Fair Scheduling over multiple servers with flow-dependent server rate

Satya R. Mohanty and Laxmi N. Bhuyan
satya,bhuyan@cs.ucr.edu

***Abstract*— We address the Quality of Service (QoS) provisioning problem in an aggregated multi-server environment. The input packet stream traffic to the system is categorized at a macro level into a few "flow-classes" that require service differentiation; each class is further subdivided into "flows" at the micro-level. Flows can be serviced by any server: however; the service rate is a function of the class and the particular server. Given such an environment, we have a multi-criteria optimization objective (i) provide differentiated service to flows (ii) achieve load balancing of the servers and (iii) maximize their throughput (amount of bytes serviced). We present an on-line fluid-based approximation scheme to schedule packets. Modeling the accumulated traffic in a class as fluid we use linear programming to first determine the optimal fractions that should be directed to different servers while ensuring fairness and high server throughput. We propose a packet scheduling strategy for the multi-server framework (by extending a well-known fair round-robin algorithm for single link systems) that effectively incorporates the optimal service fractions determined in the previous step. We validate the proposed algorithm with extensive simulations. The results show that the algorithm imparts high throughput with good service differentiation. We also evaluate reordering of packet requests within flow streams by presenting relevant metrics that quantify reordering.**

I. Introduction

Applications using the ubiquitous Internet infrastructure like video and audio conferencing, image processing, distributed gaming, peer-to-peer applications etc. continue to proliferate. Diverse applications have different service requirements and needs. Service differentiation is usually accomplished by dividing the traffic stream into disjoint *flows* (a distinguishable source-destination pair, whose data packets are always stored in a logically distinct per-conversation queue at a router), and treating the flows in accordance with their reserved rates of service.

We consider aggregated multi-server systems in which each server can perform a variety of services. In addition to the increase in throughput because of availability of many processing entities such systems have the added advantages of increased reliability and load balancing. The input to these systems are different flows with varying service requirements. Flows are aggregated into classes. A flow consist of packet streams and each packet is associated with some task.

The problem domain addressed in this work is not specific to any one application but applies at large to the multi-server

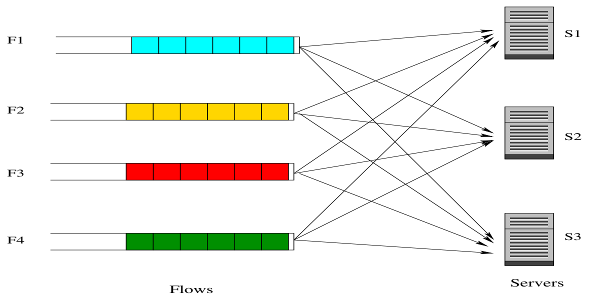

Fig. 1. Aggregated multi-server model with class-server *rate coupling*

context. One formulation applies to a scenario when arriving requests to a heterogeneous web-server cluster demand different service rates at different servers [1]. Another motivating example is a network processor system in which the different network processors are of different rates and make [2], [3] and the throughput of each processor depends on the application workload. The formulation can apply to as disparate contexts as service differentiation in a clustered networked environment [1], link striping [4] and even to the time-varying channel rates in the wireless scheduling context [5]. Yet, another application domain can be a transcoding cluster in which each server in the cluster provides the same transcoding function, but, at a rate that depends on its configuration parameters and the level of transcoding required [6], [7].

In such multi-server systems, an issue of overriding importance is the *fair sharing* of the aggregated server resource amongst the contending flows. Fairness ensures immunity from misbehaving flows and allows for better congestion control and rate-adaptive applications. Stringent QoS requirements also lead to strict delay bounds and better throughput by selective admission control. We consider such "fairness" issues in the aggregated multi-server context. We generalize the problem formulation by incorporating the requirement that the rate of service depends on both the class to which the flow belongs and the server on which it is scheduled i.e. there exists a *flow-server rate-coupling*. Figure 1 shows such a system with 3 different servers and 4 flow-classes and the service is *flow-server* dependent.

The system model under consideration is very general. Addressing the "fairness" provisioning problem this way implicitly takes into consideration the *heterogeneity* of the servers and *flow-server affinity* in the aggregated system. Our focus is to study these type of fair resource allocation problems from

Satya R. Mohanty and Laxmi N. Bhuyan are with the Department of Computer Science and Engineering, University of California at Riverside, Engineering BU2, Riverside, CA 92521. This research was supported in part by NSF grant CCR-0311437.

a mathematical perspective and model them analytically.

The formulation requires that flows be associated with a quality of service (QoS) parameter. Our optimization objectives are manifold. We aim to simultaneously (i) provide differentiated quality of service to backlogged flows, (ii) ensure high server utilization and (iii) achieve high throughput. To the best of our knowledge such a formulation has not been addressed elsewhere in the literature.

Mathematically, we abstract the problem into two distinct components. Using linear programming on an idealized fluid model, we first determine the fractional volumes of flow-classes (fluid) that need to be allocated to different servers. Next, once these volume fractions are determined, actual scheduling of tasks is based on a direct application of the Smoothed Round Robin [8] (SRR) algorithm (a fair round-robin algorithm proposed for single link systems). We give this two-component algorithm a name: *Linear Programming based scheduling algorithm for multi-servers* (LP-M). Algorithms like SRR and Deficit Round Robin (DRR) are meant for single-link systems. We extend them in an obvious way to the problem domain in the multi-server setting. We call the corresponding extensions SRR-M and DRR-M respectively.

