VISTA: A New Fair Queuing Algorithm for Packet Switches

XIA Yu
School of Info. Science & Tech.
Southwest Jiaotong University
Chengdu 610031, China
rainsia@gmail.com

GUO Zirong
School of Info. Science & Tech.
Southwest Jiaotong University
Chengdu 610031, China
zirongguo@163.com

GAO Zhijiang
School of Info. Science & Tech.
Southwest Jiaotong University
Chengdu 610031, China
Santi1009@gmail.com

***Abstract*—Fairness and QoS have been main concerns in designing modern routers or switches in multimedia environment. This paper proposes a new packet-based fair queuing algorithm for packet switches called VISTA (Virtual Clock with Virtual Start Time Alignment) to improve fairness in existing VC (Virtual Clock) algorithm. A novel concept of virtual start time alignment is introduced to improve fairness of VC algorithm and to reduce the computing complexity to constant time with a hardware-based Earliest Packet Selector (EPS).**

Keywords- Packet-based Fair Queuin; QoS guarantee; VISTA algorithm; Virtual Clock; WF2Q+

I. INTRODUCTION

Undoubtedly, different network applications need different bandwidths and transmission delays. Currently, more and more real-time and interactive applications are emerging, for example, VoD (video on demand), video conference and IP telephone. These new applications need more bandwidths and lower transmission delays and jitters than traditional text-based applications such as web surfing and e-mail. However, traditional Internet switches/routers cannot ensure different QoS (Quality of Service) for different flows (or sessions). Figure 1 illustrates three flows with different throughput requirements are competing the capacity of an output port without fairness consideration; as a result, they are statistically sharing the output capacity without any priority.

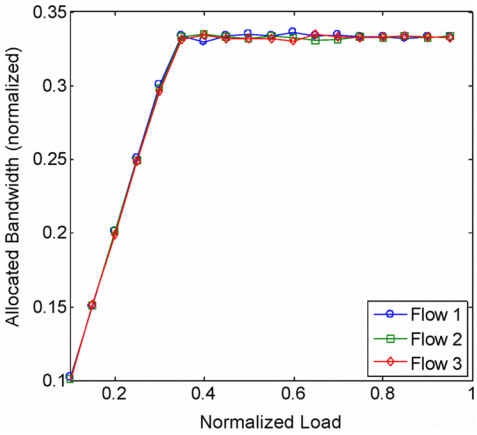

Figure 1. The flows get the same bandwidth allocation

Many improved algorithms consider fairness, but some of them only involve the fairness among different input ports. The fairness of the flows within the same port or among different ports with different QoS requirements, such as throughput, transmission delays, and jitters, is still not guaranteed. Some algorithms, such as VC (Virtual Clock), WFQ (Weighted Fair Queuing), etc. allocate the output link capacity for different flows in proportion to their bandwidth requirements. However, they are either too complicated to implement, or cannot provide exact fairness to each flow in some special conditions [1].

This paper proposes a new fair queuing algorithm called VISTA (Virtual Clock with Virtual Start Time Alignment) that is capable of providing the known best fairness as by WF^2Q [2] or WF^2Q+ [3]. In order to decrease the time complexity of the new algorithm, a hardware-based Earliest Packet Selector (EPS) is introduced, which allows the algorithm can be completed in constant time, i.e., independent of the number of flows.

The rest of this paper is organized as follows: Section 2 is a brief analysis to relevant work. Section 3 describes our new fair queuing algorithm. Section 4 further demonstrates the merits of VISTA with a hardware-based EPS design and Section 5 validates the new algorithm via simulation. Finally, Section 6 concludes the work with a forward view on future work.

II. RELATED WORK

The Generalized Processor Sharing (GPS) [4], which is based on the fluid flow model, has been thought as being absolutely fair. However, the GPS server assumes that flows can be divided into infinitely fine units; while in real switches, flows are served in terms of non-dividable packets (or cells). For this reason, numerous packet-based fair queuing (PFQ) algorithms have been developed to approximate GPS. These algorithms can be divided into two categories.

The algorithms in the first category are named timestamp-based algorithms. For each of these algorithms, the scheduler maintains three variables: system virtual time $V(t)$, virtual start time $S_{i,k}$ for the k-th packet from flow i, and corresponding virtual finish time $F_{i,k}$. All these algorithms use a similar scheduler to select the packet with the smallest virtual finish time to be output next but differ from each other in calculation of system virtual time, virtual start time and virtual finish time. Another important difference is whether they test the eligibility

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7659
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7659

of packets [2]. Eligible packets are those whose virtual start times are no more than current system virtual time.

The algorithms in the second category are called frame-based algorithms. Within a frame, a frame-based algorithm provides different flows with different amount of services according to their reservation. The frame-based algorithms are usually less complex than timestamp-based algorithms, but they cannot provide exact fairness as timestamp-based algorithms do.

