Proceedings of the 5th WSEAS International Conference on Applications of Electrical Engineering, Prague, Czech Republic, March 12-14, 2006 (pp123-130)

Providing Delay Constraint Services with the Palindrome Scheduler Schemes

John Tsiligaridis
Math and Computer Science,
Heritage University, USA
3240 Fort Road Toppenish, WA, 98948

Raj Acharya
Computer Science and Engineering,
Penn. State University, USA
220 Pond Lab., University Park, PA 16802-6106

Abstract:- Multimedia communications involving digital audio and video has rather strict delay requirements. A Palindrome Schedule (PS) can guarantee service of real and non real-time packets. The deadline monotonic priority assignment (or inverse –deadline) is achieved by the component of PS called the scheduler table (STAB). Additionally we consider strict streams deadlines. For the service of the real time packets two proposed schemes are introduced. The first is a partitioning scheme, the Upper Bound (UB) scheme and the second is the Group Sharing (GS) scheme. The idea of a successive increase of the weight enables the server meet the strict deadlines of consecutively serviced packets streams. The technique of the cooperative flows makes the scheduler able to meet the deadlines of a pair of consecutive flows (in different queues) and provides a new solution to the problem of scheduling multiple tight delay constraints streams. To this direction two algorithms are provided: the Increasing Weight Algorithm (IWA) and the Cooperative Flows Algorithm (CFA). The framework of PS, including the Compound Round Robin (CRR) for large size non real time packets and the IWA with CFA for real-time packets, with the two schemata, will overcome the service difficulties of the streams with (m,k)-firm deadlines and will provide a new dimension for servicing real-time packets under tight deadlines (multimedia applications). Simulation results are provided.

Key-Words:- scheduling, QoS, real-time scheduling, network management

1. Introduction

High speed networking offers opportunities for new multimedia applications (such as tele-medicine, virtual environment, etc) and has stringent performance requirements in terms of throughput, delay , delay jitter and loss rate[1].

In our previous work [2] we presented the framework of PS (PS with two adaptive credit algorithms CRR, DIW) that serve non real-time packets. PS working as timestamp scheduler avoids the use of virtual times and the sorting priority operation and finally ensures the priority of the flows while the second one, which belongs to the round robin scheduler, provides two feedback mechanisms: the Composed Round Robin (CRR) and the Integrating Weight Algorithm (IWA) that guarantees the service of large size non real-time packets (l-packets) and service of the real time data respectively.

For the real time (or time constraints) flows the deadline regulates the behavior of the server. The deadline can be considered as the *latest time* a packet can be serviced and is determined by the specification of the maximum allowable time of servicing consecutive packets (session) of the same stream. These packets are serviced according to their arrival priority while for the non real-time packets we can assume that a number of them can be sent by the server later. In [3] the feasibility of providing real-time services is examined in Wide Area Network. The proposed system can provide a service approach better than the (m,k)-firm deadlines flows do [4] since the instantaneous increase of the weight makes the server able to meet the deadline for all the consecutive packets of a flow. Additionally, PS, can avoid the "empty times", the time that the scheduler spends around the queues until finding a non empty one, by using the scheduler table (STAB) information. The PS offers a different approach than the Dynamic Window-Constraints Scheduling (DWCS) [5] DWCS deal with loss tolerance, is more complicated and without strict deadlines, while our system overcomes all these difficulties and guarantees the service of the streams by using the

service cooperation of consecutive streams. A similar cooperative policy is applied to multihop networks upon failure of primary channel its backup is activated in order to guarantee the better QoS [9]. Analogous schemata with reservation bandwidth constraints have been developed for Differentiated Services MPLS Traffic Engineering in [10].

