一种具有 ECN 能力的智能分组丢弃算法*

樊燕飞+, 林 闯, 任丰原, 赵达源

(清华大学 计算机科学与技术系，北京 100084)

An Intelligent Packet Dropping Algorithm with ECN Capability

FAN Yan-Fei+, LIN Chuang, REN Feng-Yuan, ZHAO Da-Yuan

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62796495, E-mail: yffan@csnet1.cs.tsinghua.edu.cn, http://qos.cs.tsinghua.edu.cn/~yfan/

Received 2004-08-10; Accepted 2005-01-17

Fan YF, Lin C, Ren FY, Zhao DY. An intelligent packet dropping algorithm with ECN capability. *Journal of Software*, 2005,16(9):1636−1646. DOI: 10.1360/jos161636

Abstract: As an effective supplement to end-to-end congestion control mechanism, active queue management aims to keep high link utility while maintaining low queuing delay. As an effective mechanism, the FIPD (fuzzy intelligent packet dropping) algorithm provides a brand-new method for active queue management. However, the FIPD algorithm also has its intrinsic defects, such as high packet loss rate. The objective of this paper is to overcome the defects of FIPD, and a new active queue management method FIPE(FIPD with ECN) is also proposed. Firstly, in this paper, the algorithm FIPD is reviewed and analyzed. Then the advantages and disadvantages are also pointed out. In order to overcome the defects such as high packet loss rate, the well-known ECN mechanism is introduced, which greatly improves the successful packet transmission. At the same time, some amendments are employed and a new active queue management algorithm FIPE is developed. Finally, the validity of the new algorithm is verified by a series of simulations on NS2 simulator.

Key words: congestion control; fuzzy logic; fuzzy judge table; active queue management

摘 要: 作为端到端拥塞控制机制的有效补充,主动队列管理旨在保证高链路利用率的同时维持较低的排队延迟.FIPD (fuzzy intelligent packet dropping)算法作为一种有效的机制,为主动队列管理提供了全新的方法,但是,FIPD 也有其本身固有的缺点,比如居高不下的分组丢失率.旨在克服 FIPD 这些固有缺点的同时,提出一种新的主动队列管理方案 FIPE(FIPD with ECN).首先对 FIPD 算法进行了总结,并对其本身的优缺点进行了分析,针对 FIPD 算法分组丢失率高居不下等缺点,引进了众所周知的 ECN 机制,起到了很大的改善作用,提高了分组的有效传送.并且,在引

* Supported by the National Natural Science Foundation of China under Grant Nos.90104002, 60273009 (国家自然科学基金); the Special Research Foundation of Ph.D. Study in High School under Grant No.20020003027 (高等学校博士学科点专项科研基金项目); the National Grand Fundamental Research 973 Program of China under Grant No.2003CB314804 (国家重点基础研究发展规划(973))

FAN Yan-Fei was born in 1980. He is a master student at Tsinghua University. His research areas are computer networks, parallel and distributed systems. **LIN Chuang** was born in 1948. He is a professor at Tsinghua University and a CCF senior member. His research areas are computer networks and QoS control. **REN Feng-Yuan** was born in 1970. He is an associate professor at Tsinghua University and a CCF senior member. His research areas are network traffic management and control. **ZHAO Da-Yuan** was born in 1977. He is a master student at Tsinghua University. His current research area is computer network security

入 ECN 的这个过程中,生成了一个新的主动队列管理算法 FIPE.最后,通过一系列在 NS2 平台上的仿真实验验证了新算法的有效性.

关键词: 拥塞控制;模糊逻辑;模糊判别表;主动队列管理
中图法分类号: TP393 **文献标识码:** A

1 Introduction

Along with the fast increase of the Internet users and various applications, the current network bandwidth usually cannot meet the transmission request. Thus network congestion frequently occurs in the present Internet. Network congestion can incur the excess packet loss and intolerable transmission delay, even network breakdown. As a result, congestion control has become one of the focuses in recent academic researches.

The end-to-end TCP traffic control mechanisms which emerged in the late of the eighties, from the quick recovery mechanism in the TCP Reno to the selective acknowledgement in the TCP SACK and etc, have become the leading technology to solve this kind of problems gradually. Many related settlements are proposed in Ref.[1] and some of them have been the basis of the congestion control in the TCP network nowadays. The basic idea in all these settlements, which is simple and comprehensible, is that the TCP sender adjusts its sending rate according to the network traffic status. The modeling and the verification through simulations and real network measurements have become more and more precise and persuasive[2,3]. Thereafter, to eliminate the negative effect caused by the multiple dropped packets in the same window, many enhanced versions of the TCP traffic control algorithm are proposed, such as Tahoe, Reno, New Reno, SACK[4] and etc. But all the aforementioned work focuses on the end system. Recent researches indicate that whatever subtle mechanisms are adopted, the effect that the end systems can put on the traffic control is restricted and extending the function of the intermediate node should be an effective way to reinforce the end-to-end traffic control system.

