PERFORMANCE OF NONLINEAR QUEUE MANAGEMENT ALGORITHMS IN BEST-EFFORT NETWORKS

Timo Lehto, Hannu Koivisto, Teemu Ekola, Mikko Laurikkala [*]

[*] *Tampere University of Technology, P.O. Box 692, FIN-33101 Tampere, Finland*

Abstract: This paper presents a system theoretic approach to, and performance analysis of improved congestion control methods i.e. the RED (Random Early Detection)-like AQM (Active Queue Management) algorithms. We revisit the principles of Fuzzy-RED and its linear counterparts and examine their advantages and possible shortcomings from a control theoretical point of view. The performance comparisons are based on extensive simulations. The major interest is on the requirement of a nonlinear control law. *Copyright ©2005 IFAC*

Keywords: Traffic control (communication), simulation, fuzzy control, communication system performance, Internet

1. INTRODUCTION

Networks become congested when the load exceeds network capacity and some of the transmitted packets need to be dropped. The basic idea behind RED-like algorithms is to drop packets already before full congestion and thus indirectly affect the TCP flow control to reduce their packet sending rate.

Packets usually arrive at the router in bursts rather than as a steady flow (Leland *et al.*, 1995) which has always made congestion control difficult. To improve and preserve performance, several Active Queue Management (AQM) algorithms (Branden *et al.*, 1998) have been proposed to be used as a complementary method to TCP congestion control. One of the most promising AQM algorithms is Random Early Detection (RED) (Floyd and Jacobson, 1993). It has been extensively studied in the past few years.

Recently several alternatives and modifications of the basic RED algorithm have been proposed and analyzed by several authors. Research issues vary from implementing RED or RED-like AQMs to different network types c.f. ECN (explicit congestion notification) marking systems (Chung and Claypool, 2003) (Athuraliya *et al.*, 2001) and Differentiated Services (Rossides *et al.*, 2002) to their performance on varying traffic conditions, implementation and tuning problems. Although some of them do not strictly use the average queue to compute congestion they have performance goals similar to that of basic RED. They will be denoted here as RED-like AQMs.

The focus of this paper is on control theoretic approach. Also this type of analysis has been performed, c.f. classical PI control design (Hollot and Misra, 2001) or even chaos theoretic approach (Wang, 2002). The process itself seems to be not only nonlinear but also time varying due to changing traffic conditions. This results in difficulties in tuning and especially in finding good "global" parameter settings for different network conditions (Feng, 1999). In general RED-like AQMs require either high robustness or adaptive features. From the control system point of view the proposed solutions and corresponding research could be categorized as:

- Event-based approaches which modify RED's nonlinear drop probability decision curve: for example gentle-RED (Chung and Claypool, 2003) or REM (Athuraliya *et al.*, 2001)
- Slow (adaptive/control) algorithms which modify RED's parameters. They normally operate periodically and use average queue length measurements. The actual decision is made with event-driven RED: for example classical PI-control (Hollot and Misra, 2001) or digital PD-control (Sun and Ko, 2003).
- Direct control algorithms which use actual queue measurements and produce probability which is directly used for drop decision. They also operate periodically. They implement various ideas from linear, nonlinear and even self-tuning control. Fuzzy techniques are popular for nonlinear treatment, for example Fuzzy-RED (Chrysostomou *et al.*, 2003).

Although the approaches above seem to be efficient, it is not easy to derive general conclusions of their overall performance. This is due to two varying features: traffic conditions and the effect of tuning parameters. In a linear case one could apply robust control theory, but in nonlinear case similar analysis would require exhausting simulations and search procedures using different traffic cases and different tuning settings.

In this paper we study the nonlinearity requirements and robustness issues. We also try to demonstrate the difficulties encountered when comparing the approaches. The focus is on direct RED-like AQMs. The methods used are: basic RED, fuzzy control and its linearized counterpart. The rest of the paper is organized as follows. Chapter 2 outlines the basics of direct RED-like algorithms. Chapters 3 and 4 present the controllers and networks used in our simulations. The simulation results are presented in chapter 5. We conclude the paper in chapter 6.