For the *weight sufficiency problem* , instead of *the* continuous of back and forth transition of credits (as it happens with DRR) a new more systematic approach, *the instantaneous increase of the weight,* is presented. Some of the probable reasons that support to this idea for a flow while the scheduler is "on the fly" can be considered : (1) the short size of the Maximum Transfer Unit (MTU), to avoid fragmentation of segments given the short size of MTU [6], (2) the violation of the Service Level Agreement (SLA) is usually a result of the internal network conditions and not of the particular bad source's behavior, (3) the Latency the service delay of some flows can cause further delay to all the other flows as well, (4) for new services (or applications), like video on demand, an increase of the weight is needed in order the server meet the deadline of a flow. All the routers along a delivery chain must be able to support the desired type of QoS.

The *assumptions* on the *system design* and fairness are presented below so that the operation of the proposed algorithm cannot be blocked. For the non real-time traffic, a *predefined maximum weight* (p_weight) for each flow (not necessarily all equal) is considered but going beyond this limit is not allowed. The scheduler can cope with all the flows asking simultaneously to use the p_weight of each flow *simultaneously*. The sum of credits assigned to all of the queues by adaptive schemes, should be fixed at a sum of T. *The necessary condition for servicing the l-packets is p_weight ≥ MSLP(maximum size large packets).* For the non real-time flows, instead of punishing the flow in the next rounds (penalty solutions, as DRR,SRR provide), it is preferable *to increase the weight provisionally and diminish the service delay by speeding up the service*. More details for the service of non real-time packets are included in [2]. For the real-time flows an increase of the weight with IWA when the server tries to meet the deadline, is also used. Two schemes: the Upper Bound (UB) scheme and the second is the Group Sharing (GS) scheme are presented.

The rest of the paper is organized as follows. In section 2, the environment of the PS for the real-time flows is described. In section 3, some analytical results with the function conditions are presented. The cooperative flows algorithm (CFA) is developed in section 4. The fairness of the service of collaborative flows is examined in section 5. Simulation results are presented in Section 6.

2. PS servicing real-time flows

This work is an extension of our previous work [2]. For the non real-time flows we used the DRR scheduler (for the normal size packets) and the CRR (for the large size packets when the basic condition is not valid – Theorem 2-) [7],[2]. Our new step, is the service of real time packets with or without the presence of the non real-time packets.

When strict deadlines apply the server ability to meet the deadlines is unlikely, and so the streams are dropped. The scheduler whenever a new stream arrives, re-estimate the deadlines of the two streams (old and new) and readjusts the new weights in order to meet the deadline of the second flow.

This work focused on the problem of servicing real time packets and not real-time especially when we have frequent real-time flows with strict deadlines. Our attempt is to minimize the possibility the streams to miss their deadlines as a result of queuing delay and because of the strict deadlines. We propose the framework of PS with CRR for non real-time traffic [2] and here, with the Increasing Weight Algorithm (IWA) for the delay constraints traffic, we provide a solution to the problem of scheduling multiple streams of both types traffic. The IWA is used in order to calculate the appropriate scale weight needed for service of a flow. One further step is to estimate the necessary increase of weights for two consecutive (cooperatives) servicing queues, using the Cooperative Flows Algorithm (CFA). With the technique of the *cooperative* flows the service ability of two real-time consecutive flows (in different queues) is examined. The server can provide additional weight for flow i *not only at the beginning* of the service but also at any point of time during the service in order to be able to meet the deadline of the flow i+1. The system of cooperative flows can provide solutions in cases where other service solutions fail.

There are two proposed schemes: the Upper Bound (UB) scheme and the Group Sharing (GS) scheme. The UB scheme offers a bound for each flow

p_weight while for the GS scheme a common pool for bandwidth reservation is held for a group of flows *(group pool)* that probably have some common characteristics. For the UB scheme we assume that the available bandwidth is taken in scale (an integer multiple of the packets size). The same is considered for the GS using the group pool. For finding the best fit of the increase of the weight the IWA is developed.

