A Genetic Algorithm for the Design of a Fuzzy Controller for Active Queue Management

Giuseppe Di Fatta, *Member, IEEE*, Frank Hoffmann, *Member, IEEE*, Giuseppe Lo Re, *Member, IEEE*, and Alfonso Urso, *Member, IEEE*

***Abstract*—Active queue management (AQM) policies are those policies of router queue management that allow for the detection of network congestion, the notification of such occurrences to the hosts on the network borders, and the adoption of a suitable control policy. This paper proposes the adoption of a fuzzy proportional integral (FPI) controller as an active queue manager for Internet routers. The analytical design of the proposed FPI controller is carried out in analogy with a proportional integral (PI) controller, which recently has been proposed for AQM. A genetic algorithm is proposed for tuning of the FPI controller parameters with respect to optimal disturbance rejection. In the paper the FPI controller design metodology is described and the results of the comparison with random early detection (RED), tail drop, and PI controller are presented.**

***Index Terms*—Active queue management, fuzzy controllers, genetic algorithms, network congestion, PI controllers.**

I. Introduction

In RECENT years, the unpredictable growth of the Internet has increasingly pointed out the congestion problem. The network congestion phenomenon is induced when the amount of data injected in the network is larger than the amount of the data which is delivered to destinations. The approach which represents the beginning of network congestion control is a so called end-to-end approach. In this approach responsive data sources reduce their transmission rate when they infer congestion occurrences from packet losses. This is, for instance, the approach adopted by the transmission control protocol (TCP). The effectiveness of a control system where sources are responsible for congestion control is based essentially on the assumption that most of the applications running through the network obey the control laws. End-to-end peers detect congestion level by inferring it from packet losses. A packet loss could mean that one of the intermediate routers does not have enough buffer space to store the packet before its transmission on the appropriate link toward the destination.

The simplest and currently most deployed policy adopted by routers to manage their queues, is a first-come-first–serve strategy, which is implemented by means of a first-in-first-out queue management. This policy, known as tail drop, has two main disadvantages [1]. It allows routers to keep full queues for long periods as it generates congestion notifications only when the queue is already full. As a consequence, packets may suffer high delays when they meet long queues. Moreover, sources may experience multiple losses in packet bursts which arrive at full queues. The second drawback of the tail drop policy is that few connections may monopolize the queue space and prevent other connections to get a share of the link bandwidth. Such a look-out phenomenon is mainly due to a global synchronization of flows. While the look-out phenomenon may be solved by simple policies such as “random drop on full” and “drop front on full,” more complex and proactive queue management strategies are required to solve the high delay problem due to full queues. To this aim, the adoption of an active queue management (AQM) strategy has been proposed [1]. In AQM congestion notifications are generated by dropping (or marking) incoming packets before router queues become full. In recent years, several different approaches have produced different AQM policies [2]–[5].

AQM policies, when properly used, provide better network utilization and lower end-to-end delays than tail drop. Nevertheless, many AQM policies may induce network instability if not properly configured. The tuning of their parameters may result extremely difficult and in some situations it is achievable only by means of heuristic methods or simulations.

However, most of the AQM policies are unable to maintain their performances in a wide range of operational conditions (e.g., number of connections, link capacity, propagation delay) [6].

In order to avoid such problems and provide better adaptation under different network conditions, nonlinear and adaptive AQM controllers have been proposed [7]–[9]. Among these, a congestion control algorithm based upon a fuzzy logic controller can be adopted [10] and [11]. The design of a fuzzy logic controller is not straightforward, because of the heuristic involved in control rules and membership functions. Currently there are no automatic methods for the design of the fuzzy knowledge base nor for the tuning of the fuzzy controller’s parameters. Therefore, the designers have to devise a fuzzy knowledge base using heuristic methods and experience. Accordingly, the parameters of a fuzzy control system are usually tuned by means of a trial and error method. An alternative procedure to set the parameters of a fuzzy logic controller is based on evolutionary algorithms. Evolutionary algorithms provide a universal optimization technique that imitates processes of genetic adaptation that occur in natural evolution [12] and [13].

Manuscript received July 1, 2002; revised March 25, 2003 and July 22, 2003. This work was supported by the Italian National Research Council. This paper was recommended by Guest Editors W. Pedrycz and A. Vasilakos.

G. Di Fatta, G. Lo Re, and A. Urso are with the Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Palermo 90128, Italy.

F. Hoffmann is with the University of Dortmund, D-44221, Dortmund, Germany (e-mail: g.difatta@icar.cnr.it; lore@icar.cnr.it; urso@icar.cnr.it; hoffmann@esr.e-technik.uni-dortmund.de).

Digital Object Identifier 10.1109/TSMCC.2003.818946

Unlike mathematically more rigorous optimization methods, they require, other than the objective itself, no particular knowledge about the problem structure, such as gradient information. This property makes them applicable to optimization tasks, in which the optimization function is evaluated through experiments or simulations rather than computed directly in closed form. Evolutionary algorithms develop a population of genetically encoded competing candidate solutions [12] and [13]. The next generation evolves from the current population by means of selection, recombination and mutation. The interplay of exploiting solutions that demonstrated superior fitness via selection and exploring the search space by recombination and mutation of known solutions constitutes the fundamental theme in evolutionary optimization. The different approaches, genetic algorithms (GA), evolution strategies (ES), evolutionary programming (EP), and genetic programming (GP), are distinguished by the genetic structures that undergo adaptation and the genetic operators by which they generate new variants.

In this paper, the adoption of a fuzzy proportional integral (FPI) controller is proposed for AQM. The FPI parameters are tuned by a genetic algorithm (GA). Several simulations are carried out to demonstrate the effectiveness of the proposed design methodology for AQM policies.

The remainder of the paper is structured as follows. Section II reviews the background on the AQM policies proposed in the literature. Section III describes in more details the PI regulator for AQM and the design of the FPI controller. Section IV deals with the optimal tuning of the FPI parameters by means of a genetic algorithm. In Section V the sets of experiments carried out are presented and the results discussed. Finally, the last section is devoted to the conclusions.

II. Active Queue Management

End-to-end congestion control [14] requires that flows reduce their transmission rate when they receive congestion notifications. TCP flows employ the additive increase multiplicative decrease (AIMD) algorithm to adjust the congestion window size and to determine a fair transmission rate. In the additive increase operation, a flow gradually and linearly increases its transmission rate to probe the status of the network. AIMD leads to high utilization of network resources and usually dominates the aggregate traffic dynamics. The relationship between throughput and congestion status (loss probability) for a single TCP flow has been widely studied (e.g., [15]). It has been shown that the throughput is inversely proportional to the round trip time and to the square root of the packet loss probability. Unresponsive flows which do not react to congestion indications, make any congestion control mechanism ineffective. Unfortunately, currently there is no solution for those aggressive flows. Although they represent a minority of only 10–20% of Internet traffic and no overall change in the TCP/UDP traffic ratio has been measured [16], unresponsive flows may soon become an important issue to be faced in order to guarantee the stability of the Internet.

The traditional technique to manage router queue lengths is to drop incoming packets when the queue is full (tail drop). This policy is very simple to be configured; the maximum queue size is often set to the delay-bandwidth product. Although tail drop is widely adopted in the Internet, it presents important drawbacks, such as long delays experienced by packets and the lock-out phenomena [1]. To eliminate the tail drop disadvantages and to anticipate the source answers to incipient congestion situations, AQM policies [1] have been proposed. AQM uses the principle of the feedback of congestion to end hosts by the mechanism of dropping/marking packets at router queues. The end hosts then react to the dropping/marking of packets by reducing their trasmission rate. Consequently, the queue length in routers is reduced and the end-to-end delay experencied by packets is also reduced. Moreover, this mechanism ensures a more efficient use of the network resources by reducing the loss of packets occurring when queues overflow. The first proposed AQM policy was random early detection (RED) [2]. RED adopts the preemptive dropping, or marking, of packets when the average queue length ranges between a minimum and a maximum threshold. More precisely, the probability of packet dropping/marking increases linearly between the minimum and maximum thresholds, and equals zero for average queue lengths below the minimum. All packets are dropped if the average queue length increases above the maximum threshold.