The queuing management algorithms on the intermediate node affect the performance of TCP up to the whole network. An inappropriate queuing management algorithm, such as DropTail, may induce the global synchronization of all the TCP connections, the long-time full state of queue and subsequently the dispensable network delay along with the low tolerance to the burst traffic[5]. Therefore, Braden *et al.* firstly brought forward the research proposal of Active Queue Management (AQM)[6] in 1998. As an enhancement to the end-to-end congestion control system, Active Queue Management mainly aims at keeping a low queuing delay while maintaining a high link utility[7]. Meanwhile, Active Queue Management includes a series of secondary objective, such as improving the equity by eliminating the prejudice against the burst traffic, avoiding the unnecessary packet dropping, preventing the global synchronization as well as enhancing the tolerance to the instantaneous congestion and etc. The RED algorithm[8] put forward by Floyd in 1993 met the technical objective of AQM and RFC2309[6] recommended it as the only candidate algorithm. Subsequently, the applications and researches around the AQM and RED were booming up gradually. Due to the consideration of performance, router manufacturers support the RED algorithm in their products, such as Cisco 7500, Juniper M40 or M80 and so forth.

As the researches go deep, however, various restrictions of RED have been realized, such as the traffic oscillation, dispensable overhead in quick forwarding and so on[9]. On the other hand, there exists another kind of operation point problem in RED algorithm. That is, the algorithm is very sensitive to the configuration parameter and network state, and an improper configuration may result in the unsteadiness of the whole network and even the traffic's breakdown. Therefore, in the subsequent long period, some modifications and amendments were put onto the RED algorithm and many improved versions of the RED algorithm emerged, such as Stabilized-RED[10], Self-configuring RED[11], Adaptive RED[12] and so on. Besides the betterments to the RED, many novel AQM

mechanisms were also put forward, such as BLUE[13], PI controller[14] and so forth. Summarizing the existing AQM algorithm, and majority of the algorithms follow the probability dropping mechanism of RED and its usefulness is indubitable. However, the process of determining whether the packet is dropped or not is essentially a decision-making process based on the particular objective. Supported by the intelligent decision-making technology, FIPD (fuzzy intelligent packet dropping) algorithm can achieve the technical objective of AQM. As a brand-new packet dropping mechanism, FIPD have become a prospective queue management algorithm.

In this paper, we first review the philosophy of FIPD algorithm and the ECN mechanism. From the review, the defect of the FIPD algorithm like high packet loss rate is revealed. In order to overcome the defect, we combine them into a new algorithm FIPE (FIPD with ECN) and further make several steps of improvement. These measures, such as introducing non-ECN queue length, tuning queue excursion, give the new algorithm a better performance. After these amendments, we verify the new algorithm from several aspects, such as the packet loss rate, queue evolvement, equity and robustness. The simulation results show out the advantages of the new algorithm.

2 Related Work

2.1 FIPD algorithm

The FIPD algorithm[15] applies the intelligent computing technology of Fuzzy Logic to the decision-making process of packet dropping. It brings forward a brand-new idea in the queue management and has achieved a fairly good result in the network simulations.

The FIPD algorithm adopts the queue length and its change as the indicators of network traffic state. It maps the two indicators from their basic universe of discourse to the linguistic variables respectively, which also have their own universe of discourse. In the mapping process, the designer needs to determine the quantifying factor of each indicator. After the quantifying and mapping, the fuzzy set and the relevant subject function of each linguistic variable are also required. For the sake of computing, the designer may build a fuzzy assignment Table for each linguistic variable. The FIPD algorithm also defines a new observing variable —— congestion indicator to characterize the network traffic state. The congestion indicator is also required for quantifying, mapping, determining the fuzzy set and subject function of the relevant linguistic variable along with the fuzzy assignment Table. After all of the aforementioned operations are done, the designer is required to determine a fuzzy reasoning rule Table, which is built based on the human's experience and intelligence. Uniting the above three fuzzy assignment Tables and depending upon the fuzzy reasoning rules, we can do the fuzzy reasoning computation and get a final fuzzy judge Table, which is the guide line of determining whether a packet is dropped or not. The fuzzy judge Table of FIPD employed in Ref.[15] is shown in Table 1.