We perform extensive simulations with all three algorithms LP-M, SRR-M and DRR-M using the same packet trace and pre-specified server-flow rates for an example prototype. The simulation results reveal that these obvious extensions of single link round-robin algorithms to solve the multi-server multi-class with differentiated service problem give sub-optimal performance. Using LP-M on the other hand, the *throughput* and *utilization* of the system can be *substantially increased* while ensuring the *inter-class fairness* requirement. In short all our objectives are satisfied. In addition the "intra-class inter-flow" reordering is considerably reduced.

A. Our Contributions

We develop a fair scheduling algorithm for the multi-server *flow-server rate coupled* problem domain using a linear programming approach. The framework consists of solving an on-line optimization problem to generate *desired* service rates that give optimal performance in the fluid model. We propose a round-robin based packeted scheduler that takes these rates as inputs and then schedules packets. Thus we decouple the problem into two parts and solve the first sub-problem. The output of the first sub-problem serves as input to the second sub-problem but the algorithm in the second sub-problem is itself independent of the first sub-problem. We verify the proposed algorithm with simulation results.

The paper is organized in the following way: Section II describes the related work. In section III (only for the purpose of analysis and without any loss of generality) we consider the case when the set of competing flow-classes is invariant i.e. the flows are always backlogged. We consider two cases (i) the aggregate bandwidth is shared proportionally according to the class' reservations and (ii) each class states its need using two independent parameters, a minimum rate to be guaranteed (floor rate) and the maximum rate it could use of (target). For each of these cases we determine the rates at which the servers should serve the flow-classes. In section IV we present an algorithm for scheduling these flow-classes (based on a round-robin algorithm that takes into account the computed rates in section III). The simulation results along with their interpretations are provided in section V. Section VI concludes the paper.

II. Related Work

A whole body of recent research in Fair Queuing [9], [8], [10], [11], [12], [13] has addressed the issue of scheduling competing flows fairly over an *individual* link in accordance with their reserved rates of service. QoS scheduling mechanisms for the single link do not trivially extend to the multi link case as they do not consider problematic out-of-order packet delivery issues especially when the links operate at different rates (an application adversely affected is TCP Blanton and Allman [14]). Another issue is maintaining packet order may cause considerably low link utilization and throughput.

Link striping algorithms that achieve load balance of multiple links in the presence of variable length packets and guarantee in-order packet delivery at the receiver application were proposed by Adiseshu *et al.* [4]. Cao *et al.* [15] discuss performance of load balancing schemes for multiple links by different hashing methods. Jo *et al.* [16] introduce a dynamic hashing with flow volume algorithm that achieves load balancing and also reduces reordering. Shi *et al.* [17] propose a load balancing scheme that exploits the bursty nature of Internet flows and present a scheme that achieves load balancing, reduces reordering and also attempts to preserve temporal locality by mapping flows to particular forwarding engines for high system utilization. However, no QoS is discussed.

The problem of provisioning QoS among competing flows over a system of links was addressed by Blanquer and Özden [18]. Their algorithm (MSFQ) is based on the Generalized Processor Sharing system (GPS) [12]: an idealized service discipline. Xiao *et al.* [19] provide an analysis of multi-server round robin scheduling disciplines. Cobb and Lin [20] propose a general technique of constructing a multi-link scheduling algorithm from a single link one. They also consider several sorting techniques to avoid packet reordering. However the methods that they propose entail sorting algorithms at the end hop to correct the packet reordering. The work in [21] address the same problem and provide scheduling algorithms based on dynamic partitioning and flow mappings; however the schemes cannot provide guarantees on hard deadlines. Zhu *et al.* [1] describe a demand driven service differentiation scheme in a clustered network server environment that operates via a dynamic scheduling scheme aware of varying request resource demands and periodic server partitioning. The system is modeled as a multi-class open queuing network, and the performance metric is the stretch factor (determined through an unconstrained minimization problem). Fariza *et al.* [2] consider an interesting fairness problem in allocation of multiple resources in a network processor. In all of the above works there is no notion of the rate of the server as a *function* of the flow, i.e. there is no *flow-server rate-coupling*. In the wireless domain a somewhat similar work is by Liu *et al.* [5]. They consider a multi-user scheduling problem in which the goal is to maximize total system throughput and ensure long-term deterministic and probabilistic fairness. In this work the air channel rate varies with time and the algorithm consists of an optimization problem and a control-update problem. Again an off-line problem in heterogeneous task processing using the generalized stable marriage technique has been discussed in [22].

III. Linear Programming Formulation

Assumptions:

We only make a few simplifying assumptions in this work. As can be seen these assumptions are rather mild. We list these as follows.

- The rate of service of all flows belonging to a class on any one server is constant.
- The processing time of a packet depends on its length. This means that shorter length packets take smaller processing time than longer packets of the same flow on the same server.
- The number of flow-classes are few compared to the total number of flows which can be in the millions.

The last assumption is reasonable with the standardization efforts taken by the Internet Engineering Task Force (IETF) on Differentiated Services [23]. The Differentiated Services framework relies on a *small* number of service levels, or *Per Hop Behaviors* (PHBs), that each specify how a routers should treat the corresponding packet.