The IWA first finds the *total time* that the packets of a real-time flow need in order to be serviced and then decides for the feasibility of the service. If, with the existed weight, the service time of the packets of a stream is greater than their deadline a move to the next increase scale of the weight (integer multiple), is required. It is considered that *the weight is equal to the multiple size of the packet of this stream* so that working like DRR , where a continuous and endless change of credits' amount in every round occurs, is avoided. We also consider that: (1) the packets of the same stream all have equal size,(2) in a time unit, the scheduler services one or multiple number of packets of a real-time flow with the existed weight. (3) the PS can work like a frame-based server (4) *with the additional weight (≤p_weight) any stream becomes feasible.* The IWA works as follows:

```
The IWA psedocode (for the real time traffic):
// num_packets : is the # of packets in the stream
// service_time_stream : is the service time of a
stream
// packet_size : is the size of a packet of the stream
// deadline: is the latest allowed service time of the
//    packets of a real-time stream
// weight : is the existed weight ; weight=0;
for each arrived real-time stream
  {        // with the existed weight
  service_time_stream = num_packets
                          * serv. time /packet
  if (service_time_stream > deadline)
    { //increase the weight by a k (minimum integer )
      //      multiple packet size
     k=1;  //initial value
        do
          { weight = weight + k * packet_size
*           if (weight ≤ p_weight )  //for UB
              k=k+1; }
        while (the stream is not feasible) }
    }
```

The if command with the star (*) for the GS scheme can be replaced by:

if (weight ≤ available weight of the group pool) //for GS

From all the above we summarize that when non real-time streams arrive the PS works as Deficit Round Robin (DRR) (l-packets included). The real-time stream is selected from STAB according to EDF and an *exhaustive* service (service of all the packets of the session) follows. When the service of a real-time stream is not feasible with the existence weight, PS using the IWA borrows the additional bandwidth needed (≤ p_weight) until the end of the service. After the completion of service of the real-time flows the server will continue the service of the previously suspended non real time flows.

3. Some Analytical Results

A scheduler is considered as *efficient,* only if the whole work complexity for enque ueing and dequeing is O(1) for each packet. The work complexity of a packet is O(1) only if it is certain that *at least* one packet for each flow is served within a round. There are two *main operations* of our scheduler: the *enqueue* and the *dequeue* [7].

Theorem 1. The proposed increasing weight algorithms (CRR, IWA) are O(1) complexity.

Proof. We have to confirm that the enqueuing and dequeuing operations are held within a constant period of time for each flow. For the enqueuing process it takes O(1). First, the new packet is inserted in the appropriate flow and if the flow id is not in the ActiveList, the flow is added to it. The addition of a packet to the end of a linked list takes O(1) operation. For the dequeuing process the scheduler determines which flow is going to be serviced next. The additional weight is offered in definite number of steps with CRR or IWA up to the maximum permitted value (p_weight). The dequeing process (when only non real-time packets are in lines) update the ActiveList etc [4] held at a constant period of time (O(1) complexity), because all of them represent a constant number of operations. ■

Theorem 2: The *service conditions* for a non real-time data flow i is:

$p\_weight \le weight_i + rem_i \ge packet\ size_i$ (1)

and for the real-time data is: $(\text{total bits of the packets stream})_i / p\_weight_i \le deadline$ (2)

Proof: For (1), the rem $_i$ (the remainder of bits of the previous round i-1 that can be used in the next round i) with the weight for service of the packets must not less than the packet size (DRR fashion). For (2) means that the flow i is feasible since it can be serviced in time or ahead of its deadline. ■

For the case of real time streams with strict deadlines a new approach is developed. The key idea is that the service time of a flow often provides delay to the next servicing flow. We define as *cooperative* flows those flows whose service time affects the service time of the others.

Definition 1: Two flows (f_1,f_2) can be considered *cooperative* when the arrival of the f_2 occurs while the f_1 is being serviced.

Definition 2: N flows (f_1,f_2,..f_n) can be considered cooperatives when the arrival of all flows except f_1 occurs while f_1 is being serviced.

Definition 3: The *service condition for two cooperative flows* f_1,f_2 *(SCC)* is: $r\_st_1 + st_2 \leq d_2$, where: r_st_1: is the service time of the remaining packets until the end of service of f_1, st_2: is the service time of f_2 .