In the last several years, objections have been raised against RED [17]–[19], among which the difficulty of properly setting its parameters according to network conditions, and the dependence of the queue length in steady state from the number of flows. Namely, a growth in the number of flows increases the average queue length, and, when it exceeds the maximum threshold, all packets are dropped. However, it is not possible to increase too much the maximum threshold, because this would mean higher queueing delays. On the other hand, if the maximum threshold is set to a low value, severe buffer oscillations would occur. Many variants of RED have been proposed to resolve these problems [20]–[23].

Furthermore, different approaches than RED have been recently proposed; among these random exponentially marking (REM) [4], adaptive virtual queue (AVQ) [5], sliding mode variable structure control [9], and proportional integral (PI) controller [3]. All these AQM techniques get over the heuristic approach of RED and its variants, and they are based on a control theoretic approach. REM is the result of a linear modeling of the problem and of its resolution in its dual form. Differently from RED, REM distinguishes between the congestion measure of each router and the dropping probability. In REM the authors introduce a measure called price which eliminates the dependence of the dropping probability from the current value of the queue size. Their algorithm uses the current queue size and its difference from a desired value to calculate the dropping probability accordingly to an exponential law. Such a feature owns the additivity property, so thus a source can calculate the price of a whole path from the total number of the dropped packets along the path itself.

The adaptive virtual queue proposes a methodology for finding the fastest rate at which the marking probability adaptation can take place, given certain system parameters like the maximum delay and the number of sessions, so that the system remains stable. The sliding mode variable structure control is a policy based on the advanced theory of robust

control; it is designed with the aim to obtain a controller which is robust against disturbances and variance of parameters. Finally, proportional integral (PI) controller and REM, although obtained independently, and following full different theoretical approaches, represent, accordingly to their authors, the same solution [3]. In Section III, a review of PI controller for AQM is presented and a methodology to design a fuzzy proportional integral (FPI) controller for the management of router queues is developed.

III. PI AND FPI CONTROLLERS

The PI controller proposed in [3], is based on the observation that an AQM algorithm controls network congestion by randomly dropping or marking packets at the router queue with probability $p(t)$ depending on the queue error defined by:

$$e(t) = q(t) - q_r \tag{1}$$

where q_r is the reference queue size and $q(t)$ is the current queue size. When TCP sources detect congestion by packet losses, they reduce their transmission rates and the queue size of the router decreases. This process constitutes a closed loop feedback control system where the controlled variable is the queue size and the control variable is the dropping/marking probability. In general, a PI controller consists of a proportional and an integral term as follows:

$$p(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau \tag{2}$$

where K_p and K_i are, respectively, the proportional gain and the integral gain [24]. The PI properly regulates the queue size when it operates around the reference queue size. The goal is to minimize disturbances in queue size caused by variations of incoming flows. In order to design the the PI AQM controllers, a nonlinear dynamic model for TCP/AQM has been developed in [25]. Once the model is linearized around an operating point, a stable PI linear controller is designed in order to satisfy the specifications. The authors in [3] show that in the PI controller the proportional part is equivalent to RED when the input low-pass filter is removed. The usage of a proportional controller leads to a lower time of response but also to lower stability margins; moreover, the proportional controller has a steady state regulation error, where such an error is defined as the difference between the steady state output queue and the reference value. In order to overcome the above disadvantages, the integral term is added which has the characteristic to give steady state error equal to zero and to give higher stability margins.

Nevertheless, the PI AQM policy suffers the disadvantage that it is unable to maintain good dynamic performances as the number of TCP flows increases. This disadvantage is essentially due to the fact that the controller design is carried out considering a particular operating point, i.e., particular values of number of TCP flows (N), round trip time (RTT) and link capacity (C). When these values vary, for example the number of TCP flows increases, the high frequency gain of the open loop transfer function decreases and the system bandwidth becomes lower, which implies a slower system in terms of rise and settling time.

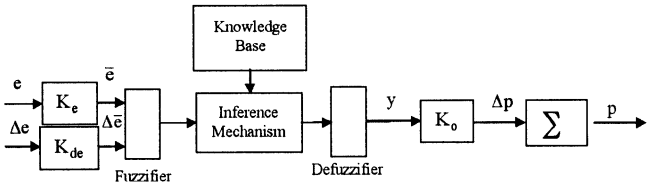


Fig. 1. Structure of the FPI controller.

The proposed fuzzy logic controller is aimed to overcome this disadvantage. In recent years, fuzzy logic controllers, especially FPI controllers, have been widely used for process control owing to their heuristic nature associated with simplicity and effectiveness for both linear and nonlinear systems [26]–[28]. In fact, for single-input single-output systems, fuzzy logic controllers can be essentially seen as PI type associated with nonlinear gain. In most situations, FPI controllers can achieve better system performances than the conventional PI controllers, because of their nonlinear properties.

However, a method which allows to obtain the parameters of the fuzzy controllers is needed. Systematic methods for the design of the FPI scaling factors have been developed in [14], [29]–[31], which take advantage from the analogy between FPI and conventional PI controllers. More precisely, a set of relations between the scaling factors and the gains of PI controller are obtained from such an analogy. Then, the gains of PI controller are determined so as to satisfy, for instance, requirements on bandwidth of the control loop. Finally, scaling factors are computed from the gains of PI controller.

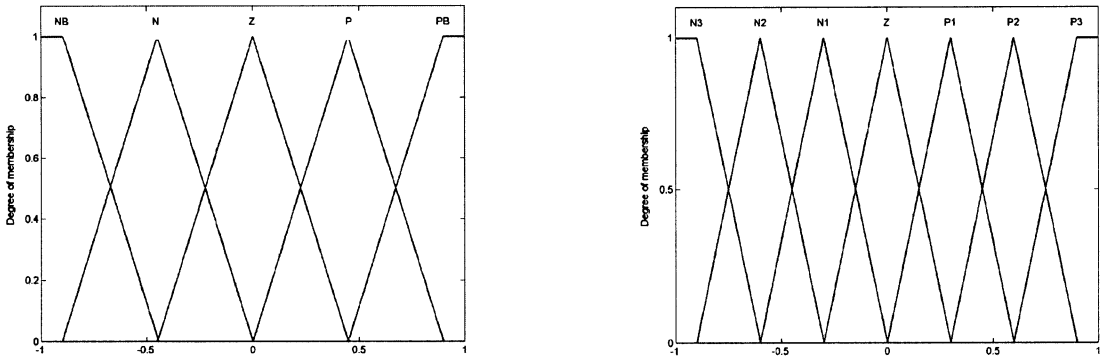
The fuzzy controller proposed in this paper is a PI-type fuzzy controller, as depicted in Fig. 1, where all quantities are considered at the generic discrete instant kT_s and where T_s is the sampling period; $e(kT_s) = q(kT_s) - q_r(kT_s)$ is the error on the controlled variable q (queue size); $\Delta e(kT_s) = e(kT_s) - e((k-1)T_s)$ is the variation of the error; $\Delta p(kT_s)$ is the increment of the control variable p (probability of packet marking/dropping); K_e, K_{de}, and K_o are the scaling factors.