Our starting point is the fluid model. It is helpful to keep in mind that a flow-class may consist of many flows. Accordingly we treat each flow-class as fluid which can be served at any continuous rate. We formulate the problem mathematically in the form of a linear program (LP) [24]. Let n be the number of flow-classes and m the number of servers. Let ϕ_i be the reservation or weight associated with flow-class i. Let r_{ij} ($r_{ij} \in \mathcal{R}^+$) be the rate at which flow-class i is processed by server S_j. Let x_{ij} be the time-fraction at which flow-class i is serviced by server S_j. These rates can be arranged in the form of a matrix and denoted by R and X, i.e.

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{bmatrix}$$

and

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}.$$

Let Γ be the instantaneous throughput of the overall system. The goal is to maximize Γ while ensuring fairness to each flow-class.

A. Elastic applications

In the first formulation we target fairness for elastic traffic in which available bandwidth is shared according to the weights or reservations.

The problem formulation takes the following optimization form:

$$\text{Maximize } \Gamma = \Sigma_{i \in n, j \in m} x_{ij} r_{ij} \tag{1}$$

subject to

(i) Work Conservation

We only deal with work-conserving systems in which the servers are never idle. This means that fractions of flow-classes incident on any individual server add up to 1.

$$\begin{aligned} \Sigma_i x_{ij} &= 1 \ , \forall j \\ 0 \leq x_{ij} &\leq 1 \ , \forall j \end{aligned} \tag{2}$$

(ii) Fairness

The rate of service received by flow-class i in the fluid model in the time interval (t, τ) is denoted by $W_i(t, \tau)$ where $W_i(t, \tau) = (\Sigma_j x_{ij} r_{ij})(\tau - t)$. Fairness requires that $\frac{W_i(t,\tau)}{\phi_i} = \frac{W_j(t,\tau)}{\phi_j}$. Accordingly we have the equation

$$\frac{\Sigma_k x_{ik} r_{ik}}{\phi_i} = \frac{\Sigma_k x_{jk} r_{jk}}{\phi_j} \ \forall i, j \ i \neq j \tag{3}$$

Example: Assume that the normalized server rate for each flow-class in the system shown in Figure 1 is given by

$$R = \begin{bmatrix} 0.8 & 0.6 & 0.4 \\ 0.6 & 0.5 & 0.8 \\ 0.5 & 0.9 & 0.6 \\ 0.8 & 0.6 & 0.5 \end{bmatrix} \tag{4}$$

and the vector of weights associated with the flow-classes $[\phi_1 \ \ \phi_2 \ \ \phi_3 \ \ \phi_4] = [40 \ \ 25 \ \ 20 \ \ 15]$.

Solving the LP posed by equation (1) under the constraints of equations (2) and (3) via the *Matlab linprog* function gives

$$X = \begin{bmatrix} 0.8129 & 0.4248 & 0.0 \\ 0.0 & 0.0 & 0.7072 \\ 0.0 & 0.5029 & 0.0 \\ 0.1871 & 0.0723 & 0.2928 \end{bmatrix}$$

with a throughput equal to $\Gamma = 2.263$ as against a maximum feasible throughput of $\Gamma = 2.5$ (which will however not be fair according to the definition). The service rate of flow-class 1 is $0.8129 * 0.8 + 0.4248 * 0.6$ i.e. $\frac{\partial}{\partial t} W_1(t) = 0.9052$ and similarly for flow-class $2 \ldots 4$. The rates are as shown below

$$\begin{bmatrix} \frac{\partial}{\partial t} W_1(t) \\ \frac{\partial}{\partial t} W_2(t) \\ \frac{\partial}{\partial t} W_3(t) \\ \frac{\partial}{\partial t} W_4(t) \end{bmatrix} = \begin{bmatrix} 0.9052 \\ 0.56576 \\ 0.45261 \\ 0.33946 \end{bmatrix}$$

As can be verified the service rate ratios are almost exactly $40 : 25 : 20 : 15$, as desired.

B. Adaptive applications

In the case of adaptive flows, we do not have a reservation as such but we consider that each flow-class i specifies its demand by two independent parameters, (i) a minimum rate that needs to be guaranteed, denoted by f_i and referred to as the floor rate and (ii) a maximum rate that it could use of and denoted by t_i (referred to as the target rate) (See [25] and references therein). We require the scheduler ensures that

$$\frac{W_i(t,\tau) - (t-\tau)f_i}{W_j(t,\tau) - (t-\tau)f_j} = \frac{t_i}{t_j} \tag{5}$$

Example: Suppose that the set of flow-classes and servers is the same as in the example of section III-A. However instead of a unique ϕ (reservation), each flow-class has a floor and target allocation as shown in the table I.

TABLE I
TABLE FOR FLOOR AND TARGET RATES

Flow-class	floor rate	target rate
1	0.6	1.0
2	0.4	0.8
3	0.4	0.7
4	0.3	0.6

The solution to the new LP taking into account the fairness requirement in equation (5) is

$$X = \begin{bmatrix} 0.7470 & 0.3178 & 0.0 \\ 0.0 & 0.0 & 0.6883 \\ 0.0 & 0.5909 & 0.0 \\ 0.2530 & 0.0913 & 0.3117 \end{bmatrix}$$

and the rate of service given by

$$\begin{bmatrix} \frac{\partial}{\partial t}W_1(t) \\ \frac{\partial}{\partial t}W_2(t) \\ \frac{\partial}{\partial t}W_3(t) \\ \frac{\partial}{\partial t}W_4(t) \end{bmatrix} = \begin{bmatrix} 0.7883 \\ 0.5506 \\ 0.5318 \\ 0.4310 \end{bmatrix}$$

As can be seen once again, the computed rates are higher than the floor rates and the for any two flow-classes i and j equation 5 is always satisfied i.e.

$$\frac{\frac{\partial}{\partial t}W_i(t) - fi}{\frac{\partial}{\partial t}W_j(t) - f_j} = \frac{t_i}{t_j} \quad \forall i,j$$

C. Existence of Solutions to the Linear Program

We state without proof that the solution to the linear program formulation in section III-A always exists. This may not hold necessarily for the second formulation (i.e. the adaptive case). However the feasibility of the linear problem in section III-B can be characterized by checking for the *consistency* of the dual. Refer for example Chvátal [24], (Chapter 9 for details).

