of source i determined by the routers along the end-to-end path; $y(t)$

is the incoming aggregate controlled flow rate; $v(t)$ is the incoming aggregate uncontrolled flow rate, and $c(t)$ is the link bandwidth (measured in bps). For a particular source-destination pair i , τ_{fi1} is the time delay of a packet from source i to the router, and τ_{fi2} is the time delay of the packet of source i from the router to the destination i , while τ_{bi} is the feedback delay from destination i back to source i . Obviously, $\tau_{pi} = \tau_{fi1} + \tau_{fi2} + \tau_{bi}$ is the RTPD (Round Trip Propagation Delay). Considering other delays en route (e. g., queueing delay), source i may update its current rate $u_i(t)$ according to the $u_i(t)$ when the ACK packet arrives after one RTT (Round Trip Time) τ_i .

Considering the possible dynamics of both incoming traffic and link bandwidth in the router in Fig. 1, we model the bottleneck link with a queue in which both the controlled arrival rate $y(t)$ and the service rate $c(t)$ may vary with respect to time. Let $q(t)$ be the router IQSize. The variations in $y(t)$ and/or $c(t)$ can cause changes in the queue size of a router, as expressed in the following differential equation.

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

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

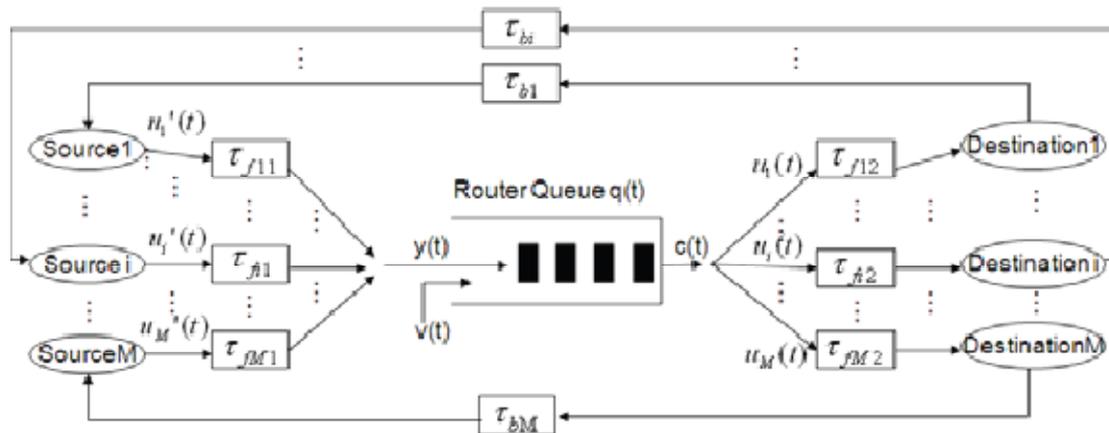


Fig. 1. System model of an AQM router.

3. THE INTEL RATE CONTROLLER DESIGN

Figure 2 depicts the components of our fuzzy logic traffic controller for controlling traffic in the network system defined in Fig. 1. Called the IntelRate, it is a TISO (Two-Input Single-Output) controller. The TBO (Target Buffer Occupancy) $q_0 > 0$ is the queue size level we aim to achieve upon congestion. The queue deviation $e(t) = q_0 - q(t)$ is one of the two inputs of the controller.

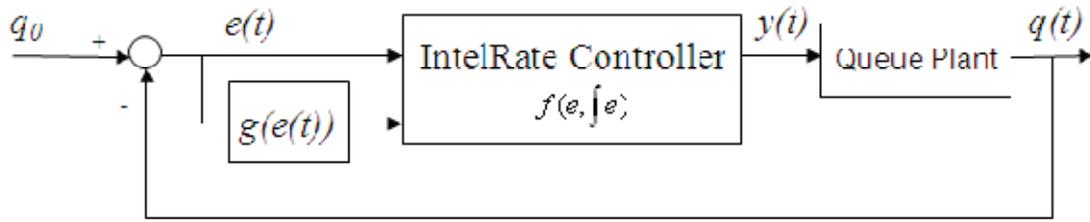


Fig. 2. The IntelRate closed-loop control system.