The sampling period T_s is set to 6.25 ms, which is by far large enough to complete a fuzzy inference cycle, and small enough to respond to fluctuations of incoming flows. In [32], the authors report on a programmable analogue fuzzy controller implemented in CMOS technology, capable of performing 5.26 million fuzzy inferences p/s. Therefore, the queue sampling period could be reduced to $T_s < 1\ \mu$s for high-volume routers on the Internet backbone.

It should be noticed that these fuzzy controllers have an intrinsic PI action. As a consequence, the steady state behavior at the operating point is identical for both controllers.

The relations between the parameters of the fuzzy controller and those of the standard PI one have to be determined, and a method of synthesis of the standard PI controller has to be chosen.

We adopt the dynamic model of TCP behavior using fluid-flow and stochastic differential equation analysis developed by Hollot *et al.* [25] in order to synthesize the standard PI controller. Furthermore, the synthesis of the PI controller is



(a) (b)

Fig. 2. Membership functions of (a) input and (b) output variables.

carried out following the guidelines to design stable controllers given in [3].

Finally, we derive the relations between the gains of the PI controller and the scaling factors of the FPI one. Let us assume that a set of rules and membership functions on normalized universes of discourse has been assigned. The fuzzy controller generates a nonlinear map of the form

$$y = f(\bar{e}, \Delta\bar{e}) \tag{3}$$

and the relation $p - y$ is that of a discretized integrator of gain K_o. The corresponding standard PI regulator can be seen as a linear map between e, Δe and Δp defined by

$$\Delta p = K_i T_s e + \left(K_p + \frac{K_i T_s}{2}\right)\Delta e \tag{4}$$

assuming trapezoidal integration. The constants K_p and K_i determine the gains of the PI regulator.

Moreover, it is possible to approximate the map (3) with the plane

$$y = K'_e \bar{e} + K'_{de}\Delta\bar{e} \tag{5}$$

where K'_e and K'_{de} are obtained by minimizing the functional

$$J = \int\int_{-\epsilon}^{\epsilon} \left(K'_e\bar{e} + K'_{de}\Delta\bar{e} - f(\bar{e}\Delta\bar{e})\right)^2 d\bar{e}d\Delta\bar{e} \tag{6}$$

and where the parameter ϵ defines a well suited (small) interval around the operating point. A least square solution for the minimization of the above index it is possible by the discretization of the integral in (6). Finally, the output Δp of the PI controller (4) is made equal to the output of linear approximation (5) scaled by K_o. Therefore, the final relationship between the fuzzy gains K_e, K_{de}, K_o, and the standard PI gains K_p, K_i is given by

$$\begin{cases} K_p = K_o K'_{de} K_{de} - \dfrac{K_i T_s}{2} \\ K_i = \dfrac{K_o K'_e K_e}{T_s} \end{cases}. \tag{7}$$

The two equations in (7) only constrain two of the three fuzzy scaling factors. The extra degree of freedom is used to determine the parameter K_o, so that the control variable Δp remains within its limits.

The fuzzy knowledge base is composed of five triangular membership functions for the input fuzzy variables and seven membership functions for the output fuzzy variable. All variables are defined in a normalized ($[-1, 1]$) universe of discourse as depicted in Fig. 2. The specific control behavior is mainly determined by the fuzzy membership functions, scaling factors and fuzzy rules. A fuzzy rule associates the output specified in the rule conclusion with an input region defined by the rule antecedent. The fuzzy inference scheme smoothly interpolates the output among those rules whose antecedent matches the current input. The resulting fuzzy map is mainly determined by the locations of the input and output membership functions and the rules that associate them. The locations of input membership functions, defined by their center points (peak values), determine the locations of interpolation points. The center points of output membership functions determine the height of the control map at these interpolation points. Tuning the membership function center points is a mean of locally shaping the control surface. Changing the scaling factors, globally effects the control surface, as it modifies the gain across the entire input space. In [33], the authors argue that genetic tuning of fuzzy controllers follows a coarse to fine order. Adaptation of scaling factors is most significant as it globally influences the control surface, tuning of membership functions operates at a medium scale and adjusting the output of individual fuzzy rules is considered as fine tuning. We, therefore, do not consider other types of membership functions, as scaling factors and center points already largely determine the control behavior.

For complex control problems the definition of the rules requires the knowledge or experience of experts. For a proportional integral fuzzy controller, with error and error rate as inputs, the design of the rule matrix is straightforward. The control output compensates the observed error, in a way that the change in dropping rate increases with increasing error and error rate.

In previous works on fuzzy controller for AQM [10] and [11] the choice of fuzzy rules and the parameters tuning process starts from considerations on the specific AQM problem and is progressively optimized in a trial and error method.

Here, a standard set of rules is chosen (cf. Table I) and the optimal tuning of the FPI controller parameters by means of a GA is carried out. In this context, (7) are used to define the scaling factors range of variation during the evolutionary optimization.

Section IV discusses the optimal tuning of the FPI controller parameters by means of a GA.

IV. Evolutionary Optimization

A. Genetic Algorithms

A GA operates on a population of competing candidate solutions. The genome, often represented as a binary string, encodes a set of parameters mapped into a potential solution to the optimization problem. A scalar fitness function evaluates the quality of a solution with respect to the optimization task at hand. According to Darwin's principle, individuals superior to their competitors are more likely to promote their genes to the next generation. Recombination and mutation are applied to the parent genomes in order to generate new variants. The current population is replaced by the newly generated group of offsprings, which forms the next generation. The genetic algorithm terminates after a previously defined maximum number of generations elapsed. The best individual that emerged over the entire span of generations is retained as the final solution. A GA requires the following major components in order to solve a particular optimization problem:

1) genetic representation of candidate solutions;
2) scalar fitness function which describes the quality of each individual;
3) genetic operators that generate new variants during reproduction;
4) GA parameter settings, such as, population size, number of generations, and probabilities of applying genetic operators.

In our application, the genome encodes parameters of the fuzzy knowledge base, such as scaling factors and peak values of membership functions. These controllers for AQM are evaluated in a simulation of the network traffic on a single bottleneck topology. The fitness function tries to minimize the mean square error between the desired and actual queue length at the outgoing queue of the bottleneck link.

B. Genetic Optimization of the FPI Controller

The objective of a genetic fuzzy system [34] and [35] is to automate the knowledge acquisition step in fuzzy system design, a task that is usually accomplished by interviewing or observing of a human expert controlling the system. The design of a fuzzy rule based system is equivalent to find the optimal configuration of fuzzy sets and/or rules, and in that sense can be regarded as an optimization problem. The optimization criterion is the problem to be solved at hand and the search space is the set of parameters that code the membership functions, scaling factors and fuzzy rules. An evolutionary algorithm adapts either part or all of the components of the fuzzy knowledge base.

Genetic tuning processes are based on the assumption that the rule base has been already designed, either by a human expert or by a prior learning process. The controller is based on the standard rule matrix shown in Table I. Tests on specifically designed rule matrixes did not show significant performance improvements compared to the standard rule matrix.

The tuning process optimizes the performance of a fuzzy controller with predefined rule base. The objective is to identify FPI controller parameters that best achieve the desired control behavior. The tuning process involves the adaptation of the fuzzy database, namely center points of membership functions and scaling factors for input and output variables.

TABLE I
Fuzzy Rules

de \ e	NB	N	Z	P	PB
NB	N3	N3	N2	N1	Z
N	N3	N2	N1	Z	P1
Z	N2	N1	Z	P1	P2
P	N1	Z	P1	P2	P3
PB	Z	P1	P2	P3	P3

A population of competing chromosomes that encode the tuning parameters evolves by means of selection, recombination and mutation. The fitness of a fuzzy controller parameterized by membership functions and scaling factors is determined according to its observed performance in regulating the router queue. The fitness function rewards controllers that minimize the mean square error (MSE) under different traffic loads.