This section presented LP based algorithmic techniques to determine *desirable rates* at which backlogged individual classes of traffic need to be serviced in the fluid model (an idealized abstraction). In the next section we provide a round-robin based algorithm that utilizes these *desirable rates* to optimally schedule flow packets.

IV. ROUND ROBIN SCHEDULING ALGORITHMS

The unique feature of the *flow-server rate-coupling* is that the rate at which the ideal fluid server operates varies with the backlogged flow-classes. It is not clear how to design a time-stamp based packeted scheduler that tracks the fluid model closely within a bounded approximation. We postpone the exploration and applicability of such designs for future research. For the present paper we adopt round-robin based schemes which are inherently short-time unfair (cannot guarantee the GPS fairness) but are easer to implement. Recent works in fair scheduling of contending flows over a single link (based on the round robin based approach) include Group Ratio Round-Robin [26], Stratified Round Robin [27], Fair Round Robin [28]. Though any of these algorithms could be adopted in principle, for our purpose, we choose to employ two other well-known round robin schemes (i) the older deficit round robin (DRR) and (ii) the newer Smoothed Round Robin (SRR) for scheduling packets in the present setting. These algorithms work in conjunction with the rate parameters determined in the previous section. Thus we come up with an algorithm that is sensitive to a class's reservations and aware of its service rate on any server.

In DRR a flow is continuously served for a pre-specified amount of time given by the flow's "quantum". This results in a burst scheduling for each flow. In SRR on the other hand, the burst is "smoothed" by employing a scheduling technique that uses a weighted spread spectrum (WSS) in conjunction with an associated weight matrix (WM). Suppose there are 3 flows f_1, f_2, f_3 with rates $r_1 = 64\ Kbps,\ r_2 = 128\ Kbps, r_3 = 256\ Kbps$, competing for a single link of capacity $512\ Kbps$. The normalized weights of the flows are $w_1 = 1, w_2 = 2, w_3 = 4$. Then the WM is given by

$$WM = \begin{bmatrix} WV_1 \\ WV_2 \\ WV_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

and corresponding WSS is $1, 2, 1, 3, 1, 2, 1$. The flow service sequence will be given by

$$f_3, f_2, f_3, f_1, f_3, f_2, f_3.$$

We extend DRR and SRR scheduling algorithms to the multi-class multi-server fair queueing case in a natural and straightforward way. Instead of flows we now operate at the granularity of flow-class. As usual each flow-class has an associated quantum (number of bytes served) and the flows are served in rounds (interleaved, in case of SRR). During a round whenever a server becomes free, the algorithm first checks if the head-of-line packet of the flow-class (to which the round belongs) can be accommodated in the remaining portion of the quantum. Otherwise the next flow-class that gets to "own" the round is eligible and can then send its packets to the multi-server system. We refer to these algorithms as DRR-M and SRR-M. These algorithms are *oblivious* to *flow-server rate coupling*.

To employ SRR effectively in the *flow-server rate coupled* model, we have to determine what flow-classes are allowed to be scheduled on any server. This is fairly easy. For any server j we need to consider only those flow-classes i for which x_{ij} in the solution of the LP in section III is non-zero. The adaptive case is exactly similar. Henceforth it must be understood that flow i will be scheduled for a fraction of time x_{ij} on server j whenever x_{ij} is non-zero.

The scheduler also maintains a weight matrix of flows for each server. How to determine the weights of these flows in the matrix? Notice that the amount of work done per unit time by server j on two flows i_1 and i_2 is in the ratio of $x_{i_1j}r_{i_1j} : x_{i_2j}r_{i_2j}$. Hence the weight of the flows on each server can be generated from the *entry-wise product* (*Hadamard product* [29]) of the two $m \times n$ matrices X and R, denoted by $(X \bullet R)$ i.e. the $m \times n$ matrix given by $(X \bullet R)_{ij} = x_{ij}b_{ij}$ and suitably multiplying by an appropriate power of 10 so that they are all integers. The weights in the weight matrix for an individual server are listed in the corresponding column of $(X \bullet R)$. For instance, referring to the example in the elastic case (section III-A), the *entry-wise product* (after an entry-wide multiplication by 100000) is

$$(X \bullet R) = \begin{bmatrix} 65032 & 25488 & 0 \\ 0 & 0 & 56576 \\ 0 & 45261 & 0 \\ 14968 & 4338 & 14640 \end{bmatrix} \tag{6}$$

and the WM for server 0 is

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Similarly in the example for the adaptive case (section III-B) $(X \bullet R)$ is

$$(X \bullet R) = \begin{bmatrix} 59760 & 19068 & 0 \\ 0 & 0 & 55064 \\ 0 & 53181 & 0 \\ 20240 & 5478 & 15585 \end{bmatrix} \tag{7}$$

and the WM's can be computed accordingly. Since the weights are now determined, the algorithm is fairly direct. We name our LP and SRR based algorithm *Linear Programming based scheduling algorithm for multi-servers* (LP-M), presented formally as Algorithm 1 (only scheduling routine shown). Flows are arranged in a doubly linked list. Refer to [8] for details.