2. RED-LIKE AQM ALGORITHMS

Conventional RED algorithm (Floyd and Jacobson, 1993) operates in an event-based manner. It uses queue length (q_{len}) measurements but as filtered (q_{ave}). The drop probalility (P_d) is computed in a nonlinear manner and used for actual drop decision. The behavior of RED algorithm is controlled by four fixed parameters. The ones used in our simulations are presented in Table 1. For a more detailed discussion about RED, c.f. (Stallings, 2002).

As presented earlier, direct control algorithms typically use true non-averaged queue length measurements. They operate periodically and calculate the drop probability. The drop probability can be presented as a function of past measurements or errors:

$$P_d(t) = F(q_{len}(t), ..., Q_T) \tag{1}$$

where $q_{len}(t), ...$ are the measured queue lengths at samples $t, t-1, ...$ The target queue length Q_T (setpoint) is typically constant. The function $F(\cdot)$ is some linear or nonlinear mapping. Sometimes differences $\Delta q(t) = q_{len}(t) - q_{len}(t-1)$, control errors $e(t) = q_{len}(t) - Q_T$ or error differences $\Delta e(t) = e(t) - e(t-1)$ are used instead, resulting for example

$$P_d(t) = F_1(q_{len}(t), \Delta q(t)) \tag{2}$$

$$P_d(t) = F_1(e(t), \Delta e(t)) \tag{3}$$

With a constant setpoint, these formulations do not change the control law from (1), because for (2) one could always find $F_2(\cdot)$ as

$$P_d(t) = F_2(q_{len}(t), q_{len}(t-1), Q_T) \tag{4}$$

It is more a matter of taste and tuning habits which presentation is adopted. An example of this control surface is presented in Fig. 1. Errors and their differences are generally avoided in nonlinear control, because it is impossible to distinguish between two operation points by using the error information. In our case Q_T is generally constant and both presentations can be used. For example (2) behaves like a nonlinear PD-type controller: no integral action resulting a stationary set point error. If one wants to implement integral action, a typical solution is to use incremental approach

$$\Delta P_d(t) = F(e(t), \Delta e(t)) \tag{5}$$

This is known as incremental nonlinear PI-type control. Also this could be presented using (4).

Linear versions of above presented algorithms could also be used, resulting well-known linear PD, PI and PD-algorithms. For example using (5) will result in

$$\Delta P_d(t) = k_1 e(t) + k_2 \Delta e(t) \tag{6}$$

i.e. an incremental PI-algorithm. This approach within slow adaptation is used in (Sun and Ko, 2003), although misleadingly denoted as "PD-RED". Classical PD-control can be formulated as

$$P_d(t) = k_1 e(t) + k_2 \Delta e(t) + k_3 \tag{7}$$

$$P_d(t) = k_1 (q_{len}(t) - Q_T) + k_2 \Delta q(t) + k_4 \tag{8}$$

where k_3 and k_4 may be used to achieve desired stationary P_d. An example of typical control surface is presented in Fig. 2. It should be emphasized that all presented controls are functions of past

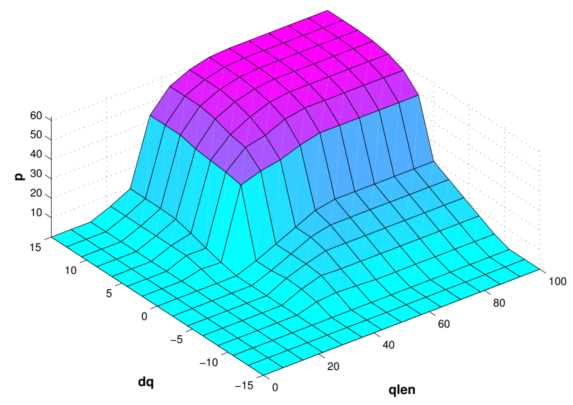

Fig. 1. An example of nonlinear control surface (Fuzzy-RED in fact). The control surface shows the relation between the inputs ($qlen$, $\triangle q$) and output variable (P_d).