Table 1 Fuzzy judge table

QC\QL	−4	−3	−2	−1	0	1	2	3	4	5	6	7	8
−4	−1	−1	−0.75	−0.75	−0.5	−0.5	−0.5	−0.5	0	0.25	0.38	0.5	0.5
−3	−1	−0.75	−0.5	−0.5	−0.25	−0.25	0	0	0.13	0.25	0.38	0.5	0.5
−2	−1	−0.75	−0.38	−0.38	0	−0.13	0	0	0.25	0.38	0.5	0.63	0.63
−1	−0.88	−0.63	−0.25	−0.25	0	0	0	0	0.25	0.5	0.63	0.75	0.75
0	−0.75	−0.5	0	0	0	0	0	0	0.25	0.5	0.63	0.75	0.75
1	−0.75	−0.5	0	0	0	0	0.13	0.5	0.38	0.63	0.75	0.75	0.75
2	−0.63	−0.38	0	0	0	0.13	0.38	0.38	0.5	0.75	0.75	0.88	0.88
3	−0.5	−0.25	0	0	0.25	0.25	0.5	0.5	0.63	0.75	0.75	0.88	1
4	−0.5	−0.25	0.25	0.25	0.5	0.5	0.75	0.75	0.75	0.75	0.75	1	1

All of the above steps are performed in off-line state, hence we can ignore the computing overhead due to the fact that only the fuzzy judge Table is used in the algorithm. The principle of the algorithm is that at every sampling time, the algorithm calculates two parameters —— the queue length QL and the change of queue length QC. After they are quantified with the respective quantifying factor, we can obtain a congestion indicator from the fuzzy judge table by using the two variables as indexes. Then by comparing the congestion indicator with the threshold, we can determine whether the packet is dropped or not. The whole process is simple and the computing overhead is very small while the simulation result is fairly good. The core routine of the FIPD algorithm is shown in Fig.1.

```
At every sampling time:
    QL = current_queue_length – expected_queue_length;
    QC = current_queue_length – old_queue_length;
    old_queue_length = current_queue_length;
    index_ql = round(index_ql*kl);
    index_qc = round(index_qc*kc);
    congestion_index = lookup(FuzzyJudgeTable, index_ql, index_qc);
Where
    The kl and kc are the two quantifying factors respectively;
    The lookup is the table lookup operation
At every packet arriving:
    If (congestion_index > congestion_threshold)
      drop the packet;
```

Fig.1 The core routine of FIPD algorithm

The FIPD algorithm, however, has its own defects. Due to the dependence on packet dropping in notifying the network congestion, it has a very high packet loss rate, which is not according to the original objective of AQM[7]. It is important that quite a few packets are dropped unnecessarily (which is used to notify the congestion in advance but the congestion is not certain to happen). So, we just think whether or not the notification is fulfilled via other way and the unnecessary packet dropping should be avoided. The ECN mechanism meets our requirements exactly.

2.2 ECN mechanism

The ECN（Explicit Congestion Notification）mechanism itself is not a novel idea and it is not proposed for IP network. Originally, Ramakrishnan and R. Jain brought forward a Binary Feedback Mechanism (DEC bit) for DNA (Digital's (proprietary) Network Architecture)[16]. Afterward, the Forward Explicit Congestion Notification mechanism (FECN) is defined for Frame Relay and ATM[17]. All of them have been used as reference in designing the ECN mechanism in IP network[18].

The ECN mechanism in IP network aims at providing an alternative manner for TCP to detect the network congestion. That is to say, the TCP sender can apperceive the network congestion, not relying on the packet dropping. The IETF has promoted the ECN on IP network as a recommended standard[19]. It is expected that the network devices with ECN support will come forth in recent future. Actually, the ECN mechanism is supported in Linux Kernel 2.4 or higher. Moreover, ECN also needs the support of Router, and Cisco has made it into their product IOS Release 12(8)T.

As mentioned early, the ECN mechanism requires the support of both the end systems and the core routers. The end systems obtain the congestion information via the ECN mechanism and then adjust their sending rate according to the end-to-end congestion control mechanism. The routers take the duty of monitoring the network state and marking the packet for notifying the end system when the congestion comes forth. It is obvious that the ECN mechanism relies on the capability of the router detects the potential congestion. Although there is no AQM mechanism designated in the last ECN criterion[20], the RED algorithm is often used in practice.