Any fair scheduling algorithm in the literature for single link sharing suffices for servicing individual flows within a flow-class. For simplicity, we adopt DRR as the preferred scheduling algorithm at this granularity. Thus ours is a hierarchical scheduler. It employs LP-M at the class level and DRR at the flow level.

Algorithm 1 LP based SRR scheduling, LP-M

Input: Given: (i) set of class-server specific rates (ii) flow-class reservations Φ_i's.

Output: Linear Programming and SRR based fair scheduling

```
1:  $P_{dl}$: Pointer to doubly linked list; $def_f$: deficit of flow $f$; $L_{max}$: Maximum Length Packet
2:  Compute server specific weight matrix for server $j$, i.e. $\mathsf{WM}_j$, and associated spectrum $\mathsf{WSS}_j$ $\forall s_j$
3:  for all Servers $s_j's$ in the system do
4:      $X := 0,\ \Psi = \Phi$
5:      while busy do
6:          $P_{dl} \rightarrow f_{id}$
7:          $def_f + = L_{max}$
8:          while $def_f > 0$ do
9:              if $(L_f \leq def_f)$ then
10:                 $dequeue(P_f)$;
11:                 $send(P_f)$;
12:                 $def_f - = L_f$;
13:             else
14:                 break;
15:             end if
16:         end while
17:     end while
18:     if $(P_{dl} \rightarrow next! = tail_{col})$ then
19:         $P_{dl} = P_{dl} \rightarrow next$
20:     end if
21: end for
```

V. Simulations

A. Packet reordering

Quantifying packet reordering is not trivial, as different protocols may be able to tolerate sequences with quite different reorder degree and type. Consider for example sequences (2, 3, 4, 5, 6, 7, 8, 1) and (2, 1, 4, 3, 6, 5, 8, 7). In the first one, only one packet is "out-of-order". However, if an application needs to read the packets in the original order to process the whole sequence, the second sequence seems to have a lower degree of reordering (only two consecutive numbers need to be read to restore the original sequence). We adopt the definitions of packet reordering as described by Marton *et al.* [30] (An alternative framework is described in [31]). The reordering definitions and metrics therein have relevance to receiver design that can impact TCP and Real-time application performance. In the following let s_i be the original sequence number of i-th packet received by the destination. We maintain a variable `NextExp`, which is equal to the highest sequence number of the packets received so far plus 1. A packet is received *in order* if its sequence number is greater than or equal to `NextExp`. Otherwise, a packet is reordered. We only report the two most important out of several metrics.

Reordered ratio (RR): Reordered ratio is simply the ratio of packets that were received out of order to the total number of packets.

Reordering extent (RE): The Reordering extent of a reordered packet with sequence number s_i is defined to be $i - j$ for the smallest value of j such that $s_j > s_i$. The first arriving packet is always in-order by definition, and has undefined reordering extent (as all packet which are received in order). The reordering extent of a flow is the maximum over the reordering extents of all the reordered packets. The reordering extent is a rough estimate on the size of the buffer which is required to store the out of order packets in order to restore the original sequence order.

Since we focus on work-conserving algorithms, the packet reordering cannot be completely eliminated. This is easy to see by considering two packets, one with length 1 and the other with length $\frac{1}{2}$ arriving at time 0, both belonging to the same flow. If only two servers of the same rate are available, then any work conserving algorithm must schedule these packets at time 0. They will, therefore, always leave the system reordered.

B. Simulation Experiments

Simulation settings:

We simulated the multi-class multi-server system using the parameters in the example of section III-A. The class-specific rates are determined by multiplying every entry in the rate-matrix of equation 4 by 1000000. Thus R_{11} is $0.8MB$, R_{12} is $0.6MB$ and so on. Flows were grouped into four classes; the assigned reservations of these classes were in the ratio $40 : 25 : 20 : 15$. Simulations were performed on a synthetic trace consisting of 10 million packets. We considered a completely backlogged system in which packet sizes were uniformly distributed between 40 and 1500 bytes respectively. The decision

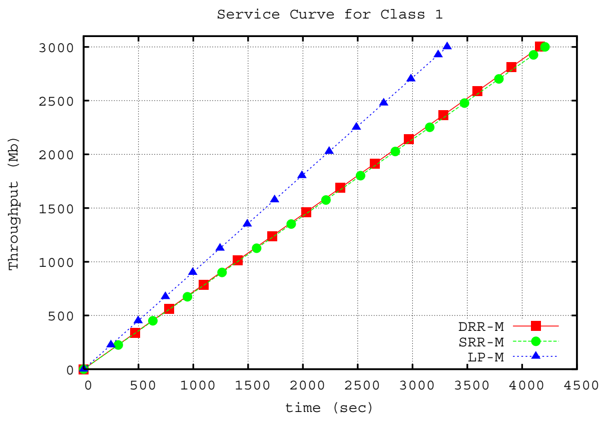

Fig. 2. Throughput of class 1

to generate synthetic traces instead of using the traces from NLANR [32] should not detract from the expected results. In the *flow-server rate coupled* model a service rate is associated with each flow and server. Such information is however not present in the real traces available on-line. However Internet traces are bursty and exhibit strong temporal locality. For a meaningful resemblance with realistic Internet applications, flow-packets with consecutively increasing sequence numbers are contiguously assigned in the trace in "bursts". The range of this "burst" is taken from a uniform distribution between $[0, \textit{BURSTSIZE}]$, where *BURSTSIZE* is a simulation parameter. Generating packet lengths from the uniform distribution leads to more reordering and hence easier evaluation by all three algorithms.