Lemma 1: For the two cooperative flows (f_1,f_2) with deadlines (d_1,d_2) and weight (w_1,w_2) the scheduler decides (on the fly) for a possible *increase* of the w_1 (w_{11}) if: $r\_st_1 + st_2 > d_2$

Proof: We know that $wt_2 = r\_st_1$, where wt_2 is the waiting time of f_2 until the start of service. In order to minimize the wt_2 of the f_2 the scheduler increases the w_1 (probably by the next scale value) so that the new estimated service time (r_st_1) of the rest of the packets of f_1 plus the st_2 becomes less than d_2. This can be expressed as: $r\_st_1 + st_2 < d_2$. It is apparent that: $w_1 < w_{11}$. The same happens when we have more than one cooperative flows where we can take: $w_1 < w_{11} < w_{12}$.

Another solution of course could be to have an increase of the w_2 at the beginning of the service of the f_2, and/or a *combination* of the service of the two flows (Lemmata 2-4).

Lemma 2: The service ability that a f_i can offer to the cooperative next f_{i+1} must be *at least* one time service unit (one round*).*The *maximum* service ability occurs when the rest of f_i can be serviced in a time unit *(best case)*

Proof: It is evident that if f_i has packets that need to be serviced (r_st_i), while the cooperative f_{i+1} has to start being serviced, the server has to minimize the service of f_i so that the service of f_{i+1} be in effect as soon as possible. This may occur after the next time period. The transition from service fi to f_{i+1} occurs at least after one service time unit. The server finds the new weight $w_{i,i+k}$ (where k is the most appropriate weight's scale) ($\leq$ p_weight) that can serve the rest of the packets of fi in at least 1tu. If this service time is only 1tu, then we have the best case solution for both the f_1 and f_2.

Lemma 3: For two cooperative flows (f_1 and f_2) if the service of f_2 seems unlikely, (because the bound of servicing f_2 is too strict), even after the f_1 weight increase, the f_2 weight increase is necessary . The *combination of the increase of the two weights enable the scheduler* to make feasible the deadline of f_2.

Proof: For the two cooperative flows f_1,f_2 the server makes feasible the deadline d_2 if $r\_st_1 + st_2 \leq d_2$ (SCC). But in case that $r\_st_1 + st_2 > d_2$ then two *consecutive* actions can take place: (1) an increase of the weight of the f_1 (w_{1k} , k =1,.., n_1), (2) or/and a gradually increase of f_2 (w_{2k} , k =1,.., n_2) when needed. If after (1) and /or (2) the SCC becomes valid, then the server with the new weight w_{2k} will serve f_2 in time.

Lemma 4: For n cooperative flows the condition for servicing the next flow is based on the $r\_st_{n-1} + st_n \leq d_n$ for different values of: w_1 (w_{11},..w_{1n}), w_2(w_{21},…,w_{2n}) …w_n(w_{n1},w_{n2},…,w_{nn})

Proof: We apply the SCC repeatedly for each pair of the flow in a consecutive manner (f_1, f_2 ,…, f_n) and we can take the general formula.

An illustrative example is presented bellow.

Example 1:

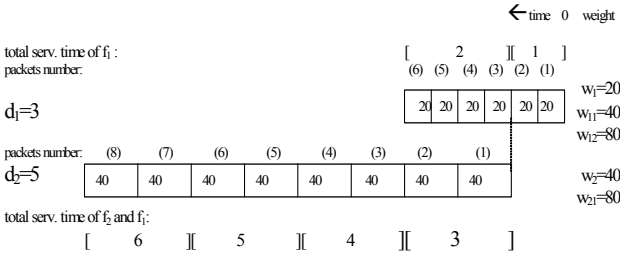

Fig. 1 The service of the two real-time cooperative flows (f_1 , f_2)