Table 1 shows the latency [5] and WFI (Worst-case Fairness Index) [2] comparison between different PFQ algorithms.

TABLE 1. COMPARISON OF PFQ ALGORITHMS

Algorithm	Latency	WFI	Complexity	Algorithm	Latency	WFI	Complexity
GPS [4]	0	0	—	BSFQ [6]	$\approx L_i/r_i+L_{max}/r$	$O(N)$	$O(1)^*$
WFQ [4]	$L_i/r_i+L_{max}/r$	$O(N)$	$O(N)$	LPVC [7]	$\approx L_i/r_i+L_{max}/r$	$O(L_i/r_i)$	$O(\log\log N)^*$
VC [8]	$L_i/r_i+L_{max}/r$	∞	$O(\log N)$	DRR [9]	$(3F-2\phi_i)/r$	$O(N)$	$O(1)$
SCFQ [10]	$L_i/r_i+(L_{max}/r)(N-1)$	—	$O(\log N)$	VD [11]	$L_i/r_i+2L_{max}/r_{min}$ $+(N-1)L_{max}/r+L_{max}/r_i$	$O(N)$	$O(1)$
WF2Q [2]	$L_i/r_i+L_{max}/r$	$O(L_i/r_i)$	$O(N)$	SmRR [12]	$2L_{max}/r_i$ $+2(L_{max}/r)(N-1)$	$O(N)$	$O(1)$
WF2Q+ [3]	$L_i/r_i+L_{max}/r$	$O(L_i/r_i)$	$O(\log N)$	Aliquem [13]	$>>L_i/r_i+L_{max}/r$	$O(N)$	$O(1)$
MSPFQ [14]	$L_i/r_i+L_{max}/r$	$O(L_i/r_i)$	$O(\log N)^*$	StRR [15]	$12L_{max}/r_i$	$O(N)$	quasi-$O(1)$

*Additional conditions are needed to achieve such complexities.

The first nine algorithms are timestamp-based; while the last five are frame-based.

L_i is the maximum length of flow i packets, r_i is the reserved rate of flow i, L_{max} is the maximum length over all packets, F is the frame size of frame-based algorithms and ϕ_i is the reserved weight of flow i.

It has been shown that to keep the fairness as WF^2Q does, the complexity is lower bounded by $O(\log N)$ [16], there is no other way to reduce the complexity in traditional method, except by hardware based parallelism. So some hardware based methods are proposed such as sequencer [17] and hardware comparison tree [18]. The former combines all packets into a single queue which increases comparison; furthermore, the parallel movement of data is too power-consuming. The latter is hard to implement, and still have $O(\log N)$ complexity.

Among all these algorithms, VC is the simplest, but it might be inappropriate in following conditions:

If a flow has sent more packets than its reservation even when no packets arrived in other flows, VC will punish this flow. Figure 2 illustrates this situation: there are two flows 1 and 2 sharing an output link. The link capacity is normalized as 1 P/S (packet/timeslot). Let $r_1=r_2=0.5$ P/S and all packets from both flows are the same size. Initially, at time 0, the virtual finish times for these two flows are the same (i.e. $F_{1,0}=F_{2,0}=0$).

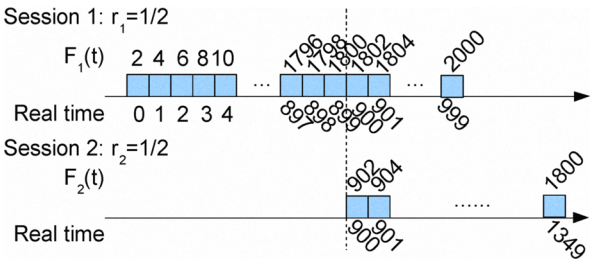

Figure 2. Problem 1 of VC

Flow 1 starts to send packets continuously from time 0, and flow 2 behaves similarly but from time 900. According to VC, $F_{1,901}=1802$ at time 900, while $F_{2,1}=902$. Thus from time 900, packets of flow 1 cannot get any service until the 499th packet from flow 2 has been served. As we can see that packets of flow 1 arrived in the interval [900, 1349) are all postponed to be served for flow 1 was served excessively in the interval [0,900) without affecting flow 2. Obviously, in this a case further refinement is needed.