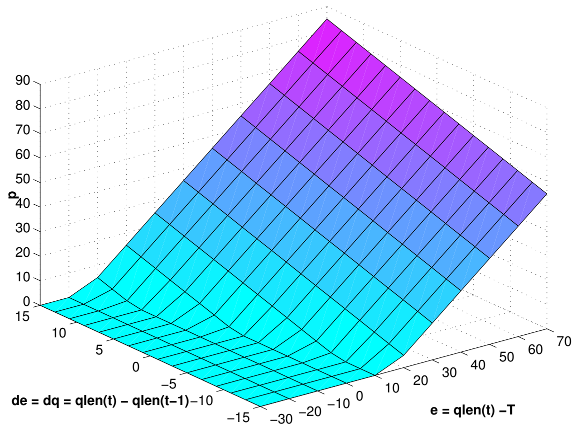

Fig. 2. The corresponding linear PD control surface. The basic relationship is linear, but due to the limiter ($Pd \geqq 0$) the actual behaviour is nonlinear.

measurements and setpoint i.e. of the form (1). The design/tuning means selection of the past history of measurements used and the mapping itself. This will define the performance of the algorithm, not whether q_{len}, Δq_{len}, e or Δe formulation is used.

Practical issues considering design and tuning are important. Some sort of convenient approaches should be used in practice. A well-known and efficient tool for nonlinear control design is fuzzy approach (Jang *et al.*, 1997). The main power lies in it's heuristic and linguistic design via rules and membership functions but also in the fact that it produces nonlinear mappings when implemented.

3. CONTROL SETUP

The main goal of this research is to study nonlinearity and robustness issues: Which sort of mapping results in robust and efficient performance? Which sort of nonlinearity is necessary?

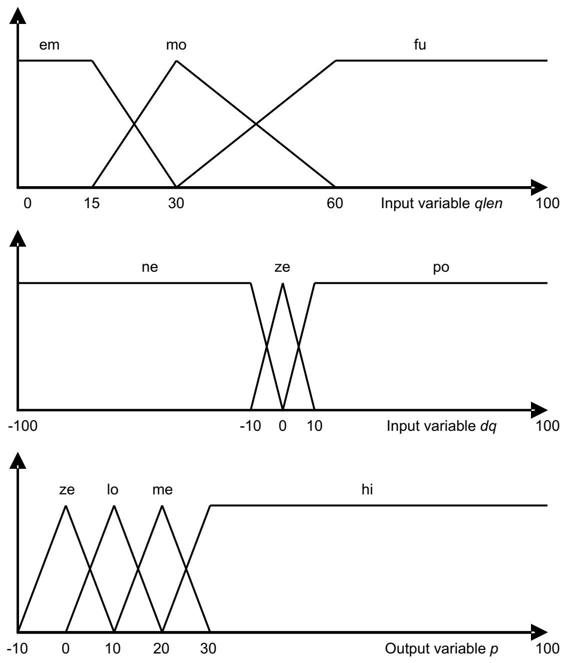

Fig. 3. Fuzzy-RED membership functions. The output variables in Mamdani-type controllers have fuzzy membership functions. The variables $qlen$, dq and p can be considered as linguistic variables and the membership functions of a variable are its possible linguistic values (e.g. *The queue length is modest*).

A fuzzy nonlinear control (2) and a linear PD-type control (8) approach are selected for a more detailed analysis. This type of fuzzy approach is used for example in Fuzzy-RED (Rossides *et al.*, 2002), (Chrysostomou *et al.*, 2003). Their application uses the formulation

$$P_d(t) = F(q_{len}(t), \Delta q(t)) \tag{9}$$

which is a suitable presentation for heuristic tuning. This fuzzy controller was implemented using Fuzzy Logic Inference Engine (FLIEC, 2004). cFLIE is adapted from the original flie of Institute of Robotics, ETH, Zurich. It implements a Mamdani-type FIS with max-type T-norm which is an adequate solution, although a Sugeno-type FIS is often favored due to its easier tuning.

Based on trial simulations (interval $\triangle t = 10ms$) we selected three membership functions for both input parameters and four for the output parameter. These membership functions are presented in Fig. 3. The fuzzy rule base obtained after trial simulations consisted of 7 rules, the corresponding control surface is presented in Fig. 1.

In the simulations the fuzzy controller calculated the drop probability only after each time interval. For a selected interval (10ms) a 10 Mbps link can transfer 0.1 Mb. If the packet size is set to 1 kB this means that only approximately 12 packets can be sent via the 10 Mbps link during the interval. This is also the limit to how much the queue length can decrease during one interval.

The tuning parameters for linear PD-type controller

$$P_d(t) = (k_1 e(t) + k_2 \Delta e(t) + k_3)/100 \qquad (10)$$

were obtained by linearizing (9) around nominal setpoint ($qlen = 30$, $dq = 0$) resulting $k_1 = 1.0$, $k_2 = 1.0$, $k_3 = 0$ and $Q_T = 30$. The corresponding decision surface is presented in Fig. 2.

