CCIT Report # 470 February 2004

Performance Analysis of a Recursive Cyclic Scheduler for Class-based Scheduling

Raphael Rom, Moshe Sidi and Hwee Pink Tan
Department of Electrical Engineering
Technion, Israel Institute of Technology
Technion City, Haifa 32000, Israel
Email: {rom, moshe}@ee.technion.ac.il and hweepink@tx.technion.ac.il

Abstract

In this paper, we consider the problem of determining a cyclic (or loop) scheduler that allocates slots to flows as periodically as possible. We use the second moment of the inter-allocation distance for each flow as its periodicity metric. We establish the optimality of a Weighted-Round Robin with spreading (*WRR-sp_2*) scheduler for a two-flow scenario.

We consider a class-based scheduling scenario where flows are grouped according to their relative bandwidth demands. We propose a *C*-class scheduler that recursively performs inter-class scheduling using the corresponding *C*-1 class scheduler, prior to intra-class scheduling. Optimality is achieved for *C* = 2 with the *WRR-sp_2* as the inter-class scheduler.

Through numerical results, we show that the recursive scheduler achieves the best periodicity performance at the expense of intra-class fairness, which is desirable for class-based scheduling. Thereby, we expose a trade-off between periodicity and fairness performance in the design of loop schedulers.

I. Introduction

Consider a system that comprises an indivisible resource (time-slot) and *n* clients (or flows) share it by means of time multiplexing: in any given time, a different flow may use the resource. Many applications require that flows are served at some prescribed rate, and this rate should be as smooth as possible even in small time windows. The allocation of time slots to flows is governed by a scheduling algorithm. In other words, given a set of requested shares, $\{x^{(i)}\}_{i=1}^{n}$, the goal of the scheduling algorithm is to produce an assignment of time slots (or a schedule) to flows, while trying to optimize two different measures:

(a) **Fairness**: a schedule is said to have good fairness if the fraction of time slots allocated to each flow is close to its requested share;

(b) **Smoothness**: a schedule is said to have good smoothness if the time slots allocated to each flow are as evenly spaced as possible.

The best possible schedule is one where the allocated shares are exactly the requested shares (perfect fairness) and where each flow is scheduled exactly every *p* time slots (perfectly-periodic schedule). Although schedules that offer fairness while neglecting smoothness are available [1], it is NP-complete to decide whether a set of requests admits a perfectly-periodic schedule [2].

A. Perfectly Periodic Scheduling

Two approaches to the scheduling problem are considered in the literature. The first approach insists on maintaining strict smoothness while relaxing the fairness requirement. In [3] (and references therein), each flow *i* requests that it be scheduled exactly every $\tau^{(i)}$ time slots, and the goal is to determine a scheduler that optimizes the fairness measure under the perfect periodicity constraint. Strict smoothness requirements imply that the periods allocated to some flows will not match their requests. A suitable metric to measure the deviation from perfect fairness for each flow is the fairness ratio, given by the ratio of its requested period and its granted period. There exist schedulers [4] that guarantee that the average fairness ratio (where the weight of each flow is its requested bandwidth) is close to optimal. The maximum measure is studied in [5], where the quality of the schedule is the worst-case fairness ratio over all flows.

B. Non-Periodic Scheduling

An alternative approach is to allow different gaps between consecutive allocations to a flow, while insisting on perfect fairness. This approach was considered in [6] (and references therein), where the authors considered an *online* variant of the resource sharing problem. Given that the arrival process of packets to each flow is independent and identically distributed (i.i.d), the goal is to determine a scheduler that optimizes some performance criteria under the perfect fairness constraint.

In [7], the author deduced that for throughput optimality for n=2 and unit buffer size per queue, the schedule must be *open-loop* (or de-centralized) and *conflict-free*. This work was extended in [8] to the case of n>2. It was also verified that an optimal schedule always exists and is stationary and *cyclic* (or loop), i.e., there exists an *N* such that for all *t*, the flow allocated to slot τ is also allocated to slot τ+*N*.