VC adopts the policy of SFF (Smallest Virtual Finish Time First) in selection of the next packet to be transmitted. As a result, VC might sometimes send a packet earlier than the time with the GPS server. Figure 3 illustrates this situation, where 11 flows share a link with the normalized rate of 1 P/S. Flow 1, whose reserved rate is 0.5 P/S, sends 11 packets continuously starting at time 0, while each of the other 10 flows, whose reserved rate is 0.05 P/S, sends only 1 packet at time 0. For simplicity, we assume that the sizes of all the packets are the same. The value at the top of each block in Figure 3 represents the virtual finish time of the packet. According to the virtual finish times, 10 packets on flow 1 will be served back-to-back before packets on other flows can be transmitted, then the packets from other 10 flows are transmitted continuously before the 11th packet from flow 1 can be transmitted. Packets from flow 2 through 10 have been postponed because the SFF policy in VC is in favor of flows with larger reservations and without considering the virtual start times.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7659
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7659

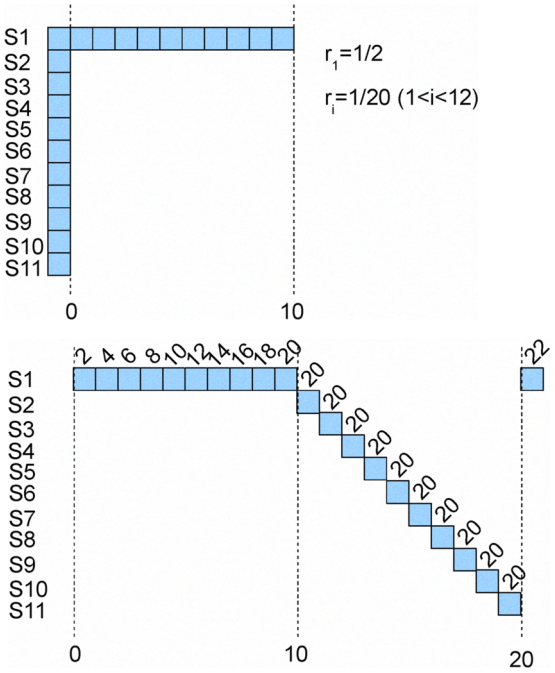

Figure 3. Problem 2 of VC

For the very reasons, we have tried to improve VC algorithm.

III. VISTA: A NEW FAIR QUEUING ALGORITHM

In this section, we will propose a new algorithm that improves VC to solve the problems it suffers from, and at the same time to keep the simplicity of its system virtual time. Since the basic idea of this new algorithm is to force all the virtual start times of HoL (Head of Line) packets to be aligned by subtracting an alignment time, we call this algorithm Virtual Clock with Virtual Start Time Alignment or VISTA for short.

For a VISTA algorithm, we choose the system virtual time the same as in VC:

$$V^{VISTA}(t) = t$$

and let the packets of flow i be enqueued into queue i. Only one $S_i(t)$ and $F_i(t)$ for each queue are maintained, and they represent the virtual start time and virtual finish time of HoL packet of queue i. Virtual times are updated only on the arrivals of new HoL packets of each queue.

HoL packets in a VISTA server can be divided into three classes: Class (1). An unserved HoL packet is still HoL packet; Class (2). A packet becomes a HoL packet after its predecessor departs in a non-empty queue; Class (2). A packet arrives to a previously empty queue (there is no packet in the queue and no packet is send out from the queue in previous time slot), it becomes a HoL packet immediately.

Virtual start times $S_i(t)$ of different classes are calculated differently.

To calculate $S_i(t)$ of the first two classes, the assistant virtual start time $S_i'(t)$ is introduced and calculated as follow:

$$S_i'(t) = \begin{cases} S_i(t^-), & Class\ 1 \\ F_i(t^-), & Class\ 2 \end{cases} \quad (1)$$

where t is the time when there is new HoL packet, and t^- stands for the time just before t.

When all the $S_i'(t)$ of the first two classes are updated, we can calculate the alignment time m by:

$$m = \min_i\{S_i'(t)\} - t, \quad \forall i \in B(t) \quad (2)$$

where $B(t)$ is a set at time t that contains all the flows whose HoL packets belong to the first two classes. Then the virtual start times for the first two classes are aligned from the assistant virtual start time by subtracting the alignment time:

$$S_i(t) = S_i'(t) - m, \quad \forall i\,. \quad (3)$$

For the third class of HoL packet, its virtual start time equals to current system time directly:

$$S_i(t) = t, \quad Class\ 3\,. \quad (4)$$

The calculation of $F_i(t)$ is the same as in VC:

$$F_i(t) = S_i(t) + \frac{L_i^{HoL}}{r_i}, \quad \forall i$$

where L_i^{HoL} is the length of the HoL packet of flow i and r_i is the reserved rate.

For a VISTA server to choose a HoL packet to transmit, instead of using SFF policy as in VC, it adopts a policy called SEFF (Smallest Eligible Finish Time First) [2], which is also used by both WF^{2}Q and WF^{2}Q+. With this policy, the server chooses the eligible HoL packet with the smallest virtual finish time to be next. A HoL packet is said to be eligible if its virtual start time is no greater than the current system virtual time.