The conventional RED algorithm is used for comparison. The parameter values used are presented in Table 1. All the values were selected according to the common guideline of RED parameter settings (Floyd, 2004*b*).

Table 1. RED parameters used in simulations.

Symbol	Quantity	Value
q_{min}	lower threshold for queue	15
q_{max}	upper threshold for queue	45
P_{max}	value for P_b at $q_{ave} = q_{max}$	0.1
W_q	queue weight	0.002

4. SIMULATED NETWORKS

All simulations were made using network simulator *ns2* (ns2, 2004). The network topology is presented in Fig. 4. The link between routers R0 and R1 was set to 10 Mbps and it represented a bottleneck link in the network. This link was controlled by an AQM algorithm and the buffer size was set to 100 packets. The rest of the queues were normal dropTail queues and the links other than the bottleneck link were set to 45 Mbps. All of the TCP sources implemented the congestion control measures of TCP Reno while naturally the sources using User Datagram Protocol (UDP) did not pay notice to the congestion in the network. UDP sources kept on transmitting packets at Constant Bit Rate (CBR) throughout the simulation regardless of whether the network was congested or not. Most of the sources using TCP protocol also used File Transfer Protocol (FTP) and simulate large file transfers through the Internet.

Main purpose of the simulation was to generate a set of different traffic scenarios to study the overall behaviour of the proposed control algorithms. In the simulations we varied the number of active connections as well as the round trip times (RTT) seen by the sources. A few simulations were also made where most of the TCP flows were short and small to simulate HTTP-like web traffic. Short and small flows contained only few packets and they did not last throughout the simulation. The importance of mixing different kinds of flows in simulations is discussed briefly in (Brownlee and Claffy, 2002). In most of the simulations we assumed that the time it takes for the router to forward a packet from the queue does not depend on the actual size of the packet to be sent. With this assumption, a fixed packet size (we used 1 kB) could be used in the simulations, except in scenario 7 (see below).

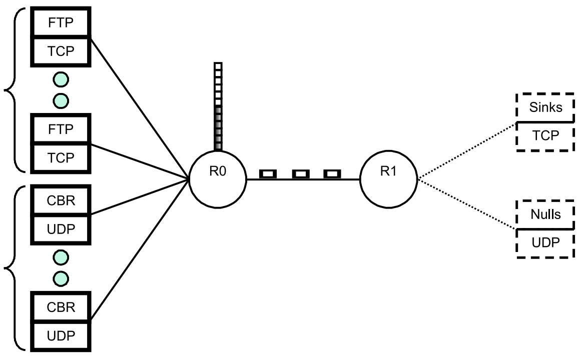

Fig. 4. The network topology. All packets that arrive at their destination traverse through the bottleneck link between the two routers R0 and R1.

Due to the space limitations the used simulation scenarios are only outlined here:

Scenario 1: A base level simulation with 30 active TCP sources (FTP) each having 50 ms RTT.

Scenario 2: A light traffic load case with only 10 active TCP sources (50 ms RTT).

Scenario 3: A heavy traffic load case with 120 active TCP sources (50 ms RTT).

Scenario 4: An increased round trip time: 15 active TCP sources, each having a 200 ms RTT.

Scenario 5: An UDP traffic case. Added three UDP sources with constant bit rate (CBR). UDP portion of total traffic is about 30%.

Scenario 6: The purpose of this scenario was to simulate a more realistic situation where most of the flows are short and the number of active connections varies during the simulation. The WebTraf module of ns2 is used for simulating HTTP-like web traffic. The distributions and parameters are based on (Chrysostomou *et al.*, 2003).

In this scenario we used 10 active FTP sources and 20 HTTP-clients. This means that the number of active TCP connections varied between 10 and 30 during the simulation.

Scenario 7: In real networks (unlike in previous simulations) packets are of various sizes. This scenario uses fixed size buffer and queue length is measured in bytes instead of packets (Floyd, 2004*a*).