We consider a real time flow (f_1) with 6 packets of 20b each with deadline ,d_1=3 at time t=0. The weight (w_1) of the flow is 20b. So the time to be serviced is 6 tu. An increase of the weight is needed. So the new weight w_{11} will be 2*20b = 40b and the service will be completed at 3 tu. In the meantime a new arrival of a real time flow (f_2) with 8 packets of 40 b and deadline d_2=5 appears at time t=1. If the scheduler

keeps the same size of weight until the end of the f_1 it will finish the service at t=3 and f_2 will finish the service with the weight (80b), (w_2 = 40b, w_{21} = 80b) at time t=6 . Thus the f_2 can not be feasible if the w_{11} remains the same. A new increase of the weight (w_{11}) is needed. The w_{12} must be 80 so that the service of f_1 ends at the end of t=2. The service of f_2 starts at time t=3 and with weight w_2 = 80 will finish after 4tu. The f_2 will be feasible only after the increase of the weight of f_1 from 40b (w_1) to 80b(w_{11}). The total service time of f_1, and f_2 will be 6 tu (2tu for f_1 and 4tu for f_2). In Fig.1 the total service time of f_1, f_2 appears in two separated lines.

4. The Cooperative Flows Algorithm (CFA)

The CFA investigates two flows f_1, f_2 in respect of their need for additional weight and service feasibility. It works as follows:

```
CFA:
//two cooperative flows (f1 ,f2 ) are considered
//flow f1 is being serviced, with w1 (from IWA)
//while f1 is being serviced a new arrival f2 happens
// the st2 is computed according to the w2 = p_size2
//r_st1 : the service time of the rest of the packets of the
        flow being serviced (flow 1)
if  r_st1 + st2 > d2 { no increase to w1 is needed ,
                       exit}
if  r_st1 + st2 < d2 {
   //1st action: an increase of w1 is needed
                          (from current flow f1)
   the new weight (w1k), (k=1,..n, n : is the max scale
                          value)
     is finding (using IWA , take the next scale etc)
                examine the f2 feasibility,
     if f2 is feasible then exit //not examination of w2
     if f2 is not feasible
     //2nd action: an increase of w2 is needed
                          (from next flow f2)
       {
        a new increase of the weight of f2 is needed
        w2k (k= 1,..n,  n : is the max scale value)
        take any next scale in order f2 become feasible
        } }
```

5. Fairness

For the fairness measure (FM), we follow the idea of Golestani [8].The worst case difference for the normalized service received by two different backlogged flows at any time interval, is examined. The fairness of the i and j flows, in an interval (t_1, t_2) is defined as the difference between the number of sent packets: $sent_i(t_1, t_2) / f_i - sent_j(t_1, t_2) / f_j$, where f_i , f_j are the share of the i and j flow respectively. For the service discipline of the ideal fluid-flow model, offering small increments of service, the FM becomes zero. We found that a flow servicing with UD policy have the same relative fairness with DRR. Examining the fairness, we can distinguish the case of two cooperative flows where the first flow is backlogged with UD and the same happens with the second one. We consider that the fairness for the DRR is M. Comparisons with DRR are provided.

Lemma 5. Let us consider an interval (t_1, t_2) during which flows i and j are backlogged consequently. For the i flow, after k rounds, for the UB policy we have:

$$k^* (m-1)^* p\_size_i / f_i \le sent_i(t_1, t_2) \le k^*(m+1)^* p\_size_i / f_i$$

Proof: We consider two queues (with one flow each) with the real-time policy service discipline, UB. The server in each round services integer number of consecutive packets.

For k rounds, the flow i during this interval, the number of bytes that will be served is k* weight. Let us $serv_{k,i}$ is the number of serviced packet during the defined interval. for flow i until the k round, we can take:

$serv_{1,i} = weight_{1,i}$

$serv_{2,i} = weight_{2,i}$

…..