For the VISTA algorithm we have the following theorems:

Theorem 1: In a VISTA server All the virtual start times of HoL packets at time *t* cannot be less than *t*.

Proof: Combine Equations (2) and (3), we have:

$$S_i(t) = S_i'(t) - \min_{j\in B(t)}\{S_j'(t)\} + t \quad (5)$$

And $S_i'(t) - \min_{j\in B(t)}\{S_j'(t)\} \geq 0$ is always true, we get:

$$S_i(t) \geq t\,. \quad \blacksquare$$

Theorem 2: In a VISTA server, eligible packets are those HoL packets whose virtual start times equal to the current system time.

Proof: Let Set $E(t)$ to be the set of eligible packets at time t, according to the definition of eligible packets, we have:

$$S_i(t) \leq t, \quad \forall i \in E(t)$$

And according to Theorem 1:

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7659
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7659

$$S_i(t) \geq t .$$

So for all eligible packets we have:

$$S_i(t) = t . \quad \blacksquare$$

Theorem 3: In a VISTA server, at any time, if the queues are not empty, there must be at least one eligible packet.

Proof: According to equation (5), when the queues are not empty, if the HoL packets belong to the first two classes, there must be at least one HoL packet has the minimum assistant virtual start time, and for the generality, we let it be queue *k*, thus:

$$S_k(t) = S'_k(t) - \min_{j \in B(t)}\{S'_j(t)\} + t = t$$

Otherwise, if the HoL packet belongs to the third class, its virtual start time must equal to current system time according to equation (4).

By applying Theorem 2, we know that Theorem 3 is true. ■

Theorem 3 ensures that at any time when there is at least one backlogged queue, an eligible packet can be found. For some of the other algorithms, this is not guaranteed. To let the server work-conserving, some complicated processes should be invoked, but a VISTA server has no such problem.

Now consider again the two problems that VC meets, but this time under a VISTA server.

The first situation is illustrated in Figure 4, in which the value inside each block represents the length of the queue and the values at the top of each block are the virtual start time and the virtual finish time of the HoL packet at that time. Not like in VC, in a VISTA server, the virtual start times is aligned to system virtual time when there is a new active flow (flow 2) at time 900. Therefore, the flow 1 and flow 2 packets can get the service alternately according to their reservation after the arrival of the first flow 2 packet as shown in Figure 4.

Consider another situation shown in Figure 5, which is the second problem that VC meets, and the values above or below each block represent the virtual start time and virtual finish time of that packet. Note that the VISTA server uses SEFF policy when choosing the next packet to send out. At time 0, all the HoL packets are eligible because they all have the virtual

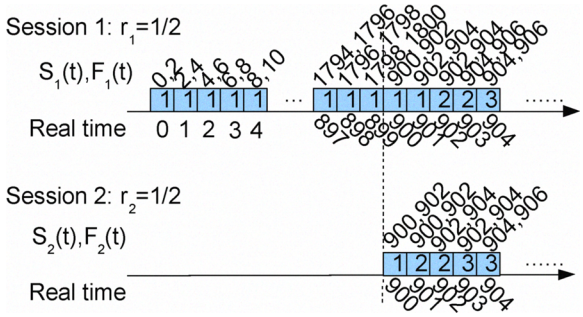

Figure 4. Solution to problem 1

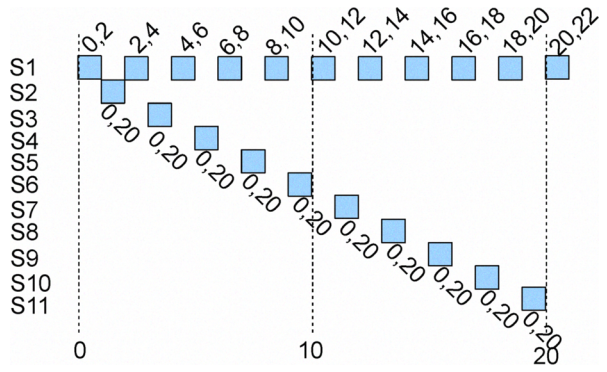

Figure 5. Solution to problem 2

start time of 0. The HoL packet of flow 1 is selected because it has the smallest virtual finish time of 2 while others have virtual finish time of 20. When finished transmitting the first packet from flow 1, the second packet becomes the HoL packet of queue 1. The system state is in Case II, the virtual start time of this packet would be 2, and its virtual finish time would be 4. Now the system virtual time is updated to 1, only queue 1 is not eligible and all the HoL packets of other 10 queues have the same virtual start time and virtual finish time, so a packet from other 10 queues is selected arbitrarily. After that, system virtual time is updated to 2, so the HoL packet of queue 1 is eligible now and has the smallest virtual finish time. The departure sequences are illustrated in Figure 5. The output under the VISTA server is much smoother than under VC and on the same level as that of WF^2Q.