Table 2. Scenarios - Key ratios

Test set	Drops	Through Mbps	Queue mean	Queue st.dev
Scenario 1				
RED	2068	10.00	33.3	8.3
Fuzzy	1975	10.00	26.2	7.6
PD (T=30)	1866	10.00	29.9	9.3
Scenario 2				
RED	639	9.98	19.8	9.0
Fuzzy	548	9.95	18.9	8.7
PD (T=30)	505	9.92	20.4	11.1
Scenario 3				
RED	6399	9.85	47.4	30.3
Fuzzy	6274	10.00	36.8	4.1
PD (T=30)	6284	10.00	42.1	4.4
Scenario 4				
RED	145	9.34	8.5	10.2
Fuzzy	88	9.96	13.9	7.9
PD (T=30)	85	9.88	13.4	10.7
Scenario 5				
RED	2034	10.00	32.6	11.6
Fuzzy	1836	10.00	26.1	7.7
PD (T=30)	1879	9.99	30.8	9.1
Scenario 6				
RED	4154	9.05	40.6	29.2
Fuzzy	1969	9.57	23.0	16.5
PD (T=30)	1524	9.79	24.1	13.3
Scenario 7				
RED	2095	5.20	17.5	8.1
Fuzzy	2034	5.19	12.9	5.0
PD (T=30)	1869	5.20	14.7	5.9

For the scenario 7 we used network data collected from a campus border gateway using NetFlow (Cisco, 2004) software. Recorded flows were classified into five classes according to their average packet size. Finally, based on the actual amount of packets in each class a number of active TCP sources were assigned to the classes accordingly.

Control algorithms were scaled to measure the queue in bytes using average packet size. In this scenario the average packet size was calculated to be 522 bytes which is quite close to the ns2 default value of 500 bytes. To make this simulation more comparable with the previous simulations we dropped the link bandwidth between the routers from 10 Mbps to 5.22 Mbps.

5. SIMULATION RESULTS

The overall comparative simulation results in Fig. 5 presents mean queue length and its standard deviation obtained in different simulation scenarios. It can be used for analyzing the control system overall performance. For example a PD-control type of behavior: stationary/steady state error is clearly present ($Q_T = 30$). The variance is a good measure for controllers capabilities.

More detailed overall comparative simulation results are presented in Table 2. They include also throughput measures and drop counts.

It is well known that RED algorithm has severe robustness problems. It is difficult to find good general settings so that RED behaves adequately. As it can be seen from Fig. 5, RED somewhat fails during scenarios 3 (heavy traffic) and 6 (HTTP-like traffic).

On the other hand, Fuzzy-RED and linear PD show clear robustness during all scenarios. A non-linear one seems to behave slightly better in most of the cases, but differences are small.

RED seem to perform well on some cases but have remarkably lower throughput on scenarios 3 (heavy traffic), 4 (longer RTT) and 6 (HTTP traffic). Fuzzy-RED performs better in these cases.

Scenarios 6 and 7 are perhaps most interesting from the practical point of view because these scenarios try to simulate more realistic traffic distribution. Within scenario 6 RED's throughput is bad and the linear PD is best. This can also be seen from the standard deviations. Scenario 7 is not so clear: RED and the others perform well, especially from the throughput point of view.

The performance of the linearized PD control is interesting. Its throughput is somewhat less than Fuzzy-RED in most cases, but clearly better in scenario 6.

A large set of different scenarios were simulated to study control system overall performance. Still, it seems that the system operates most of time near similar operation points so that a linearized control behaves almost as good as a nonlinear one. It also shows some robustness and didn't fail in any scenarios.

Does this mean that a linear treatment is enough? If we use the linearized control law with a different setpoint, for example with a setpoint change ($Q_T : 30 \longrightarrow 60$), control performance is drastically decreased. This could be seen as a tuning issue, because one could design a linear controller for that setpoint. It is nevertheless a clear indicator of some system nonlinearities. Because the setpoint is never changed in practice this has mainly academic interest and a linear controller seems to be

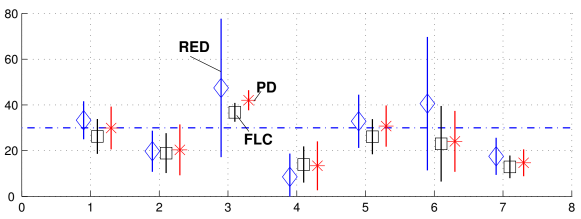

Fig. 5. Resulting average queue length and queue st.dev. during simulation scenarios 1...7 (x-axis). Control algorithms: RED $\diamond$, Fuzzy $\Box$, Linear PD $*$. The stationary steady state errors are clearly visible.

an adequate solution from the stabilization point of view.