When a TCP connection begins to establish, the end system and the router inquire each other whether the ECN mechanism is supported. If they both support, the TCP sender will set the ECT (ECN-Capable Transport) bit in the outgoing IP packet. Once the router captures the network congestion, it will set CE (Congestion Experienced) bit in packet header before forwarding the IP packet. Thus, when the TCP receiver takes in the marked packet, it knows that the network is in congestion. Then the receiver will notify the sender of the congestion by setting the ECE (ECN-Echo) bit in the ACK packet header before forwarding it. When the sender receives the ACK packet, it will start the congestion control algorithm to decrease the sending rate. In this way, we can achieve the objective of congestion control. In addition, the sender will set the CWR (Congestion Window Reduced) bit in the next data packet. Because the ACK packet may be discarded in transmission, the congestion notification mechanism may be of no effect. In order to prevent the situation, the receiver will set the ECE bit in the subsequent ACK packet until receiving the data packet with CWR bit set which indicates that the sender has reduced the sending window.

The researches in Ref.[18] show that when the ECN mechanism is employed, the failure overhead of TCP transmission decreases greatly and the traffic block decreases too. But only the ECN mechanism itself couldn't improve the throughput of TCP effectively. The ECN mechanism should be combined with AQM to fully exert its effectiveness.

2.3 The combination of FIPD algorithm and ECN mechanism

In the intelligent packet dropping mechanism proposed in Ref.[15], it is by the simple packet dropping that the routers notify the traffic sources of the network congestion. The mechanism is a brand-new AQM technique. But it has many restrictions due to the simple packet dropping. Firstly, although the overhead of packet dropping in router is quite small, it is a waste of network resource. When a packet is dropped at a router, the network resources that are occupied before, such as network bandwidth and memory, will go to waste. So we should reduce the unnecessary packet dropping. Secondly, the packet dropping is not the best way to notify the end system of the congestion. There are two passages for the packet dropping mechanism to do the notification, Fast Retransmission and Overtime Retransmission. In the two passages, neither of them can be as quick and direct as the ECN manner.

The ECN mechanism is simple. It doesn't need the state information of each flow and therefore is well scalable. After the ECN mechanism is introduced, we can reduce the dispensable packet dropping. Furthermore, the ECN mechanism can notify the traffic source of the congestion in network as soon as it happens. For FIPD algorithm, it is quite necessary to introduce the ECN mechanism. As it is by the packet dropping that the FIPD algorithm itself can control the packet queue length, when the congestion appears in network, it may lead to the high packet loss rate. For the FIPD algorithm, the feature is a very adverse factor.

We combine the FIPD algorithm and the ECN mechanism together and build a new algorithm FIPE. The FIPE algorithm has many favorable characteristics. At first, it gets over the most severe defect and avoids the unnecessary packet dropping. Secondly, it inherits the almost overall virtues of FIPD, which will be seen in later simulations. The shortage is that, the queue length graph disperses a little more than that of FIPD, which is caused by the ECN mechanism. When the ECN mechanism is added to RED algorithm, the queue length graph disperses too.

3 FIPE Algorithm

Actually, it is a difficult process to add the ECN mechanism to FIPD algorithm and build the new algorithm FIPE. If the ECN mechanism is just simply introduced, the graph of queue length will disperse greatly. As is shown in Fig.2. Here is the network simulation configuration: the classical Dumbbell topology; 100 ftp traffic source; the bottleneck link bandwidth 10Mb/s, delay 20ms; the other links bandwidth 2Mb/s, delay 10ms; the key router's queue length is 200 packets and the queue management algorithm is FIPE. Unless otherwise specified, the later

simulations adopt the previous simulation configuration.

It is known that in the original FIPD algorithm, we use the two parameters, queue length QL and its change QC, to indicate the network congestion. The FIPD algorithm calculates the two parameters every sampling period and quantifies them as indexes of the fuzzy judge table. Then we can get a congestion indicator. Comparing the indicator with the threshold, we can determine whether the packet is dropped or not. Now, we introduce the ECN mechanism into the FIPD algorithm. That is to determine whether the packet is marked (set the ECN bit, not dropped) or not. The method can also achieve the objective of notifying the traffic source of the network congestion. Furthermore, it can reduce the unnecessary packet loss.