Over the course of evolution the algorithm identifies those set of parameters, for which the fuzzy controller performs optimally with respect to the given performance index.

We define an overall performance index F of the system which depends on the FPI controller parameters as follows:

$$F = f(PMF's, SF's) \tag{8}$$

where $PMF's$ are peak values (center points) of the triangular membership functions and $SF's$ are the scaling factors of the FPI controller. Usually the performance index depends on the design parameters of the FPI in a nonlinear fashion and standard optimization algorithms, such as gradient-descent optimization, may become trapped in local minima. The choice of a genetic algorithm for the optimal tuning of the FPI controller parameters has been made in order to avoid this drawback. By mimicking principles of the natural evolution, GA are able to find feasible solutions to real-world optimization problems. GAs are not considered as mathematically rigorous optimization algorithms, as they do not require the computation of derivatives of the optimization function. Instead GAs are based on a discrete, stochastic, process of exploration and exploitation. The obtained solution emerges from the condensation of partially adapted individuals in previous generations, of which the favorable attributes are carried forward into the following generation. The main advantage of the GA is its capability of achieving global optimization solution even for nonlinear, high-dimensional, multimodal, and discontinuous problems [13].

Our optimization procedure is based on the "simple" GA developed in [36]. The ten real-valued parameters subject to optimization are encoded by a binary chromosome of length $10 \times 8 = 80$ bits. The optimization process is driven by the maximization of the fitness function F, defined as the inverse of the mean square error between the current value of the queue length and the desired reference queue length

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}[q_r - q_{(i)}]^2 \tag{9}$$

where n is the total number of the queue values sampled at discrete time intervals.

The primary goal of the controller design is to reject disturbances in the queue caused by overloading the capacity of the node. The disturbances emerge from fluctuations in the rate at which packages arrive at the node, and typically appear on different magnitudes and time scales. One can distinguish between two basic scenarios, a steady state situation in which the input rate does not exceed the link capacity, and a situation in which the incoming flows exceed the node's bandwidth. The FPI controller is supposed to operate equally well in both operating regimes. The design goal is to minimize the MSE. As the magnitude of errors differs significantly between both scenarios, the MSE for steady state behavior MSE_s and for responsiveness to overload MSE_r are observed separately. The MSEs are then normalized by the equivalent indices $\mathrm{MSE}_r(PI)$ and $\mathrm{MSE}_s(PI)$ of the PI controller. The total $\mathrm{MSE}_{\mathrm{Tot}}$ is calculated as the average between the normalized MSEs.

$$\mathrm{MSE}_{\mathrm{Tot}} = \frac{1}{2}\left(\frac{\mathrm{MSE}_s\,(\mathrm{FPI})}{\mathrm{MSE}_s\,(\mathrm{PI})} + \frac{\mathrm{MSE}_r\,(\mathrm{FPI})}{\mathrm{MSE}_r\,(\mathrm{PI})}\right). \tag{10}$$

The term MSE_s measures how well the FPI suppresses small signal disturbances under steady state conditions. The term MSE_r takes into account how the FPI responds in highly dynamic situations, such as link capacity overloads caused by a sudden increase in the number of flows.

The overall fitness which the GA aims to maximize is computed as the inverse of the averaged $\mathrm{MSE}_{\mathrm{Tot}}$

$$F = \frac{1}{\mathrm{MSE}_{Tot}}. \tag{11}$$

V. Implementation, Tuning and Performance Evaluation

A. Simulation Setup

We have implemented and tested the fuzzy controller as an active queue manager under the well known $ns2$ simulator [37]. Tail drop, RED, and the PI (according to the pseudo-code reported in [3]) are also evaluated in this environment under identical test conditions. The single-bottleneck network topology adopted in the parameters tuning process and in the first set of experiments is reported in Fig. 3. The single-bottleneck topology consists of traffic generator nodes, connected to traffic sinks by means of three hops path. The AQM controllers are installed in the router accessing the bottleneck link. The network load is generated by FTP and HTTP sources. HTTP flows are short lived flows with a bursty behavior. They cannot be easily controlled by the congestion control mechanism and are adopted as noise. FTP flows are characterized by an intensive data transfer and represent the traffic load to be controlled in the fluid flow dynamic model. All the flows are conveyed in the 16 Mb/s bottleneck link between the AQM router R_1 and the router R_2. The routers HS, FS, HT, and FT are introduced to measure the different FTP and HTTP input and output traffic in the bottleneck link. The propagation delay of the paths between sources and destinations is uniformly distributed in the range 80–120 ms; the average round trip time is 200 ms. The maximum queue length in the AQM router is set to the delay-bandwidth product, i.e., 800 500-byte packets for a 16 Mb/s link and average RTT of 200 ms. The parameters of PI controller are determined for operating conditions where the number of FTP flows is 60 and the desired value of the queue length is 200 packets. Section V-B describes the tuning of the parameters of the FPI controller by means of a genetic algorithm.

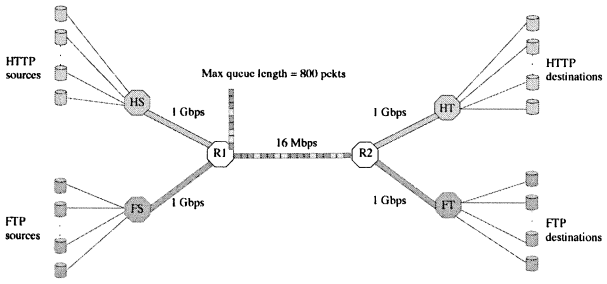


Fig. 3. Experimental topology.

B. GA-Based FPI Parameters Tuning

This set of experiments is devoted to the tuning of FPI controller parameters (peak values of membership functions and scaling factors) by means of the genetic algorithm described in Section IV. The first step is to identify a suite of representative benchmark scenarios for which the genetic algorithm evaluates the controller performance. To this regard, the main design objectives of the FPI AQM controller are: 1) disturbance rejection in steady state conditions; 2) dynamic response to load fluctuations. The simulation experiments are designed in a way that reflects these objectives. The fitness of a controller is evaluated for a constant load of 60 FTP flows over the entire simulation period (from 0 to 70 s) and a second group of 480 additional FTP flows which transmit in the time interval between 30 and 50 s. This particular scenario is motivated by the requirement that both steady-state and dynamic behavior enter the performance index. Moreover, the range of the number of flows for which it is interesting to evaluate the behavior of a queue management policy is related to the network operating conditions as follows:

$$\begin{cases} N_{\min} = \dfrac{C \cdot RTT}{8 \cdot pcks \cdot cw_{\max}} \\ N_{\max} = \dfrac{C \cdot RTT}{8 \cdot pcks} \end{cases} \tag{12}$$

where C is the link capacity, RTT is the average round trip time, pcks is the average packet size and $\mathrm{cw}_{\max}$ is the max advertised TCP window. When the flows are $N_{\min}$, maximum transmission rate at sources can reach full link utilization; when the flows are $N_{\max}$ the link capacity C allows each of $N_{\max}$ sources to send only 1 packet every RTT. Under the operating conditions of $RTT = 200$ msec, $C = 16$ Mb/s, $\mathrm{pcks} = 500$ byte and $\mathrm{cw}_{\max} = 20$ packets, the range $[N_{\min} - N_{\max}]$ is [40–800] flows. In this case, the choice of an overload of 480 flows represents a fair trade-off between the design goals of fast response to load variations that exceed the links bandwidth and optimal operating conditions in case link utilization is sufficient for all sources. The GA parameters are set as follows:

- number of generations = 80;
- population size = 40;
- cross probability $PC = 0.9$;
- mutation probability $PM = 0.004$.