By now, we showed that VISTA is one of the PFQ algorithms that are most similar to GPS, but it is rather complicated. Although the calculation of virtual times is very simple and can be done in constant time, the server has to find out the smallest virtual start time and the smallest virtual finish time among all the HoL packets. Furthermore, it has to align the virtual start times. Both the time alignment and finding of minimum value from an unordered list have very high time complexity in traditional method.

We will introduce a new hardware-based Earliest Packet Selector in the next section, which makes the alignment and finding be done in constant time.

IV. EARLIEST PACKET SELECTOR

In this section, we design a new device called EPS (Earliest Packet Selector) to make the time alignment and eligibility test be done in constant time.

As we all known, data are stored in binary format, and the higher bits of a number are more significant than the lower bits. So the basic idea of finding the smallest virtual start/finish time (called virtual time) is to evaluate the virtual times in parallel of each other bit by bit from highest to lowest. This process is also called scanning. During the scanning, we always exclude the virtual times with 1 at the current evaluating bit, because 1 is larger than 0.

The simple schematic model performing the scanning is illustrated in Figure 6. In this model, each row represents a

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7659
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7659

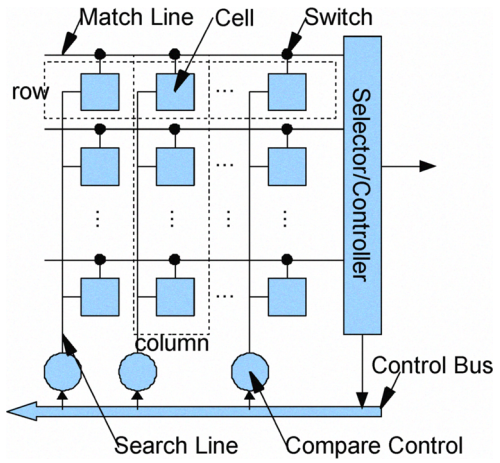

Figure 6. To find the Minimum Value in Constant Time

virtual time with each bit stored in a cell, and each cell is connected to a switch, which is also controlled by the cell. If all the switches on a match line are on, the match line is in "match" state. Otherwise, if any of the switches on a match line is set to off, that match line is "unmatched". At the beginning of a scanning, all match lines are set to match state. Then the controller sends a command to the first compare control through the control bus and, that compare control broadcast the value 0 to all the cells in that column. Each cell compare the value 0 with the value they stores, if they are not equal, the switch is set to off, thus the corresponding match line is set to unmatched state. Then the controller detects the states of the match lines and takes actions according to the four cases of the states as described below. After that, the controller sends another command to the second compare control, and repeat the process until the last bit is compared or the smallest virtual time is found.

During the scanning process, four cases may occur: (1).After comparing some bit (including the last bit), there is only one match line in match state, then the corresponding virtual time is the smallest. The controller sends a command to stop the scanning process. (2).After comparing some bit (not the last bit), all match lines are unmatched, but before the comparing more than one match line is matched. In this case, the controller sends a command to skip comparing current bit and continue to compare the next bit, because the corresponding virtual times must be all have value 1 at current bit. (3).Just after comparing the last bit, all match lines are unmatched, but before the comparison, some match lines are matched, then they must have the same (minimum) virtual time. (4).After comparing the last bit, there are still several match lines are in match state, then they must be equal and both have the smallest virtual time.

Since we only compare one bit (in parallel) at each time, the speed should be very fast. In addition, the time of the scanning process is independent of the number of active flows but only depends on the length of the times in terms of bits. Thus, we can find out the smallest virtual time in O(B) time, where B is the length of virtual times in bits, which is independent of N and is a constant for a specific system.

We must mention that if the compare controls broadcast the corresponding bits of the current system virtual time to each column, the selector can decide which virtual times equal to system virtual time. According to Theorem 2 this property can be used to test the eligibility of the packets in O(B) time, and from Theorem 3 we known that the eligible packets can be found at any time when there is at least one backlogged queue. This simplifies the process.

The concept model of EPS is shown in Figure 7. As we can see, the EPS is a combination of two scanning modules: one is used to compare the virtual start times, which is called the virtual start time scanner (VSTS); and the other is in charge of the comparison of virtual finish times, we would like to call it virtual finish time scanner (VFTS).

The match lines are driven by the queue length detectors, which activate the match lines if the queue lengths are greater than 0. If a queue is empty, the corresponding match line is deactivated, i.e. the match line will be always in unmatched state until there is a packet arrival to that queue.