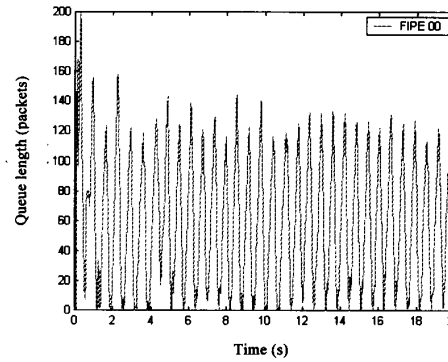

Fig.2 ECN added to FIPD directly

However this causes a new problem. As you can suppose, the marked packet is also left in queue. This will make the calculation of queue length and its change deviate from the original intention of the algorithm. The FIPE algorithm captures the congestion in the network at a sampling time, and then it will mark all the packets arriving in the next sampling period. Due to the marking but not dropping operation, the packets are left in the queue. The queue length will go bigger than that of FIPD and the change of queue length too. This will make the probability of the congestion detected at the next sampling time increase greatly, which will result in that more packets are marked. Thus, the global synchronization of all traffic in the network is brought on. At the meantime, the packet queue in the router oscillates severely and the graph of queue length disperses seriously. It will induce the indeterminacy and oscillation of the delay, which contravene the original intention of AQM (low delay and high throughput). The simulation result shown in Fig.2 verifies the analysis above.

3.1 Introducing the non-ECN queue length

In order to solve the problem, we introduce a new concept to FIPE algorithm——non-ECN queue length. In the router, we have three choices when a new packet arrives. The first adds it to the queue directly; the second drops directly; and the third sets the ECN bit before adding to queue. Although the packet adopting either of the two queuing manners is entering the same queue, we separate them logically. That is, the number of the packets not marked in the queue is called non-ECN queue length. Correspondingly, the number of the packets marked in the queue is called ECN queue length. In the quondam FIPD algorithm, there is no marked packet and naturally there is no conceptive differentiation between the non-ECN queue and the ECN queue. The non-ECN queue length is calculated in the way shown in Fig.3.

Now, we introduce the ECN mechanism into the FIPD algorithm and substitute the non-ECN queue length for the original queue length. Thus we get the FIPE algorithm. The corresponding change in the routine is shown in Fig. 4. Via the simulation experiment, we get the evolvement of queue length shown in Fig.5. After the improvement, as you can image, the calculation of the congestion indicator may differs little from the FIPD algorithm. A little difference is that the ECN marked packets will occupy some bandwidth, which equals to tuning the available bandwidth a little lower. But the occupied bandwidth is not wasted. It is used to transmit the marked packet, thus the throughput of the packets is enhanced.

```
In the enque function:
    If (the packet is not marked) {
        non_ecn_queue_length++;
    } else {
        ecn_queue_length++;
    }
In the deque function:
    If (the packet is not marked) {
        non_ecn_queue_length--;
    } else {
        ecn_queue_length--;
    }
```

Fig.3 Introduction of the non_ecn_queue

```
At every sampling time:
    QL = non_ecn_queue_length
- expected_queue_length;
    QC = current_queue_length
- old_queue_length;
    old_queue_length = current_queue_length

Where
    The expected_queue_length is a constant that is
initialized at the start
    The current_queue_length is the sum of the
non_ecn_queue_length and ecn_queue_length
```

Fig.4 Non_ecn_queue introduced algorithm

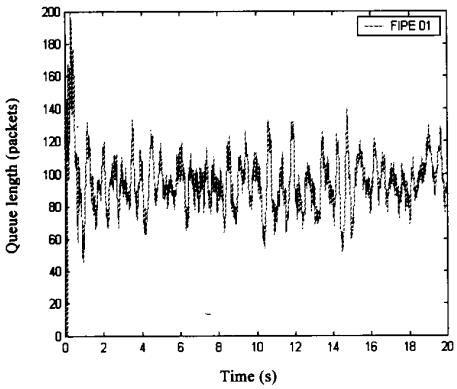

Fig.5 Non-ECN queue introduced FIPE

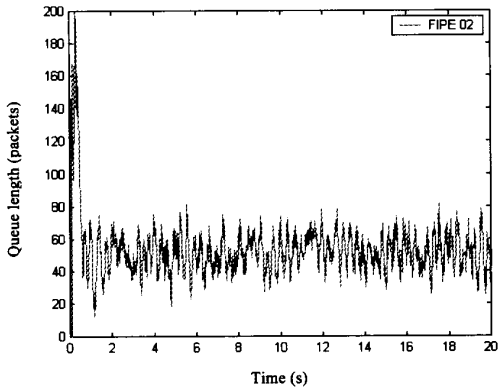

Fig.6 Excursion reserved FIPE

3.2 Tuning the queue excursion

Another noticeable fact is that the marked packet is left in the queue and the queue length is augmented. The only difference is that the marked packet is not taken into account when computing the congestion indicator. Accordingly, the real queue is certainly longer than that of the original algorithm. Fortunately, the increase is almost a whole shift of the queue length curve, which provides a favorable convenience for us to control the queue length. For this problem, we have two methods to adjust the algorithm.