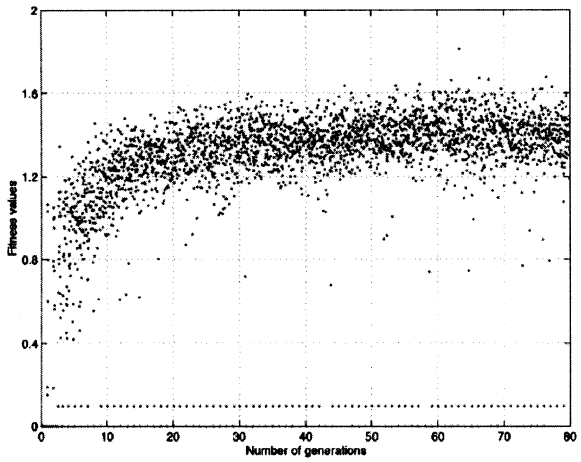


Fig. 4. Fitness values.

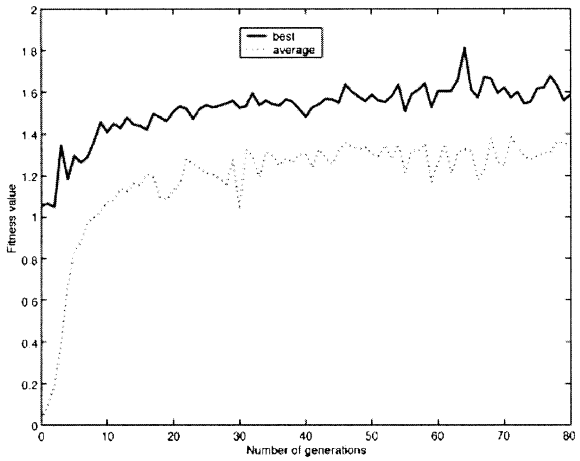


Fig. 5. Average and best fitness values.

The population size and mutation probability depend on the chromosome length. The mutation rate is chosen such that the per individual mutation probability is smaller than one. No attempt has been made to identify optimal GA parameters in advance. The population size, crossover probability and mutation rate are determined according to the heuristics in [38]. The number of generations is constrained by computational resources for running the network simulation. As the GA evaluates the individuals in one generation in parallel, the population size was chosen according to the number of available workstations.

The simulations were carried out on a forty nodes Linux workstation cluster, such that the fitness of controllers is evaluated in parallel. Figs. 4 and 5 show the evolution of fitness for the entire population and the best and average fitness. As shown in Fig. 5, the fitness of the population stabilizes after about 40 generations. Table II shows parameters of the FPI controller which represent the optimal solution generated by the GA.

In order to validate the choice of an overload of 480 flows, evolutionary runs have been carried out for additional overloads of 60, 120, 240, and 600 flows. The results of these evolutions are summarized in Table III. The evolutionary run with 480 additional FTP flows gives better performance both in terms of overall best fitness value and in terms of average fitness value in the last generation.

Finally, the nonlinear surface corresponding to the optimal FPI controller generated by the GA is shown in Fig. 6. In Section V-C we evaluate the performance of the optimal FPI controller whose parameters are given in Table II.

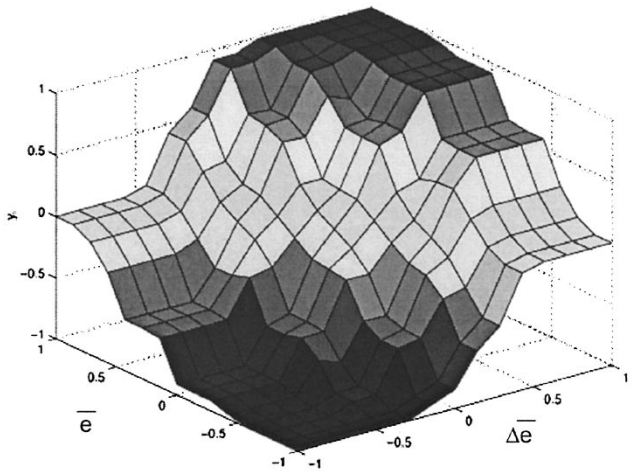


Fig. 6. FPI surface.

C. FPI Controller Evaluation

A queue management policy should match input rate to link capacity to maximize utilization and to reduce packet queuing delay at a bottleneck link. Full queue should be avoided even if they provide high utilization. Buffer capacity is needed to absorb packet bursts. Congestion typically happens when the number of connections which share the link bandwidth increases. In general, when the number of connections changes, AQM task is to redistribute link bandwidth among the connections, thus each connection can get a fair share.

The main criteria which can be adopted to evaluate AQM policies are efficiency, fairness, regulation and responsiveness. Efficiency is measured both as link utilization, in terms of the ratio between the aggregate throughput and the link bandwidth, and a low queuing delay. When multiple connections share link and buffer resources a measure of fairness quantifies how equally the resources are distributed and reveals if there are connections which are discriminated. The first design objective is queue length regulation, which is the performance criterion based on which the GA tunes the FPI controllers. The system should reach a steady state in minimal time and subsequent fluctuations in queue length around the reference value should be small. Responsiveness is defined by the rate at which the queue stabilizes at the reference value after an overload has been introduced. It can be observed in the controlled variable (queue length) as well as in the controller output (dropping probability).

Finally, any AQM policy, when used for responsive flows, should minimize the number of dropped packets in routers.

1) Utilization, Queue Length, and Dropped Packets: In this set of experiments a first group of 60 FTP flows (*design conditions*) and 180 HTTP flows (*noise*), which last the whole simulation time (from 0 to 120 s), are adopted. Three different sets

TABLE II
OPTIMAL VALUES OF THE FPI CONTROLLER PARAMETERS

$PB(e) = -NB(e)$	$P(e) = -N(e)$	$PB() = -NB(\Delta e)$	P(Δe)=-N(Δe)	P3=-N3	P2=-N2	P1=-N1	K_o	K_e	K_{de}
0.47	0.26	0.81	0.45	1.0	0.97	0.78	0.000379	0.00049	0.06

TABLE III
SUMMARY OF FITNESS RESULTS

Overload flows	60	120	240	480	600
Best fitness	1.229	1.171	1.532	1.813	1.771
Avg. fitness (last gen.)	0.929	0.917	1.184	1.342	1.327

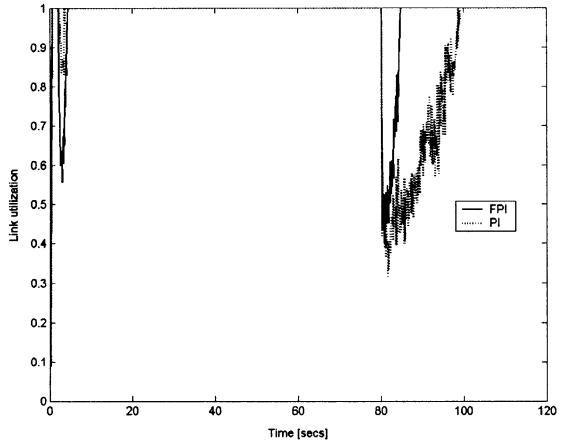


Fig. 7. Link utilization.