$serv_{k,i} = weight_{k,i}$

We have different values of the weight for the i flow so that the server be able to see the deadline of the second flow (flow j). Adding all the equations and considering that

$$all\_serv_{k,i} = \sum_{j=1}^{k} serv_{j,i}$$ we can take:

$$all\_serv_{k,i} = \sum_{j=1}^{k} weight_{j,i} \qquad (3)$$

We consider that $\forall k \in S$, where S is the set of the rounds:

$$weight_{k,i} = m * p\_size_i \le p\_weight_i \qquad (4)$$

where: m: is an integer, that defines the scale of the weight, $m \leq n$ (the maximum number of scales for a flow) , so that $n * p\_size_i = p\_weight_i$,
p_size: is the packet size, p_weight_i : is the maximum permitted value of the weight for i flow.
For the DUB, it is apparent that: $m * p\_size \leq p\_weight_i$. We also consider that: minwei = min ($weight_i$) and f_i stands for the share for the i flow. Consequently, $f_i = weight_i / minwei. \leq k* p\_weight_i$
Using (3) and (4) : $all\_serv_{k,i} = k* m* p\_size \leq k*(m+1)* p\_size$ (5)
It happens: $k*(m+1)* p\_size \leq k* p\_weight_i$
We consider: $sent_i(t_1, t_2) = all\_serv_{k,i}$
Divided by f_i ,we have:

$$sent_i(t_1, t_2) / f_i \leq k*(m+1)* p\_size_i / f_i \quad (6)$$

From (5): $all\_serv_{k,i} = k* m* p\_size \geq k* (m-1)* p\_size$
In the same way we take:

$$sent_i(t_1, t_2) / f_i \geq k* (m-1)* p\_size / f_i \quad (7)$$

From all the above the Lemma has been proved. ■

Theorem 3. The Relative Fairness Measure (RFM) for the DUB is: *M*
Proof: From (6) we have:

$$sent_i(t_1, t_2) / f_i \leq k* (m+1)* p\_size_i / f_i$$

For a second flow j according to (7) we take similarly:

$$sent_j(t_1, t_2) / f_j \geq k' *(m-1) *p\_size_j / f_j$$

(where $k' \geq k-1$, or $k- k' \leq 1$)
After multiplying by (-1): $-sent_i(t_1, t_2) / f_j \leq (- k') * (m-1) * p\_size_j / f_j$
We consider that:

$$(m-1) * p\_size_j / f_j \leq (m+1)* p\_size_i / f_i \leq M$$

(k=1, ..,n , n=total number of active flows)
Adding these two inequalities we take:

$$sent_i(t_1, t_2) / f_i - sent_j(t_1, t_2) / f_j = k* (m+1)* p\_size_i / f_i + (- k') *(m-1) *p\_size_j / f_j \leq k * M + (- k') M \leq M*(k- k') = \leq M \text{ (like DRR)}$$ ■

6. Simulation

In our simulation experiments, we have assumed that we have isolated flows and each flow has its own queue. The space of buffers is non-restricted. A system with three levels is developed; the Application, the Queue and the List level. In the *Queue level,* we use the enqueue and dequeue functions according to each queue's service policy. These two essential functions are finally performed in the last internal *list level* with the corresponding functions of *insertAtBack* and *remove FromFront* of the linked lists. *The PS is developed in the Application level (before the IWA, CRR) in order to find, according to the STAB, the next flow for service.*
We have used Poisson arrivals, random size packets, queues with the same size packets (normal or large) for the non real-time data and have developed four scenaria.
Scenario 1: Three flows, (from which the first is delay constraint) are serviced. The first flow has the service priority and the after finishing the service the other two are serviced. Again CRR has better time results than DRR.(Fig.2)
Scenario 2: Four flows, (from which the first two are delay constraint) are serviced. For the first flow an increase of the weight to the second scale is needed in order the flow to be serviced in time. The weight from 200b becomes 400b (according to IWA) so that the server can meet the deadline of 720 tu for a total size of 288kb. The last two flows give the same results as it has been explained above. Since the deadlines are loose there is no interference between the two first flows. (Fig.3)
Scenario 3: (UB schema) Three real-time flows are served according to UB schema. Analytically:
f_1: 5 packets of 50b, w_1 = 50b, d_1 =5tu, arr time = 0tu,
f_2: 4 packets of 100b, w_2=100b, d_2 =3tu, arr time= 3tu,
f_3: 6 packets of 300b, w_3=300b, d_3=2tu, arr time= 4tu,
- the server starts the service of f_1 at t=0, and since s_packet = 50b the w_1 = 50b (IWA). At t =3 the f_2 arrives. The IWA estimates the service time of f_2, with w_2=100b, and estimated the end of the service: at t=4. Since $4>d_2$, the re-estimation (CFA) of the additional weight for the f_1 is needed. Thus w_{11}= 3*50=150 b and the r_st_1 = 1 (base case). Total service time f_1=3tu
- at t=4 the server starts servicing f_2, queuing delay(q_delay_2 = 1tu) . The total service time of f_2 (t_st_2) $\leq d_2$ is equal to queuing deadline (q_delay_2) and the service time (st_2). It is obvious that: $t\_st_2 = q\_delay_2 + st_2$, $t\_st_2 \leq d_2$ or $st_2 \leq d_2 - q\_delay_2 = 3-1 = 2tu$. The IWA starts with w_2=100b. CFA computes