Simulation results:

Simulations were carried out on the same trace as input to the multi-server system employing each of the three different algorithms in turn DRR-M,SRR-M and LP-M with a flow-size of 1000 per class and $\textit{BURSTSIZE} = 16$. The performance is evaluated through service figures and relevant reordering metrics. Figures 2 and 3 show the service performance for

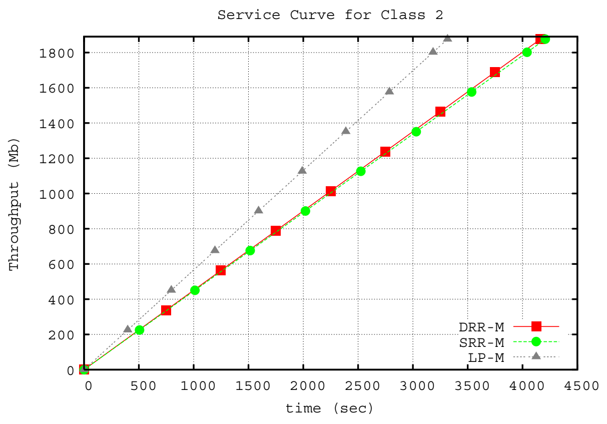

Fig. 3. Throughput of class 2

class I and class II respectively while Figures 5 and 6 show the service performance for class III and class IV respectively. As can be seen the performance of LP-M is much superior to that of SRR-M and DRR-M. Also DRR-M and SRR-M perform very similarly. This can be seen from the respective service curves which are very closely spaced. However from a closer inspection it is seen that SRR-M is less bursty in general compared to DRR-M (agreeing with theory). This is evident from Figure 4 which focuses on a short time window. The

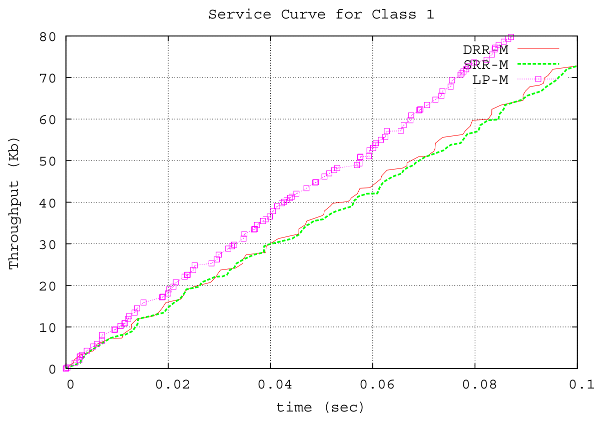

Fig. 4. Throughput of class 1 (small window)

degree of the burst will depend in general on the difference in flow-server specific service rates.

The simulation results as illustrated in the service curves establish that using LP-M, the throughput increases to almost 24.99% for class I, 25.01% for class II, 25.57% for class III and 24.95% for class IV over that obtained by SRR-M or DRR-M (i.e. almost by a fourth more, with a confidence interval of 99.9%). The throughput curves for the system can be deciphered from the service curves under various algorithms by simply adding the latter for various classes. We note that the overall throughput increases substantially

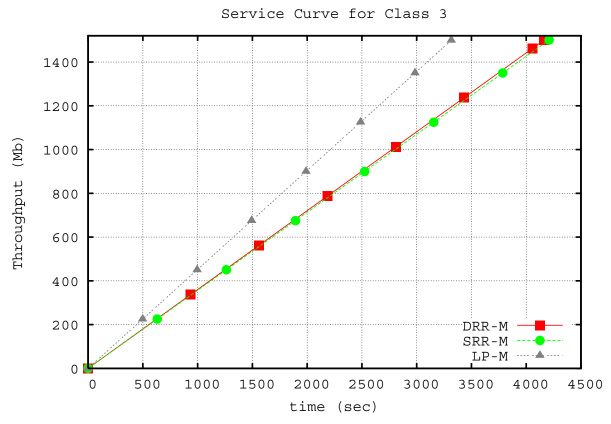

Fig. 5. Throughput of class 3

when LP-M is employed. The simulations are performed for a completely backlogged system but can easily be extended to the on-line case. Each time a class becomes backlogged or vanishes the new service rates can easily be computed. With moderately high number of classes, the complexity is

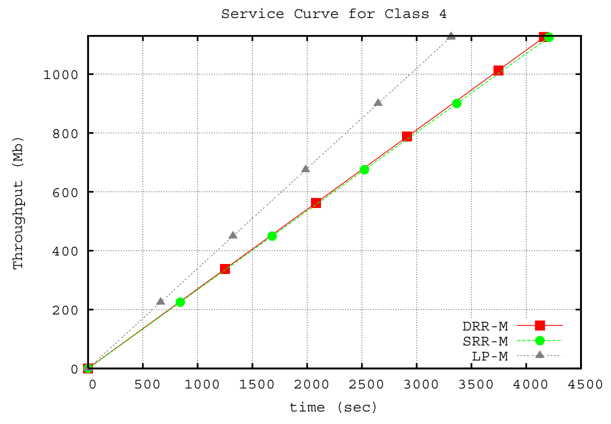

Fig. 6. Throughput of class 4

almost polynomial is time (the revised simplex method [24]). As mentioned earlier the Differentiated Services standards [23] specify only a few service classes, reinforcing this point.