The number of rows in EPS is equal to the maximum number of possible flows (queues). Each row associates with one queue and is responsible for calculating the virtual start time and the virtual finish time for that queue. Note that in other PFQ algorithms memories are also needed to store the virtual times, but in our algorithm, this "memory" is in EPS and it is also responsible for the calculation and comparison.

The calculation of virtual start time and virtual finish time is only performed on the arrivals of new HoL packets. The virtual start time is updated according to equation (1) for the first two classes of HoL packets. Then VSTS finds out the smallest virtual start time in O(B) time. In addition, the virtual start times of all HoL packets should be aligned by subtracting the difference between the smallest virtual start time and current system virtual time. This can be done in O(1) time by using the parallel subtractors. Let the virtual start times equals to current system time directly according to equation (4). Then, all the virtual finish times are updated with the help of parallel adder. Consequently all virtual finish times are updated according to new virtual start time and the calculation of virtual finish times is done in constant time with the help of the

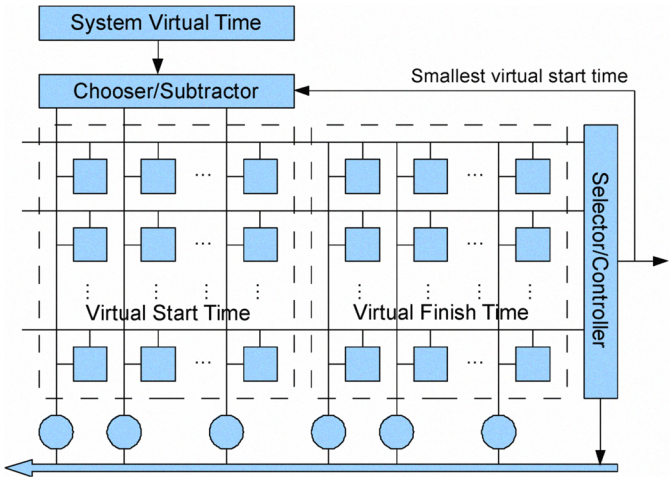

Figure 7. Concept Model of EPS

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7659
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7659

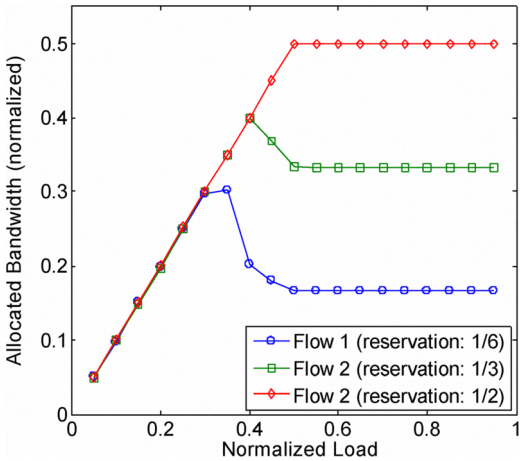

Figure 8. Bandwidth guarantee by VISTA

parallel adder.

The process of finding a packet to transmit consists of two steps: finding eligible packets/alignment of virtual start time and finding minimum finish time.

Step 1: When finding eligible packets, all the start times are comparing with current system virtual time, and all the packets whose virtual start times are no greater than current system virtual time are eligible. This can be done by VSTS in O(1) time.

If there is only one eligible packet, the selector selects it and no further step should be done. If there are several eligible packets, the second step should be invoked.

Step 2: The VFTS can find out all the eligible packets (have been evaluated in last step) that have the smallest virtual finish time in O(B) time. If there is only one packet having the smallest virtual finish time, the selector selects it as the next packet to send out. Otherwise, if there are some packets having the same (smallest) virtual finish time, the selector chooses one randomly.

We should also mention that in other algorithms the maximum value of system virtual time is assumed infinitely large; however, it is limited in a real system. If the system virtual time reaches its maximum value, it turns back to 0. In EPS, when the system virtual time get its maximum value, all the virtual start times of HoL packets are aligned by subtracting this maximum value. This keeps the packets still in the right order after the system virtual time turned back to 0. The process of this operation is the same as normal virtual start time alignment, but the alignment time is the maximum value in this case, so the running time is also in constant time.

V. SIMULATIONS AND RESULTS

Consider the situation shown in Figure 1 again: three flows sharing a link with normalized rate 1 P/S, and the bandwidth reservations of flows 1, 2 and 3 are 1/6 P/S, 1/3 P/S and 1/2 P/S respectively. However, this time, we schedule these three flows with a VISTA server. The loads of these three flows vary from 0% to 90%. As we can see from Figure 8, when the aggregated load is no more than 100%, all flows can be satisfied; when some flows send packets exceed their reservations, the spare bandwidth is shared between the flows according to their reservation. When all the flows send packets more than their reservations (i.e. the aggregated load is more than 100%), the allocated bandwidths are just equal to their reservations. This simulation shows that the VISTA server can guarantee the bandwidths that flows reserved.