In order to remove the steady state error, we choose the integration of $e(t)$ as the other input of the controller, i. e. $g(e(t)) = \int e(t) dt$. The aggregate output is $y(t) = ui(t - \tau)$. Under heavy traffic situations, the IntelRate controller would compute an allowed sending rate $u_i(t)$ for flow i according to the current IQSize so that $q(t)$ can be stabilized around q_0 . In our design, IQSize $q(t)$ is the only parameter each router needs to measure in order to complete the closed-loop control. FLC is a non-linear mapping of inputs into outputs, which consists of four steps, i. e., rule base building, fuzzification, inference and defuzzification. The concepts of fuzzy set and logic of FLC were introduced in 1965 by Zadeh, and it was basically extended from two-valued logic to the continuous interval by adding the intermediate values between absolute TRUE and FALSE.

A. Linguistic Description and Rule Base

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

B. Membership Function, Fuzzification and Reference

Our IntelRate controller employs the isosceles triangular and trapezoid-like functions as its MFs (Membership Functions). Figure 3 describes the MFs used to determine

the certainty of a crisp input or output. We let P_1 be the UoD¹ for the input $p_1 = e(t)$, and P_2 be the UoD for the input $p_2 = g(e(t))$. The value of MFs (i. e., the certainty degree) for crisp inputs p_i ($i = 1, 2$) is designated by $\mu_{P_i}(p_i)$. Similarly, we let Z be the UoD of the output $z = u(t)$, and the certainty degree of the crisp output z is designated by $\mu_U(z)$. The above $\mu_{P_i}(p_i)$ or $\mu_U(z)$ is obtained by the ‘‘Singleton Fuzzification’’ⁱ method. Thus the input and output fuzzy sets can be defined with $P_i^j = \{(p_i, \mu_{P_i}(p_i)) : p_i \in P_i\}$, $i = 1, 2$ and, $U_j = \{(z, \mu_U(z)) : z \in Z\}$, $j = 1, 2, \dots, N$, respectively.

TABLE I RULE TABLE FOR INTELRATE CONTROLLER (9 LVS)

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

To determine how much a rule is certain to a current situation, our controller applies the Zadeh AND logic to perform the inference mechanism, e. g. for crisp inputs p_1 and p_2 , the final certainty (also referred to as the firing level) of a rule is computed with $\mu_{P_1} \cap \mu_{P_2} = \min \mu_{P_1}(p_1), \mu_{P_2}(p_2) : p[i] \in P_i, i = 1, 2, \dots, N$, where min is the minimum operation in the Zadeh AND logic. While Fig. 3 is used to illustrate the design of a general FS, the designated values actually come from an example of $N = 9$ LVs with the absolute values of both the upper and lower limits of $g(e(t))$ set to mq_0 . Since $e(t)$ is bounded by the physical size of a queue, we have the boundaries according to the limits $q_0 - B \leq e(t) \leq q_0$. The vertical dashed lines in Fig. 3 denote those boundaries of inputs or output. Accordingly, the dashed box in Table I contains the rules that the IntelRate controller operates on. There are three LVs that $e(t)$ can take (i. e., ‘‘NS,’’ ‘‘ZR’’ and ‘‘PS’’) compared with the nine values (from NV to PV covering all the LVs) for $g(e(t))$. As shown, nine different LVs (from ZR to MX) are used to give a gradual change in the output $u(t)$ for each combination of $e(t)$ and $g(e(t))$.

C. Defuzzification

For the defuzzification algorithm, the IntelRate controller applies the COG (Center of Gravity) method to obtain the crisp output with the equation $u(t) = (\sum_{j=1}^k c_j S_j) / (\sum_{j=1}^k S_j)$

S_j) [36], where k is the number of rules; c_j is the bottom centroid of a triangular in the output MFs, and S_j is the area of a triangle with its top chopped off as per $\mu_{P1}^m \cap P2^m$ discussed above. Since each parameter in the crisp input pair $(p1, p2)$ can take on two different values in the IntelRate controller, we have altogether $k = 4$ rules for defuzzification each time.