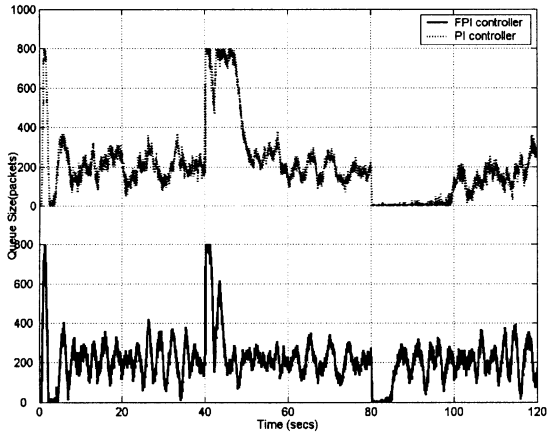


Fig. 8. Queue length.

of experiments are carried out, in order to show the utilization of the link bottleneck and to verify the behavior of the optimal FPI controller (Table II) under operating conditions that differ from the design ones. The average link utilization for different AQM policies has been measured under different conditions (Table IV), respectively, under heavy load fluctuations, different RTTs, and different link capacity. In the first set of simulations an additional group of FTP flows (*overload*) transmits data in the time interval between 40 and 80 s. In the second set the round trip time of FTP and HTTP flows has been uniformly distributed in different ranges in the interval 80–480 ms. In the third set the bottleneck link capacity has been changed. All AQM policies show high utilization in almost all cases; nevertheless, the FPI controller exhibits a better performance under heavy overload in comparison to the PI controller. The difference between the FPI and the PI controllers has been further investigated in the particular case of additional 480 FTP flows. Fig. 7 shows the aggregate throughput at the bottleneck link. An underutilization period shows up at $T = 80$ s when the additional FTP flows stop transmitting data. However, the underutilization phenomenon is significantly shorter in FPI than in PI controller. The better responsiveness of the FPI is confirmed in the queue length (Fig. 8) and the dropping probability (Fig. 9). Finally, in the analysis of the cumulative number of dropped packets (Fig. 10) the two controllers have a similar behavior till 80 s. In the final part of the simulation, although the link underutilization, less responsiveness of PI determines a higher number of dropped packets than FPI. The expected faster response time of the FPI controller has been confirmed under heavy load fluctuations.

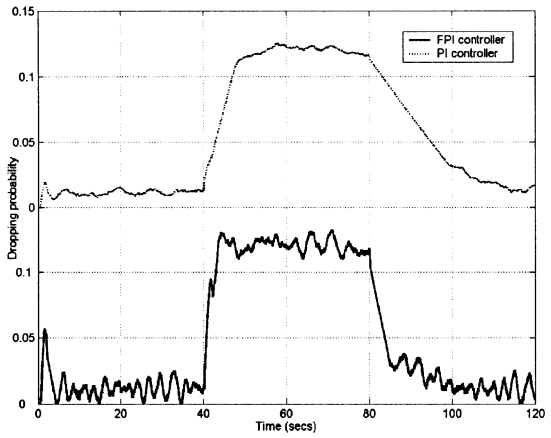


Fig. 9. Dropping probability.

2) Fairness: A fair AQM policy would be one which does not penalize any flow arbitrarily. In general, fairness implies that each competing flow gets the same share of the congested resource. Nevertheless, AQM policy is not the only responsible for fairness toward flows. In the congestion control and avoidance algorithm of TCP Reno there is an unavoidable source of unfairness: the transmission rate is inversely proportional to round trip time. This means that two TCP flows which experience the same congestion (loss probability) will achieve different average transmission rates according to their own round trip times. Another fairness issue is the discrimination of long-lived (FTP)

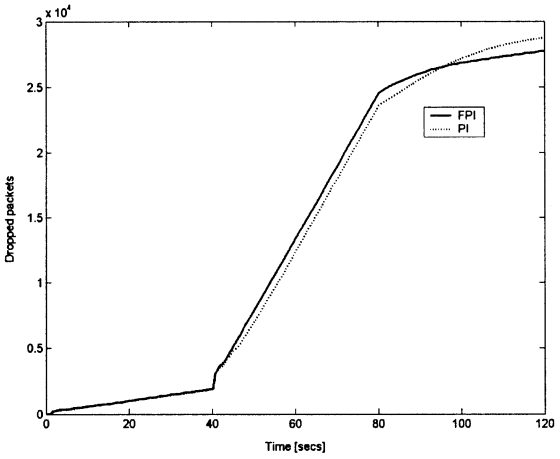


Fig. 10. Cumulative number of dropped packets.

and short-lived (HTTP) flows. Short TCP flows are generally more conservative than long flows and tend to get less than their fair share when they compete for the bottleneck bandwidth [39]. Furthermore, when a TCP flow traverses more congested links, it experiences larger end-to-end loss probability and, thus, gets less bandwidth. In the latter case the flow does consume more resources and it is arguable if this case is unfair.

We adopt the Jain's fairness index [40] to investigate fairness attributes of the FPI-based AQM policy in relation to round-trip delay bias and HTTP/FTP discrimination. Given a set of measures $\{x_i\}$ of which we want to evaluate the fairness distribution, Jain's index is defined as

$$F(x) = \frac{\sum_{i=0}^{N} [x_i]^2}{N \sum_{i=0}^{N} [x_i^2]}. \tag{13}$$

The fairness index (13) is applicable to any resource sharing or allocation problem. It is independent from the total amount of the resource and is a continuous function always bounded between 0 and 1. An index value of 1 means that the distribution of x_i values is totally fair, and a value of 0 means it is totally unfair.

We ran two simulations to evaluate the fairness among FTP flows with different round trip times and the fairness between FTP and HTTP flows of different AQM policies.

In the first simulation 100 FTP flows with different round trip times uniformly distributed in the range 60–900 ms share the single bottleneck of topology in Fig. 3. In [41] it is shown that 85% of the RTT values in the Internet lies in that range.

In the second simulation 60 FTP flows and 180 HTTP flows share the bottleneck link and both groups have RTT uniformly distributed in the range 160–240 ms.

We compute the fairness index of the average throughput $(F(t))$ for each flow during the entire simulation (50 s). Furthermore, we compute the fairness index of the overall packet drop rate $(F(p))$ for each flow. In the latter case x_i equals the drop packet rate of flow d_i divided by the fair dropping probability (total number of drops over total number of packets arrived at the bottleneck link). Table V shows the values of the fairness index obtained with different AQM policies. Both PI and FPI controllers provide better fairness than RED and tail drop policies in all cases. In the "FTP/HTTP" case, even though the index $F(t)$ indicates a low fairness in the throughput distribution for all policies, $F(p)$ shows that PI-based policies do not discriminate HTTP flows in packet dropping as much as RED and tail drop.

TABLE IV
SUMMARY OF LINK UTILIZATION RESULTS

		FPI	PI	RED	Tail Drop
overload	60	0.988	0.988	0.995	0.995
	120	0.986	0.978	0.995	0.995
	240	0.984	0.962	0.996	0.996
	480	0.979	0.932	0.992	0.992
	600	0.974	0.916	0.996	0.996
rtt	80/160	0.997	0.998	0.999	0.999
	160/240	0.992	0.992	0.991	0.991
	240/320	0.984	0.986	0.992	0.992
	320/400	0.974	0.974	0.985	0.985
	400/480	0.963	0.969	0.978	0.979
c_link	8	0.996	0.997	0.998	0.998
	16	0.992	0.992	0.991	0.991
	24	0.987	0.988	0.993	0.993
	32	0.984	0.987	0.989	0.989

TABLE V
FAIRNESS INDEXES $F(t)$ AND $F(p)$

Simulation	index	Tail Drop	RED	PI	FPI
RTT	F(t)	0.938	0.947	0.965	0.976
	F(p)	0.589	0.591	0.953	0.957
FTP/HTTP	F(t)	0.529	0.527	0.536	0.540
	F(p)	0.491	0.510	0.833	0.839

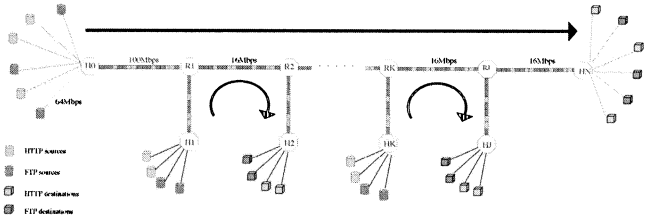


Fig. 11. Parking lot topology.