The first method is very simple. That is to tune the threshold of the congestion indicator. But the method has some restrictions. First, the range for tuning is limited. It will be incapable when the excursion exceeds the tuning range. Second, the method may induce the bad performance of the algorithm, and the situation is not easy to find.

The second method is a little complex, but it gets over the limitation of the first method. Relatively, it is more stable and reliable. The method is to reserve some excursion for the shift of the queue. We can estimate the excursion beforehand by the simulation, which is a useful method if some error is saved. The simulation result is shown in Fig. 6. We can also embed the estimating routine in the algorithm to acclimatize to the dynamical network state. But if we do so, more detail work needs to be accomplished. Because the question is not the emphasis of this paper, we jump it over due to the limited words. The pseudo-code of the two methods is shown in Fig.7.

```
The first method:
    congestion_threshold += regulating_value;
The second method (excursion reservation):
    Index_ql += excursion_reservation_value;
```

Fig.7 Excursion compensation

3.3 Reducing the fuzzy judge table

It is supposed that the fuzzy judge Table is relatively bigger, but is the supposition right or not? If it is the case, can we achieve the same effect when substituting a smaller Table for the current fuzzy judge table? All these will lie on the result of the experiments.

At first, we trace the whole fuzzy judge table for the utility statistic of every cell, as shown in Table 2. The statistic result is fairly gratifying. Our supposition is completely right. The whole fuzzy judge table is a 9×13 matrix. The statistic result indicates that the utility of the Table is almost the focus in a 5×5 region (which is marked with gray background in the Table). Only a few used cells lie outside of the area and they are hardly be used. So, we have the full reasons to reduce our fuzzy judge table to a 5×5 matrix.

Table 2 The statistic result of the fuzzy judge table

0	0	0	0	1	4	4	2	0	0	0	0	0
0	0	0	1	6	52	65	13	1	0	0	0	0
0	0	0	1	14	152	276	102	6	0	0	0	0
0	0	0	0	9	115	234	96	2	0	0	0	0
0	0	2	0	5	232	636	298	15	0	0	0	0
0	0	0	0	1	102	422	244	28	0	0	0	0
0	0	0	0	0	1	3	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1

Following the process depicted in Ref.[15], we can finally get a 5×5 fuzzy judge table, as shown in Table 3. When we employ it into the algorithm, the simulation result is fairly satisfying. The new 5×5 fuzzy judge table can suit for the work of the original 9×13 table and the performance is as good as it is, as shown in Fig.8. Certainly, the big table has its own advantage. It is more tolerant than the small table. So we don't expect to see that the performance of the algorithm is damaged for the sake of memory capacity. In practice, we could make a tradeoff among all the performance indicators on the basis of the practical requirements.

Table 3 The reduced table

QC\Q	−2	−1	0	1	2
−2	−1.00	−0.75	−0.50	−0.25	0.00
−1	−0.75	−0.50	−0.25	0.00	0.25
0	−0.50	−0.25	0.00	0.25	0.50
1	−0.25	0.00	0.25	0.50	0.75
1	0.00	0.25	0.50	0.75	1.00

4 Simulation Verification

In this section, we will show the advantages against the well-known RED algorithm by simulation. We also adopt the NS2 (Network Simulator version 2) as the simulation tool. Meanwhile, we will reveal some defects of the new algorithm.

4.1 Comparing the queue evolvement

The main purpose of AQM is to keep the high link utility while maintaining the low delay. The FIPE algorithm

achieves the objective by and large. First, as shown in the Fig.7, the queue is hardly empty in the whole simulation, which is the guarantee of the high link utility. Second, the queue length is controlled into a small range, which is the key factor to attain the low queuing delay. The effect owes to the management to the queue range. If the queue range is quite big, it is impossible for us to control the queue in a low range and hardly empty.

In the same network configuration, we review the performance of the RED algorithm (the ECN mechanism is launched too). All the configuration parameters are the default values in NS2. The queue evolvement of it is shown in Fig.9. The RED algorithm can by and large ensure a high throughput while the queue range is quite big. In controlling the queue range, the RED algorithm behaves unfavorable. Hence, the RED algorithm can hardly keep high throughput while maintaining low queuing delay.