The reordering information is provided with the relevant reordering metrics (i) the reordering extent and (ii) the reordered ratio. We performed simulations with different *BURSTSIZE* and different number of flows within a class. Results are reported only for the case when *BURSTSIZE* $=$ 16 and the number of flows in a flow-class equals 100 and 1000 respectively.

The reordered-ratio of the top twenty flows with the highest reordered ratio is also shown in Figure 7 and 8. LP-M results in the lowest RR, almost half that of SRR-M or DRR-M.

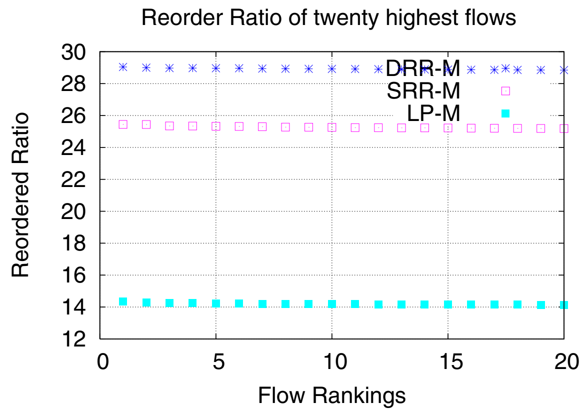

Fig. 7. Reorder Ratio (Flow-Size=100)

SRR-M performs better than DRR-M. This is to be expected since SRR-M is less bursty than DRR-M (a higher probability exists under DRR-M of same-flow packets being concurrently serviced by multiple servers). It may be noticed that the degree of reordering actually increases (by a small percentage) with the increase in the number of flows when the same packet trace is classified into many flows. This is explained by the fact given the total number of packets stays invariant, the number of packets per flow decreases with an increase in the number of flows. Therefore, even though the absolute number of packets that undergo reordering decreases, the RR is ,therefore, likely to increase. The simulations also show that the classes with higher reservation have higher reordered ratio rates. Table II shows the RE of flows of the different classes

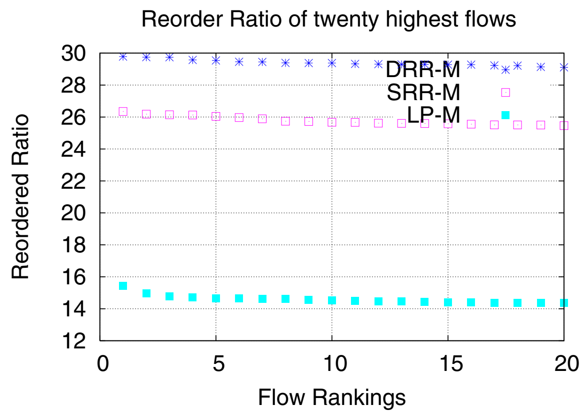

Fig. 8. Reorder Ratio (Flow-Size=1000)

(flow-size=1000) under each of the three algorithms. All flows under DRR-M and SRR-M have an RE of 2 whereas all class 1 flows under LP-M have an RE equalling 1. Class 2 and 3 flows do not undergo any reordering since they are scheduled on one server. Under LP-M, some 55 flows in class 4 have an RE of 1 whereas the rest 945 have an extent equal to 2. These numbers (although by no means exhaustive), nevertheless, are suggestive of improvement in reordering performance.

TABLE II

THE NUMBER OF FLOWS WITH A GIVEN REORDER EXTENT (RE)

Class	Reorder Extent	DRR-M	SRR-M	LP-M
1	1	0	0	1000
	2	1000	1000	0
2	1	0	0	0
	2	1000	1000	0
3	1	0	0	0
	2	1000	1000	0
4	1	0	0	55
	2	1000	1000	945

VI. Conclusions and Future Work

We present the LP-M algorithm for the QoS provisioning problem in multi-server systems with *flow-server rate coupling*. LP-M aims at increased throughput for each class, fair sharing of bandwidth across flow-classes and optimum utilization of servers. Simulation results show that LP-M provides better throughput and service differentiation to flows compared to natural extensions of packeted round-robin algorithms (for single server systems) to the multiple server system. Evaluating reordering quantitatively through relevant metrics indicate that intra-flow reordering is also minimized.

One may suspect, that when the number of flows is increased (keeping the packet count of each flow invariant) the reordering decreases for all algorithms, as the probability that the same flow is serviced concurrently decreases. It is also very likely that when servers are of similar rates the performance of all the three algorithms is similar. Ours is an on-line work-conserving algorithm. The only assumption is that the number of flow-classes is not very large and the *flow-server rate-coupling* is quantified apriori. The first assumption is met by most current QoS classification schemes like DiffServ. The second assumption may be satisfied in case of a known workload. Future work can focus on hard guarantees, other effective extension of single server scheduling algorithms to the multi-server case, and extensive evaluation of reordering via the proposed metrics [30], [31]. The overall improvement suggests that LP-M appears to be potentially useful in the context of a multi-server setting like the web-server, multi-media transcoding and applications, and other task processing systems where *flow-server rate coupling* exists.