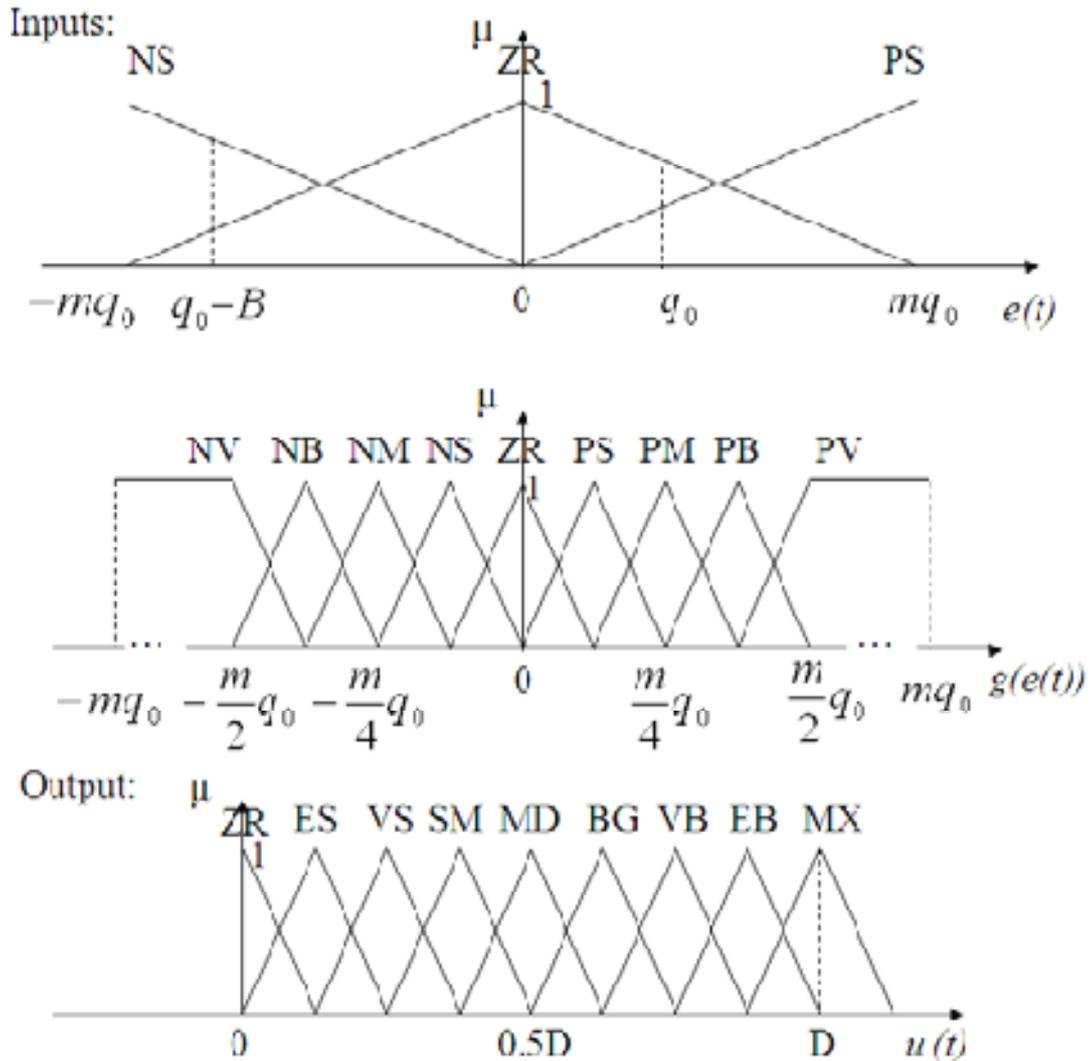


Fig. 3. Membership functions with FS.