6. CONCLUSIONS

This paper presents results when analyzing direct RED-like AQMs from the control theoretic point of view. Major goal was to study the requirements for a nonlinear control law and to analyze resulting robustness. Results outline some interesting performance issues of two selected control laws.

It is obvious and well known that the proposed methods (Fuzzy-RED and even linear PD control) have clear benefits over conventional RED. They have also been shown to be robust and effective. They are less affected by network delay than RED and in certain scenarios able to control the queue with significantly lower packet loss rates.

An interesting result is that a linear decision surface seems to perform almost as well as a nonlinear one. These results are somewhat preliminary and require more future research to ensure the analysis results.

The comparisons clarify also another feature: it is rather difficult to derive general conclusions how a RED-like AQM should be designed. Some sort of generally accepted benchmark set could be a partial solution, but even this would not remove the effect of different tuning.

REFERENCES

Athuraliya, S., S.H. Low, V.H. Li and Yin Qinghe (2001). REM: Active queue management. *IEEE Network* **15**(3), 48–53.

Branden, B., D. Clark, J. Crowcroft and B. Davie (1998). Recommendations on queue management and congestion avoidance in the internet. http://www.faqs.org/rfcs/rfc2309.html. RFC 2309.

Brownlee, N. and K. Claffy (2002). Understanding internet traffic streams: Dragonflies and tortoises. *IEEE Communications* **40**, 110–117.

Chrysostomou, C., A. Pitsillides, L. Rossides, M. Polycarpou and A. Sekercioglu (2003). Congestion control in differentiated services networks using fuzzy-RED. *Control Engineering Practice* **11**, 1153–1170.

Chung, J. and M. Claypool (2003). Analysis of active queue management. In: *Proceedings of Second IEEE International Symposium on Network Computing and Applications (NCA'03)*. pp. 359–366.

Cisco (2004). Cisco IOS netflow. http://www.cisco.com. Accessed: Sep 2004.

Feng, W. (1999). Improving Internet Congestion Control and Queue Management Algorithms. PhD thesis. The University of Michigan.

FLIEC (2004). Fuzzy logic inference engine. http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/fuzzy/systems/flie/0.html. Accessed: Sep 2004.

Floyd, S. (2004*a*). Measuring the queue in bytes or in packets. http://www-nrg.ee.lbl.gov/floyd/REDaveraging.txt. Accessed: Sep 2004.

Floyd, S. (2004*b*). RED: Discussions of setting parameters. http://www.icir.org/floyd/REDparameters.txt. Accessed: Sep 2004.

Floyd, S. and V. Jacobson (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Networking* **1**(4), 397–413.

Hollot, C.V. and V. Misra (2001). A control theoretic analysis of RED. In: *Proceedings of IEEE INFOCOM 2001*. Vol. 3. pp. 1510–1519.

Jang, J-S.R., C-T. Sun and E. Mizutani (1997). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall.

Leland, W., W. Willinger, D. Wilson and M. Taqqu (1995). On the self-similar nature of ethernet traffic. *Computer Communication Review, ACM SIGCOMM* **25**, 203–213.

ns2 (2004). The Network Simulator. http://www.isi.edu/nsnam/ns/index.html. Accessed: Sep 2004.

Rossides, L., A. Sekercioglu, A. Pitsillides, A. Vasilakos, S. Kohler and P. Tran-Gia (2002). Fuzzy RED: Congestion control for TCP/IP diff-serv. In: *Advances in Computational Intelligence and Learning: Methods and Applications* (H.J. Zimmerman, G. Tselentis, M. Van Someren and G. Dunias, Eds.). pp. 343–352. Kluwer Academic Publishers.

Stallings, W. (2002). *High-speed Networks and Internets, Performance and Quality of Service (2nd ed.)*. Prentice Hall.

Sun, J. and K-T. Ko (2003). PD-RED: To improve the performance of RED. *IEEE Communications Letters* **7**(8), 406–408.

Wang, Xiao Fan (2002). Controlling bifurcation and chaos in internet congestion control system. In: *Proceedings of the 4th World Congress on Intelligent Control and Automation*. Sanghai. pp. 573–576.