the additional weight (w_{21} must be 200b) so that the estimated service time of f_2 (400b) become 2tu (st_{21} = 2).
Finally the total service time f_2=3tu, and the queuing delay (q_delay_2) = 1tu.
- at t=5 scheduler sees (scheduler table) that f_3 has arrived and starts servicing. The scheduler ends the service of f_2 in one round (q_delay_3=1tu). CFA decided not to increase the w_2 because f_2 will be serviced in the next round (by the w_{21} = 200b) and increases the w_3 (w_3=300b) in order to service the f_3 in 2 tu. The new weight is w_{31}=600b and $t\_s\_t_3$ = 3tu, q_delay_3=1tu (Fig.4). In Table 1 an illustration of the server's work is given.

serv. time(tu)	1	2	3	4	5	6	7	8
f1,#pac. serv:	1	2	3,4, 5					
$weight_1$:	50	50	150					
q_delay_2 (tu)			1					
f2,#pac. serv:				2	2			
$weight_2$:				200	200			
q_delay_3 (tu)					1			
f3,#pac. serv:						2	2	2
$weight_3$:						600	600	600

Table 1. The server plan for service of the f_1, f_2,f_3

7. Conclusion

The importance of the proposed framework is that the PS using CRR with IWA and CFA can provide service priority, and flexibility on service of the real-time packets with strict deadlines. Additionally, the UB scheme provides relative fairness while the GS better performance. The usefulness, the efficient and easy implementation makes this framework amenable to multimedia applications for the several parts of the network (routers).

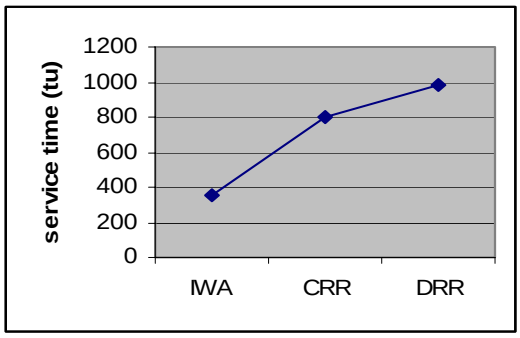

Fig. 2 A real-time flow(IWA) servicing with CRR and DRR

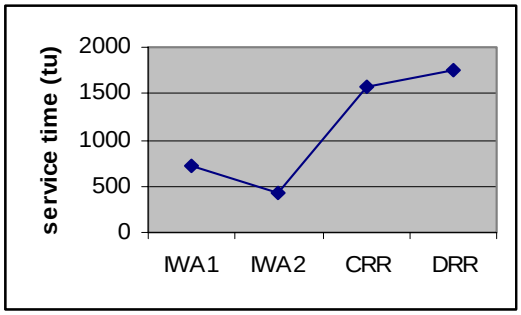

Fig. 3 Two real-time flows (IWA1, IWA2) servicing with CRR, DRR

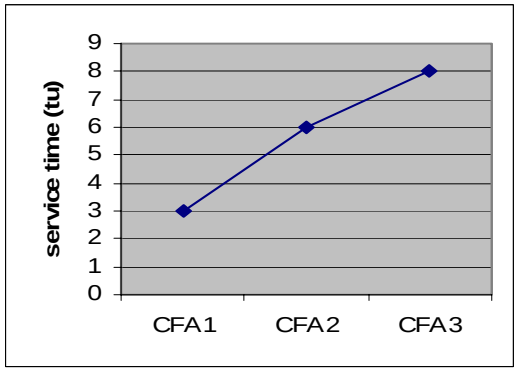

Fig. 4 Three real-time flows consecutive are served (UB scheme)