In the next simulation shown in Figure 9, we show the delay performance of four different PFQ algorithms. In fact, we do simulations of a lot of algorithms, but only the algorithms which have very similar performances are shown here. In this scenario, the reservations of three flows are the same as the previous simulation. The load of each flow is 55%, and flow 1 keeps sending packets from time 0 and flow 2 begins to send packets from time 500, while flow 3 starts to send packets from time 1000. As we can see from Figure 9, VC punishes flow 1 for its innocent over sending. WF^{2}Q shows better fairness than VC, and MSPFQ, which need some assumption on traffic, shows very similar performance as WF^{2}Q, but not exactly the same, however, VISTA, which requires no assumption on traffic, performances exactly like WF^{2}Q.

The system virtual times of different algorithms during the above simulation is shown in Figure 10. As we can see, VISTA maintains the smallest system virtual time, which means the binary bits (B as introduced in last section) of system virtual time would be smallest. Thus, the EPS for VISTA can find next packet much faster than for other algorithms such as WF^{2}Q+.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new fair Queuing algorithm, which is based on existing Virtual Clock algorithm because of its simplicity of calculating virtual times. We also analyzed the

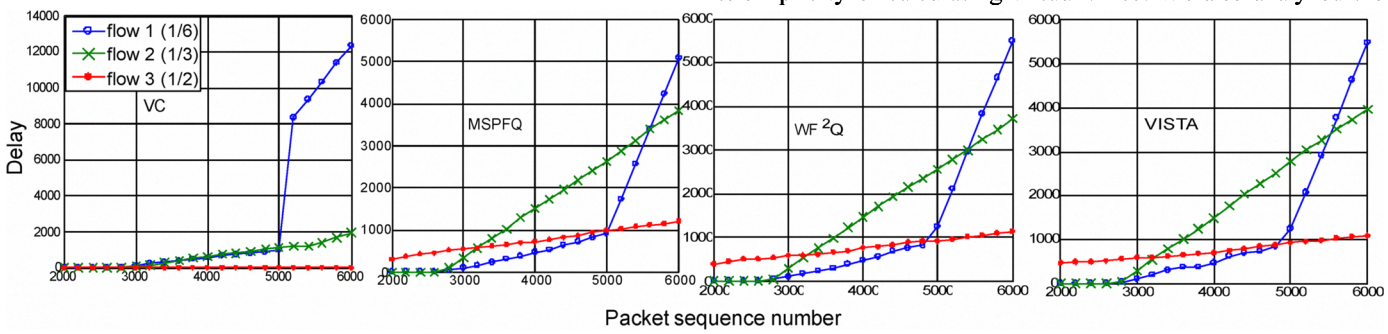

Figure 9. Delay Performance of PFQ Algorithms

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7659
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7659

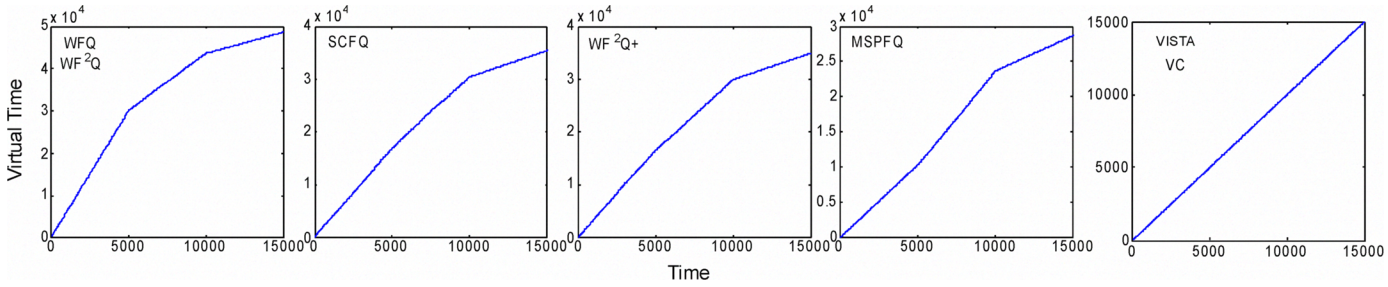

Figure 10. Virtual Time Traces of PFQ Algorithms

disadvantages of VC, and developed the time alignment method to correct them. The basic idea of time alignment method is to align the virtual start times of all HoL packets to current system virtual time when there is a new active flow. By doing this, all the flows can re-allocate the bandwidth according to their reservation. As we have shown in this paper, with the VISTA algorithm and the SEFF policy it adopts, the problems that VC meets are eliminated.