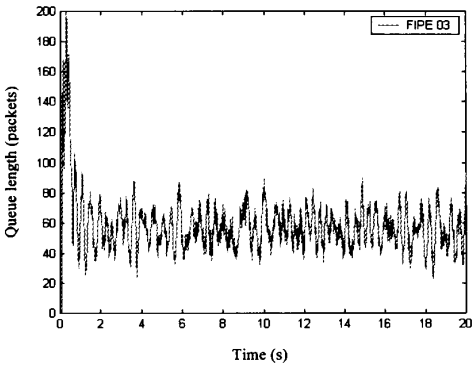

Fig.8 Fuzzy judge table with reduced FIPE

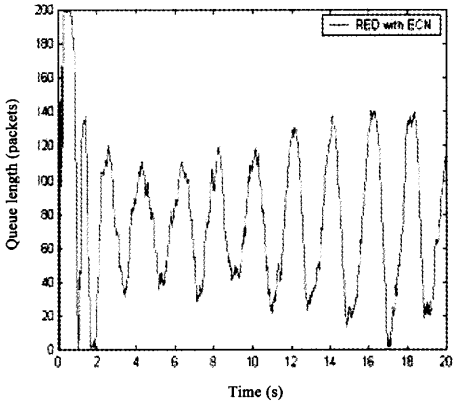

Fig.9 The queue evolvement of RED algorithm

4.2 Comparing the loss rate, equity and converging rate

We can compare the packet loss rate of the two algorithms. After the statistic of various indicators in the simulation, we obtain a lot of convictive data. In the 20 seconds' simulation, the statistics of the FIPE and RED algorithm is shown in Table 4. From the table, we could expressly make out the FIPE algorithm's advantages in reducing the unnecessary packet loss. In the simulation, the loss rate of the RED algorithm is above 20 times bigger than that of FIPE. Actually, the packet loss of FIPE is almost at the beginning of the simulation. At that time, the network traffic is experiencing a great burst (100 ftp traffic start up at the same time). In the common network state, few packets are dropped. Although the RED algorithm drops a lot of packets at the beginning, the great mass of the dropped packets occur in the common network state. Viewing from that point, the superiority of FIPE goes without saying. If the queue capacity is reduced to 100 packets, as you can image, the FIPE algorithm can work commendably while the RED algorithm will be forced to drop more packets.

Table 4 Comparing the loss rate and equity of FIPE with that of RED

	Throughput	Lost packets	Loss rate (%)	Equity index
FIPE	24026	117	0.485	0.965
RED	23985	2846	10.6	0.9267

In the converging rate of the queue, FIPE is also provided with good performance. From the queue evolvement graph, we could conclude that the queue convergence of FIPE is very fast. The longest converging time is only about 0.5 second, which is less than 1 second. While the converging rate of the RED algorithm is obviously slower and the

converging process often needs many seconds to be accomplished.

4.3 The robustness of the algorithm

Now let's review the performance of FIPE interfered with the HTTP and CBR traffic. In the first simulation below, we add 100 HTTP traffic to the 100 FTP traffic. The queue evolvements of the RED and FIPE are illustrated in the same figure for the sake of comparison, as shown in Fig.10. In current network research, the HTTP traffic is also called small mouse traffic and the FTP traffic is called elephant traffic. The effect added by the HTTP traffic is very small, so the queue evolvements of RED and FIPE have only little change. In the same figure, we could find the obvious advantage of FIPE against RED. In the second simulation, we substitute 5 CBR traffic for the 100 HTTP traffic. The CBR traffic is a non-responsive traffic, which can provide a big and smooth network load. After the 5 CBR traffic is added, the performance of RED and FIPE algorithms deteriorate, in some degree, as shown in Fig.11. The queue oscillation of the RED algorithm goes more violent. The queue evolvement of FIPE not only oscillates greater but also shifts a great distance. Even though, the performance of FIPE is still much better than that of RED.

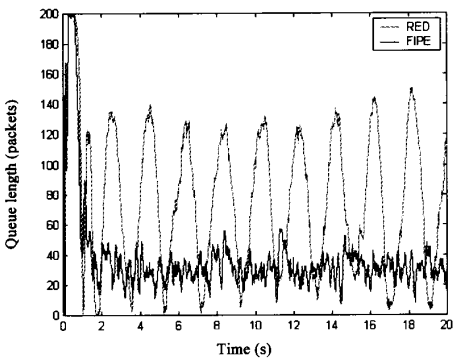

Fig.10 FIPE and RED (FTP+HTTP)

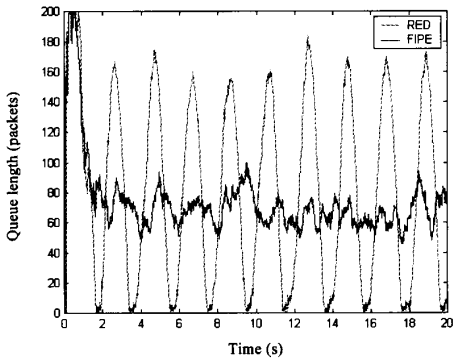

Fig.11 FIPE and RED (FTP+CBR)