D. The Control Procedure

Figure 4 shows the new field in the packet congestion header that we need to support our controller algorithm for the operation principle mentioned in Section II. As discussed, we need to include in the congestion header a new field called *req_rate* to carry the desired sending rate from the source and which will be continuously updated by the allowed sending rate when the packet passes each router.

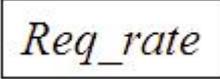


Fig. 4. Congestion header.

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

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

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

4. CONCLUSION

A novel traffic management scheme, called the IntelRate controller, has been proposed to manage the Internet congestion in order to assure the quality of service for different service applications. The controller is designed by paying attention to the disadvantages as well as the advantages of the existing congestion control protocols. As a distributed operation in networks, the IntelRate controller uses the instantaneous queue size alone to effectively throttle the source sending rate with max-min fairness. Unlike the existing explicit traffic control protocols that potentially suffer from performance problems or high router resource consumption due to the estimation of the network parameters, the IntelRate controller can overcome those fundamental deficiencies. To verify the effectiveness and superiority of the IntelRate controller, extensive experiments have been conducted in OPNET modeler. In addition to the feature of the FLC being able to intelligently tackle the non-linearity of the traffic control systems, the success of the IntelRate controller is also attributed to the careful design of the fuzzy logic elements.

5. REFERENCES

- [1] M. Welzl, *Network Congestion Control: Managing Internet Traffic*. John Wiley & Sons Ltd., 2005.
- [2] R. Jain, "Congestion control and traffic management in ATM networks: recent advances and a survey," *Computer Networks ISDN Syst.*, vol. 28, no. 13, pp. 1723-1738, Oct. 1996.
- [3] V. Jacobson, "Congestion avoidance and control," in *Proc. 1988 SIG-COMM*, pp. 314-329.
- [4] V. Jacobson, "Modified TCP congestion avoidance algorithm," Apr. 1990.
- [5] K. K. Ramakrishnan and S. Floyd, "Proposals to add explicit congestion notification (ECN) to IP," RFC 2481, Jan. 1999.
- [6] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. 2002 SIGCOMM*, pp. 89-102.
- [7] S. H. Low, F. Paganini, J. Wang, et al., "Dynamics of TCP/AQM and a scalable control," in *Proc. 2002 IEEE INFOCOM*, vol. 1, pp. 239-248.
- [8] S. Floyd, "High-speed TCP for large congestion windows," RFC 3649, Dec. 2003.
- [9] W. Feng and S. Vanichpun, "Enabling compatibility between TCP Reno and TCP Vegas," in *Proc. 2003 Symp. Applications Internet*, pp. 301-308.
- [10] [10] M. M. Hassani and R. Berangi, "An analytical model for evaluating utilization of TCP Reno," in *Proc. 2007 Int. Conf. Computer Syst. Technologies*, p. 14-1-7.
- [11] N. Dukkupati, N. McKeown, and A. G. Fraser, "RCP-AC congestion control to make flows complete quickly in any environment," in *Proc. 2006 IEEE INFOCOM*, pp. 1-5.
- [12] Y. Zhang, D. Leonard, and D. Loguinov, "JetMax: scalable max-min congestion control for high-speed heterogeneous networks," in *Proc. 2006 IEEE INFOCOM*, pp. 1-13.
- [13] B. Wydrowski, L. Andrew, and M. Zukerman, "MaxNet: congestion control architecture for scalable networks," *IEEE Commun. Lett.*, vol. 7, no. 10, pp. 511-513, Oct. 2003.
- [14] Y. Zhang and M. Ahmed, "A control theoretic analysis of XCP," in *Proc. 2005 IEEE INFOCOM*, vol. 4, pp. 2831-2835.
- [15] Y. Zhang and T. R. Henderson, "An implementation and experimental study of the explicit control protocol (XCP)," in *Proc. 2005 IEEE INFOCOM*, vol. 2, pp. 1037-1048.
- [16] K. M. Passino and S. Yurkovich, *Fuzzy Control*. Addison Wesley Longman Inc., 1998.