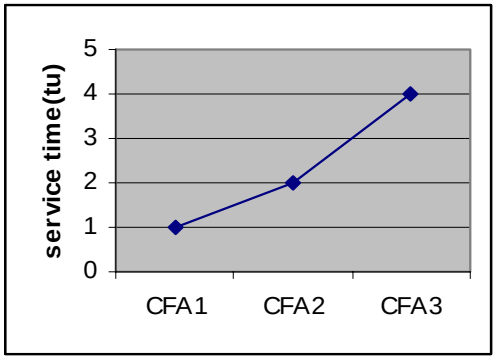

Fig. 5 Three real-time flows consecutive are served (GS scheme)

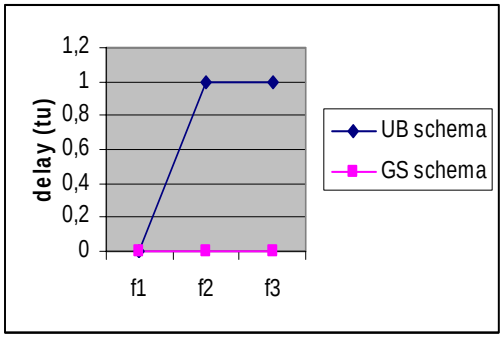

Fig. 6 Delay for the three flows for both schemata

References:

[1] D.Ferrari, "Client requirements for real-time communication services", *IEEE Communication Magazine* , 28(11), November 1990.

[2] John Tsiligaridis , Raj Acharya, "A hybrid framework of RR Scheduler to ensure priority, low complexity and delay with relative fairness", *in Fifth IEEE International Symposium and School on Advance Distributed Systems ISSADS2005*,January 24-28 Guadalajara, Jalisco, Mexico, Proceedings in *Lecture Notes in Computer Science*, Springer, Title: Advanced Distributed Systems, Vol. 3563/2005, .ISSN: 0302-9743, DOI: 10.1007/ 11533962_11

[3] D.Ferrari, D. Verma, "A Scheme for Real-Time channel establishment in Wide-Area Networks", *IEEE JSAC* , vol8, no.3, pp.368-379, Apr. 1990.

[4] M. Hamdaoui, P. Ramanathan, A dynamic priority Assignment technique for Streams with (m,k)-Firm Deadlines, *IEEE Trans. on Computers,* Vol. 44,No. 12, Dec.1995

[5] R. West, K. Schwan, "Dynamic Window-Constrained Scheduling for Multimedia Applications", *Proceedings, IEEE Int. Conf. on Multimedia Computing and Systems, ICMCS 99*, Vol II, July 1999,

[6] A. Tanenbaum, *"Computer Networks"*, Prentice-Hall, 1996.

[7] M.Shreedhar, G. Vargese," Efficient Fair Queuing using deficit Round Robin", *ACM/IEEE Trans. on Networking*, Vol. 4, No3, June 1996, pp. 375-385

[8] S.Golestani, "A self-clocked fair queueing scheme for broadband applications", *IEEE Infocom'94*, Toronto, CA, June 1994, pp. 636-646.

[9] K.Gummadi, M. Pradeep, C. Murthy "An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks", IEEE /ACM Transac tions on Networking, Volume 11, Issue 1 , February 2003 , pp.81-94

[10] J. Ash, "Max Allocation with Reservation Band- with Constraints Model for DiffServ-aware MPLS Traffic Engineering & Performance Comparisons, IETF Internet <draft-ietf-tewg-diff-te-mar-04.txt>, January 2004