5 Conclusion and Future Work

The paper firstly reviews the FIPD algorithm proposed in Ref.[15]. And then it introduces the ECN mechanism into the algorithm and builds a new active queue management algorithm FIPE. The new algorithm is mainly proposed to overcome the high packet loss rate in FIPD. Meanwhile, we also expect that the new algorithm can inherit the good characters from the FIPD. Due to the introduction of the ECN mechanism, the performance of the new algorithm FIPE deteriorates in some degree. In resolving all these questions, we bring forward some extraordinary concepts as well as the corresponding actions. Finally, we verify the new algorithm through a series of simulations on NS2 platform. The simulation results show that the new algorithm has a lot of better performance than the classical RED algorithm.

Certainly, the FIPE algorithm has its own defects. Especially in the light network load, the performance of FIPE will degrade. In addition, the excursion reservation mechanism is a simple and inaccurate implementation. The future work is to amend the algorithm to overcome these defects.

References:

[1] Jacobson V, Karels MJ. Congestion avoidance and control. In: Proc. of the SIGCOMM'88. California, 1988. 314–329.

[2] Mathis M, Semske J, Mahdavi J, Ott T. The macroscopic behavior of the TCP congestion avoidance algorithm. Computer Communication Review, 1997,27(3).

[3] Padhye J, Firoiu V, Towsley D, Kurose J. Modeling TCP Throughput: A simple model and its empirical validation. In: Proc. of the SIGCOMM'88. California, 1988. 303–314.

[4] Stevens W. TCP slow start, congestion avoidance, fast retransmit, and fast recovery. RFC 2001, 1997. http://rfc.net/rfc2001.html

[5] Mankin A, Ramakrishnan K. Gateway congestion control survey. RFC1254, 1991. http://rfc.net/rfc1254.html

[6] Braden B, Clark D, Crowcroft J, Davie B, Deering S, Estrin D, Floyd S, Jacobson V, Minshall G, Patridge C, Peterson L, Ramakrishnan K, Shenker S, Wroclawski J, Zhang L. Recommendations on queue management and congestion avoidance in the Internet. RFC2309, 1998. http://rfc.net/rfc2309.html

[7] Floyd S. Active queue management, ECN, and Beyond. Juniper brown bag lunch, 2001.

[8] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. IEEE/ACM Trans. on Networking, 1993,1(4): 397–413.

[9] Firoiu V, Borden M. A study of active queue management for congestion control. In: Proc. of the INFOCOM 2000. New York: IEEE Press, 2000. 1435–1444.

[10] Teunis J, Ott, TV, Lakshman, Wong LH. SRED: Stabilized RED. In: Proc. of the IEEE INFOCOM'99. New York: IEEE Press, 1999. 1346–1355.

[11] Feng W, Kandlur D, Saha D, Shin K. A self-configuring RED gateway. In: Proc. of the IEEE INFOCOM'99. New York: IEEE Press, 1999. 1320–1328.

[12] Floyd S, Gummadi R, Shenker S. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management [EB/OL]. http://www.aciri.org/floyd/papers/adaptiveRed.ps

[13] Feng W, Kandlur D, Saha D, Shin K. Blue: A new class of active queue management algorithm. University of Michigan Technical Reports, 1999. CSE-TR-387-99.

[14] Hollot C, Misra V, Towsley D, Gong W B. On designing improved controllers for AQM routers supporting TCP flows. In: Proc. of the IEEE INFOCOM 2001. New York: IEEE Press, 2001. 1726–1734.

[15] Fan YF, Ren FY, Lin C. Design an active queue management algorithm based on fuzzy logic decision. In: Proc. of the IEEE ICCT 2003. New York: IEEE Press, 2003. 286–289.

[16] Ramakrishnan K, Jain R. A binary feedback mechanism for congestion avoidance in computer networks. ACM Trans. on Computer Systems, 1990,8(2):158–181.

[17] Stallings W. ISDN and Broadband ISDN, with Frame Relay and ATM 4th ed., New Jersey: Prentice Hall, 1998.

[18] Pentikousis K, Badr H. An evaluation of TCP with explicit congestion notification. Annals of Telecommunications, 2003.

[19] Bradner S. The Internet Standards Process– Revision 3. RFC 2026, 1996. http://rfc.net/rfc2026.html

[20] Ramakrishnan KK, Floyd S, Black S. The addition of explicit congestion notification (ECN) to IP. RFC 3168, 2001. http://rfc.net/rfc3168.html