3) Multibottleneck Topology: The aim of this experiment is to investigate more complex scenarios where TCP flows traverse several AQM managed routers. We adopt a more complex topology, illustrated in Fig. 11, and more realistic traffic patterns in order to evaluate the performance of multiple AQM controllers. All the horizontal links are managed by AQM controllers and have the same capacity of 16 Mb/s, whereas the vertical links of 100 Mb/s carry aggregated traffic to feed the formers. The traffic which traverses all links in the straight path of the topology (the straight arrow in the figure), is constituted by

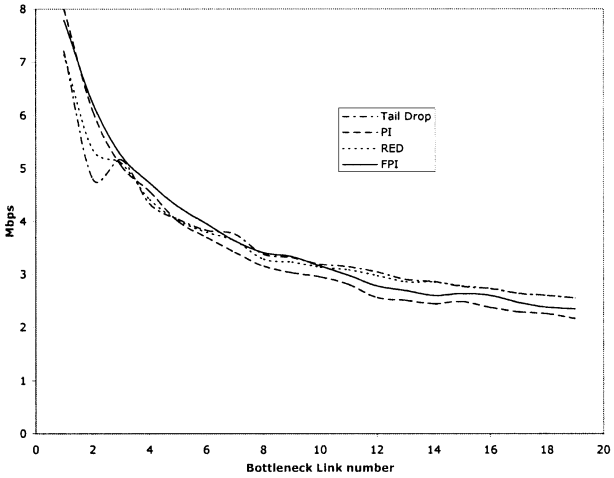


Fig. 12. Overall throughput versus the number of bottleneck links.

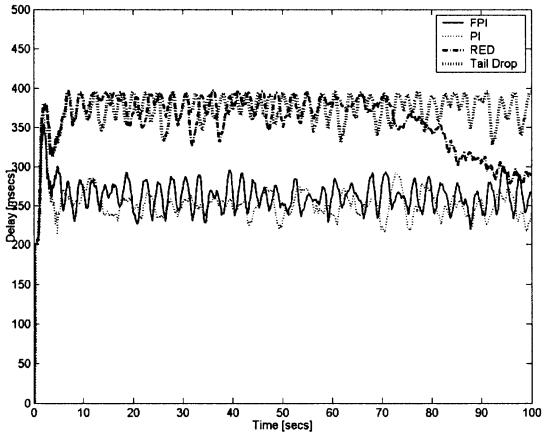


Fig. 13. Average delay with 20 bottleneck links.

sixty greedy FTP flows and sixty HTTP flows used as underlying noise. The same flow composition has also been used to produce the traffic crossing each single bottleneck, illustrated by the curved arrows. Propagation delays have been assigned to the topology links, such that the overall RTT for every flow ranges in the interval $[160, 240]$ ms, independently from the number of links in the path. In order to simulate realistic traffic, we set FTP session arrivals of the crossing flows according to a Poisson distribution [42]. The mean value of the exponential distribution is uniformly distributed in the time interval of the experiment period. Moreover, we used a Pareto distribution to set the FTP session lengths with average of 10 KB according to [43].

Fig. 12 shows the overall throughput achieved by the main group of FTP flows when the number of bottleneck links varies in the range $[1, 20]$. The performances of FPI, PI, RED, and tail drop AQM policies are compared. All the AQM policies show a similar behavior. When the number of traversed links increases the overall throughput achieved by the aggregated flow decreases. However, this can be considered a desirable behavior, since flows which use more valuable resources, such as congested links, should be more penalized.

Fig. 13 shows the average delay experienced by the packets of the main sixty FTP flows, when a parking lot topology with 20 edges is considered. As expected, the delays produced by the PI and FPI controllers are sensibly lower than the ones caused by RED and tail drop AQM policies. The figure also clearly shows the slow convergence time of RED. Finally, in order to investigate the dynamic of a multibottleneck topology, we set an experiment with a four-parking-lot topology. The crossing flow groups are composed by 40 FTP flows and the number of FTP flows in the main group is varying in the range between 20 and 100. Table VI summarizes the utilization of each link achieved by the aggregate FTP flows for FPI and PI controllers. The first column reports the number of FTP sessions which constitute the horizontal group of flows and the second one indicates the bottleneck link number. The overall percentage utilization of each link and the goodput of the main group of flows along the four bottlenecks path have been reported.

TABLE VI
SUMMARY OF LINK UTILIZATIONS IN A FOUR-PARKING-LOT TOPOLOGY

		FPI		PI	
Flow num.	#link	Link util.	Goodput	Link util.	Goodput
20	1	0.898	7.942	0.900	7.960
	2	0.754		0.756	
	3	0.711		0.712	
	4	0.639		0.641	
40	1	0.996	11.032	0.996	11.005
	2	0.940		0.939	
	3	0.903		0.901	
	4	0.833		0.831	
60	1	0.997	11.914	0.997	11.908
	2	0.982		0.980	
	3	0.948		0.947	
	4	0.888		0.888	
80	1	0.995	12.322	0.996	12.344
	2	0.988		0.989	
	3	0.961		0.964	
	4	0.913		0.915	
100	1	0.992	12.609	0.996	12.614
	2	0.987		0.991	
	3	0.967		0.970	
	4	0.931		0.932	

The results, presented in Fig. 12 and Table VI, show two different aspects of a multi-bottleneck path. The throughput of the aggregated flow is inversely proportional to the number of congested links that the flow traverses along its path. Secondly, if we look at the single link we can notice a lower utilization at those links which are more distant from sources due to the flow throughput downgrading. When a packet is dropped, the transmission capacity that was used at each upstream link for that packet transmission has been wasted.

VI. CONCLUSIONS

A fuzzy proportional integral controller for AQM has been adopted in order to achieve both good queue regulation and high

link utilization. In this paper a genetic algorithm has been proposed and evaluated for the optimal tuning of the FPI controller parameters. The main objectives of the controller design method are fast response to high load variations and disturbance rejection in steady-state behavior. These design goals are encoded in a performance index and the genetic algorithm optimally tunes the fuzzy controller parameters, such as center points of membership functions and scaling factors. The FPI controller has been tested under different traffic conditions and compared to other AQM policies. The experimental results demonstrate that the FPI controller outperforms the other AQM policies under various operating conditions, especially for traffic that exceeds the nominal bandwidth causing severe overload on the node. The improvement in terms of response time and link utilization is due to the fact that the nonlinear fuzzy controller has a variable gain that allows the AQM to recover faster from large variation in traffic loads.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions which has helped to improve the quality of this paper.

REFERENCES

[1] B. Braden *et al.*, "Recommendations on queue management and congestion avoidance in the Internet," *IETF Request Comments*, vol. 2309, Apr. 1998.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.

[3] C. HolIot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. IEEE INFOCOM.*, Anchorage, AK, Apr. 2001, pp. 1726–1734.

[4] S. Athuraliya, V. Li, S. Low, and K. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, pp. 48–53, May 2001.

[5] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," in *Proc. ACM SIGCOMM*, San Diego Ca, Aug. 2001, pp. 123–134.

[6] S. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and scalable control," in *Proc. IEEE INFOCOM*, June 2002.

[7] S. Low, F. Paganini, and J. Doyle, "Internet congestion control," *IEEE Control Syst. Mag.*, vol. 22, pp. 28–43, Feb. 2002.

[8] C. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and design of controllers for RED routers supporting TCP flows," *IEEE Trans. Automat. Contr.*, vol. 47, pp. 945–959, Jun. 2002.

[9] R. Fengyuan, L. Chuang, Y. Xunhe, S. Xiuming, and W. Fubao, "A robust active queue management algorithm based on sliding mode variable structure control," in *Proc. IEEE INFOCOM*, New York, June 2002, pp. 13–20.