Besides of showing the same performance of VISTA and WF^2Q by using simulation, we could also analytically proof that the latency and WFI of VISTA, which are two most important parameters of PFQ algorithms, are the same in as those of WF^2Q by using the LR server [5] and RP server [19] model, however, the proof is quite long, we would like to show the proof in our next paper.

The time complexity of VISTA algorithm is still high because the time alignment and finding the smallest virtual time are time-consuming operations. To reduce the complexity of the algorithm, we described a Earliest Packet Selector to make the time alignment in O(1) time and to make the smallest time finding and eligibility evaluating in O(B) time, where B is a constant for a certain system. Thus VISTA algorithm can be performed in O(B) time. However, in this paper, only the concept model of EPS is shown, and we would like to complete the design and implementation of EPS in the future work of our lab.

It has been shown that in a GPS server, if the traffic is limited by leaky bucket or token bucket, its delay can be bounded [4], thus the QoS can be guaranteed. Since VISTA is one of the algorithms fitting into the GPS server most closely, we can say that the QoS can be guaranteed in a VISTA server.

ACKNOWLEDGMENT

The work presented in this paper is supported by Chinese Natural Science Foundation (Project No. 60773102) and Sichuan University (Project name: Next Generation Internet Architecture). The authors would like to acknowledge the financial support from CNSF and SCU.

REFERENCES

[1] D. Stiliadis, "Traffic Scheduling in packet-switched networks: analysis, design, and implementation," Ph.D. dissertation, University of California,Santa Cruz, 1996.

[2] J. Bennett and H. Zhang, "WF2Q: worst-case fair weighted fair queueing," *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, 1996, pp. 120–128 vol.1.

[3] J. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *Networking, IEEE/ACM Transactions on*, vol. 5, 1997, pp. 675–689.

[4] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *Networking, IEEE/ACM Transactions on*, vol. 1, 1993, pp. 344–357.

[5] D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," *Networking, IEEE/ACM Transactions on*, vol. 6, 1998, pp. 611–624.

[6] S. Cheung and C. Pencea, "BSFQ: bin sort fair queueing," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2002, pp. 1640–1649 vol.3.

[7] S. Suri, G. Varghese, and G. Chandranmenon, "Leap forward virtual clock: a new fair queuing scheme with guaranteed delays and throughput fairness," *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 1997, pp. 557–565 vol.2.

[8] L. Zhang, "Virtual clock: a new traffic control algorithm for packet switching networks," *SIGCOMM Comput. Commun. Rev.*, vol. 20, 1990, pp. 19–29.

[9] X. Zhang and L. Bhuyan, "Deficit round-robin scheduling for input-queued switches," *Selected Areas in Communications, IEEE Journal on*, vol. 21, 2003, pp. 584–594.

[10] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," *INFOCOM '94. Networking for Global Communications., 13th Proceedings IEEE*, 1994, pp. 636–646 vol.2.

[11] S. Bakiras, F. Wang, D. Papadias, and M. Hamdi, "Vertical dimensioning: A novel DRR implementation for efficient fair queueing," *Computer Communications*, vol. 31, Sep. 2008, pp. 3476–3484.

[12] C. Guo, "SRR: an O(1) time-complexity packet scheduler for flows in multiservice packet networks," *Networking, IEEE/ACM Transactions on*, vol. 12, 2004, pp. 1144–1155.

[13] L. Lenzini, E. Mingozzi, and G. Stea, "Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers," *Networking, IEEE/ACM Transactions on*, vol. 12, 2004, pp. 681–693.

[14] D. Kwak, N. Ko, and H. Park, "Mean starting potential fair queuing for high-speed packet networks," *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, 2003, pp. 2870–2874 vol.5.

[15] S. Ramabhadran and J. Pasquale, "The Stratified Round Robin scheduler: design, analysis and implementation," *IEEE/ACM Trans. Netw.*, vol. 14, 2006, pp. 1362–1373.

[16] J. Xu and R.J. Lipton, "On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms," *IEEE/ACM Trans. Netw.*, vol. 13, 2005, pp. 15–28.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7659
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7659

[17] H. Chao and N. Uzun, "A VLSI sequencer chip for ATM traffic shaper and queue manager," *Solid-State Circuits, IEEE Journal of*, vol. 27, 1992, pp. 1634–1643.

[18] K.G. Harteros, *Fast Parallel Comparison Circuits for Scheduling*, Inst. of Computer Science, 2002.

[19] D. Stiliadis and A. Varma, "Rate-proportional servers: a design methodology for fair queueing algorithms," *Networking, IEEE/ACM Transactions on*, vol. 6, 1998, pp. 164–174.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7659
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7659