References

[1] Huican Zhu, Hong Tang, and Tao Yang. Demand-driven service differentiation in cluster-based network servers. *IEEE INFOCOM*, March 2001.

[2] Fariza Sabrina, Salil S. Kanhere, and Sanjay Jha. Implementation and performance analysis of a packet scheduler on a programmable network processor. *IEEE Conference on Local Computer Networks (LCN'05), pages 242-249*, November 2005.

[3] Jiani Guo, Jingnan Yao, and Laxmi Bhuyan. An efficient packet scheduling algorithm in network processors. In *IEEE INFOCOM '05*, 2005.

[4] Hari Adiseshu, Guru Parulkar, and George Varghese. A reliable and scalable striping protocol. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 131–141. ACM Press, 1996.

[5] Yonghe Liu and Edward Knightly. Opportunistic fair scheduling over multiple wireless channels. *IEEE INFOCOM, San Francisco*, April 2003.

[6] Surendar Chandra, Carla Schlatter Ellis, and Amin Vahdat. Differentiated multimedia web services using quality aware transcoding. *IEEE INFOCOM*, March 2000.

[7] Jiani Guo and Laxmi Bhuyan. Load balancing in a cluster-based web server for multimedia applications. *accepted by IEEE Transactions on Parallel and Distributed Systems*, 2006.

[8] Guo Chuanxiong. SRR: an O(1) time complexity packet scheduler for flows in multi-service packet networks. *Proceedings of the ACM SIGCOMM,*, August 2001.

[9] J.C.R Bennet and H. Zhang. WF^2Q: Worst-case fair weighted fair queueing. *Proceedings of the IEEE INFOCOM,*, March 1996.

[10] P. Goyal, H. Vin, and H. Chen. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. *Proceedings of the ACM SIGCOMM,*, August 1996.

[11] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1(1), 1990.

[12] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, June 1993.

[13] M. Shreedhar and George Varghese. Efficient fair queueing using deficit round robin. *Proceedings of the ACM SIGCOMM,*, August 1995.

[14] E. Blanton and M. Allman. On making TCP more robust to packet reordering. *ACM Computer Communication Review*, 32(1), January 2002.

[15] Zhiruo Cao, Zheng Wnag, and Ellen Zegura. Performance of hashing-based schemes for internet load balancing. In *IEEE INFOCOM '00*, 2000.

[16] Ju-Yeon Jo, Yoohwan Kim, H. Jonathan Chao, and Frank Merat. Internet traffic load balancing using dynamic hashing with flow volume. In *SPIE ITCom '02*, 2002.

[17] Weiguang Shi, Mike H. MacGregor, and Pawel Gburzynski. A scalable load balancer for forwarding internet traffic: Exploiting flow-level burstiness. In *ANCS '05*, 2005.

[18] Josep M. Blanquer and Banu Özden. Fair queueing for aggregated multiple links. *Proceedings of the ACM SIGCOMM*, October 2001.

[19] Haiming Xiao and Yuming Jiang. Analysis of multi-server round robin scheduling disciplines. *IEICE Transactions*, January 2002.

[20] Jorge A. Cobb and Miaohua Lin. A theory of multi-channel schedulers for quality of service. *J. High Speed Netw.*, 12(1,2):61–86, 2003.

[21] Satya R. Mohanty and Laxmi N. Bhuyan. On fair scheduling in heterogeneous link-aggregated services. In *ICCCN '05*, 2005.

[22] Zhi Cheng Li. Solving static optimal matching problem in heterogeneous processingwith generalized stable marriage algorithms. In *Eighth International Parallel Processing Symposium*, pages 248–252, April 1994.

[23] S. Blake, D. Black, M. Carlson, E. Davies, Z. WAng, and W. Weiss. An architecture for differentiated services. Technical report, Request for Comments (Proposed Standard) RFC 2475, IETF, December 1998.

[24] Vašek Chvátal. Linear programming. *W. H. Freeman and Company*, 1983.

[25] Francois Toutain. Decoupled generalized processor sharing: A fair queueing principle for adaptive multimedia applications. *IEEE INFOCOM, pages 291-298, IEEE*, April 1998.

[26] Bogdan Caprita, Wong Chun Chan, Jason Nieh, Clifford Stein, and Haoqiang Zheng. Group ratio round-robin: O(1) proportional share scheduling for uniprocessor and multiprocessor systems. *USENIX 2005 Annual Technical Conference, pages 337-352*, April 2005.

[27] Sriram Ramabhadran and Joseph Pasquale. Stratified round robin: A low complexity packet scheduler with bandwidth fairness and bounded delay. *Proceedings of the ACM SIGCOMM,*, July 2003.

[28] Xin Yuan and Zhenhai Duan. FRR: a proportional and worst-case fair round robin scheduler. *IEEE INFOCOM*, March 2005.

[29] R. A. Horn and C. R. Johnson. Matrix analysis. *Cambridge Univ. Press*, 1999.

[30] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser. Packet reordering metric for IPPM. Technical report, IETF, August 2004.

[31] A. P. Jayasumana, N. M. Piratla, A. A. Bare, T. Banka, and R. Whitner. Reorder density and reorder buffer-occupancy density - metrics for packet reordering measurements. Technical report, IETF draft (work in progress), March 2006.

[32] Internet traces user community. `http://pma.nlanr.net/PMA/Traces/`.