[10] G. Di Fatta, G. L. Re, and A. Urso, "A fuzzy approach for the network congestion problem," in *Proc. ICCS*, Amsterdam, The Netherlands, Apr. 2002.

[11] R. Y. R. Fengyuan and S. Xiuming, "Design of a fuzzy controller for active queue management," in *Comput. Commun.*, 2002, vol. 25, pp. 874–883.

[12] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. London, U.K.: Oxford Univ. Press, 1996.

[13] D. E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Norwell, MA: Kluwer, 2002.

[14] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, Aug. 2001.

[15] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP rena performance: A simple model and its empirical validation," *IEEE/ACM Trans. Networking*, Apr. 2000.

[16] S. McCreary and K. Claffy, "Trends in wide area IP traffic patterns: A view from ames internet exchange," in *Proc. ITC Specialist Seminar*, Monterey, CA, Sept. 2000.

[17] M. May, J. Bolot, C. Diot, and B. Lyes, "Reasons not to deploy RED," in *Proc. IWQoS*, Mar. 1999, pp. 260–262.

[18] W. Fen, D. Kandlur, D. Saha, and K. Shin, "A self-configuring RED gateway," in *Proc. INFOCOM*, W. V. Oz and M. Yannakakis, Eds., New York, Mar. 1999.

[19] W. Feng, K. Shin, D. Kandlur, and D. Saha, "The BLUE active queue management algorithms," *IEEE/ACM Trans. Networking*, vol. 10, pp. 513–528, Aug. 2002.

[20] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: stabilized RED," in *Proc. INFOCOM*, W. V. Oz and M. Yannakakis, Eds., New York, Mar. 1999.

[21] H. Wang and K. Shin, "Refined design of random early detection gateways," in *Proc. GLOBECOM*, New York, 1999, pp. 769–775.

[22] S. Cnodder, O. Elloumi, and K. Pauwels, "RED behavior with different packet sizes," in *Proc. ISCC*, 2000, pp. 793–799.

[23] C. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 1510–1519.

[24] G. Franklin, J. D. PoweIl, and A. Emami-Naemi, *Feedback Control of Dynamic Systems*, 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[25] C. Hollot, V. Misra, D. Towsley, and W. Gong, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, Mar. 1999, pp. 51–160.

[26] W. Li, "Design of a hybrid fuzzy logic proportional plus conventional integral-derivative controller," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 449–463, 1998.

[27] H. A. Malki, H. D. Li, and G. R. Chen, "New design and stability analysis of fuzzy proportional-derivative control system," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 245–254, 1994.

[28] G. Ascia, V. Catania, G. Ficili, and D. Panno, "A fuzzy buffer management scheme for ATM and IP networks," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001, pp. 1539–1547.

[29] I.-X. Xu, C.-C. Hang, and C. Liu, "Parallel structure and tuning of a fuzzy PID controller," *Automatica*, no. 36, pp. 673–684, 2000.

[30] D. Misir, H. A. Malki, and G. Chen, "Design and analysis of a fuzzy proportional-integral-derivative controller," *Fuzzy Sets Syst.*, no. 79, pp. 297–314, 1996.

[31] F. Alonge, F. D'Ippolito, F. M. Raimondi, and A. Urso, "Method for designing pi-type fuzzy controllers for induction motor drives," in *IEE Proc. Control Theory Applications*, ser. 1, 2001, pp. 61–69.

[32] C. Dualibe, P. Jespers, and M. Verleysen, "A 5.26 mflips programmable analogue fuzzy logic controller in a standard cmos 2.4 μ technology," in *Proc ISCAS*, Geneva, Switzerland, 1996, pp. 674–680.

[33] P. Bonissone, P. Khedkar, and Y. Chen, "Genetic algorithms for automated tuning of fuzzy controllers: A transportation application," in *Proc. 5th IEEE Int. Conf. Fuzzy Systems*, New Orleans, LA, 1996, pp. 674–680.

[34] C. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. Singapore: World Scientific, 2001.

[35] F. Hoffmann, "Evolutionary algorithms for fuzzy control system design," *Proc. IEEE*, vol. 89, pp. 1318–33, Sept. 2001.

[36] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[37] V. Authors. (2000) Ns-2 Network Simulator (Ver. 2). [Online]. Available: http://www.isi.edulnsnam!ns/

[38] J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 16, pp. 122–128, Jan.–Feb. 1986.

[39] L. Guo and I. Matta, "The war between mice and elephants," in *Proc. ICNP*, Riverside, CA, Nov. 2001.

[40] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems,", DEC Research Report TR-30l, Tech. Rep, 1984.

[41] M. Allman, "A web server's view of the transport layer," *ACM Comput. Commun. Rev.*, vol. 30, no. 5, Oct. 2000.

[42] V. Paxson and S. Floyd, "Wide area traffic: The failure of poisson modeling," *IEEE/ACM Trans. Networking*, vol. 3, pp. 226–244, June 1995.

[43] ——, "Difficulties in simulating the internet," *IEEE/ACM Trans. Networking*, vol. 9, pp. 392–403, Aug. 2001.

Giuseppe Di Fatta (M'03) received the M.S. degree in electronic engineering in 1995, and the Ph.D. degree in electronic, computer science, and telecommunications engineering in 2002 from the University of Palermo, Palermo, Italy.

In 1999, he was with the International Computer Science Institute, Berkeley, CA, as Research Fellow. Since 2000, he has been Lecturer at the University of Palermo, Palermo, Italy, and joined the High Performance Computing and Networking Institute of the Italian National Research Council (ICAR-CNR), Palerno, Italy. His research interests are in the area of computer networks, congestion control, network management, active networks, and soft computing.

Frank Hoffmann (M'03) received the Ph. D. degree in physics from the University of Kiel, Kiel, Germany in 1996.

From 1996 to 2000, he has been with the University of California, Berkeley, as a Visiting Researcher. He was Lecturer in computer science at the Royal Institute of Technology in Stockholm, Stockholm, Sweden from 2000 to 2002. He is currently Interim Chair for control system design at the University of Dortmund, Dortmund, Germany. He is General Chair of WSC8, as well as Founding Member of the World Federation of Soft Computing. His research interests are in the area of machine learning, fuzzy control, soft computing and robotics.

Dr. Hoffmann is Associate Editor for IEEE TRANSACTIONS ON SYSTEMS MAN, AND CYBERNETICS PART B.

Giuseppe Lo Re (M'95) received the Laurea degree in computer science at the University of Pisa, Pisa, Italy, and the Ph.D. degree in electronic, computer science, and telecommunications engineering at the University of Palermo, Palermo, Italy, in 1990 and in 1999, respectively.

In the 1991, he joined the Italian National Research Council, where he is currently a Senior Researcher. In 1994 he was a Visiting Scientist at the IBM Euoropean Networking Center, Heidelberg, Germany. Since 1999, he has been Lecturer in computer networks at the University of Palermo. His current research interests are in the area of computer communication networks and distributed systems.

Dr. Lo Re is a member of the Association for Computer Machinery.

Alfonso Urso (M'01) received the Laurea degree in electronic engineering, and the Ph.D. degree in systems engineering from the University of Palermo, Palermo, Italy, in 1992 and 1997, respectively.

In 1998, he was with the Department of Computer and Control Engineering, University of Palermo, as Research Associate. In 2000, he joined the research institute ICAR of the Italian National Research Council (CNR), where he is currently a Researcher in systems and computer engineering. Since 2001 he has been a Lecturer at the University of Palermo. His research interests are in the area of soft computing applications and network congestion control.

Dr. Urso is a member of the International Federation of Automatic Control.