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

Raphael Rom[*]
Sun Microsystems Laboratories
Mountain View, CA 94043, USA
e-mail: rom@ee.technion.ac.il

Moshe Sidi and Hwee Pink Tan
Department of Electrical Engineering
Technion, Israel Institute of Technology
Technion City 32000, Israel
e-mail: 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.

Keywords: Loop Scheduler, Periodicity, Recursive, Class-based Scheduling

1 Introduction

Consider a system that comprises an indivisible resource (time-slot) and K 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, $\underline{x}=\{x^j\}_{j=1}^K$, 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:

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.

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 *j* is scheduled exactly every ζ^j time slots (perfectly-periodic schedule). In this study, we consider the design of perfectly-fair schedulers that maximize smoothness; the design of perfectly-periodic schedulers can be found in [1, 2, 3, 4, 5].

1.1 Related Work

In [6], the authors considered an *online* variant of the scheduling 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 *K*=2 and unit buffer size per input-flow, the schedule must be *open-loop* (or de-centralized) and *conflict-free*. This work was extended in [8] to the case of

[*]On leave from the Technion, Israel Institute of Technology.

K>2. It was also verified that an optimal schedule always exists and is stationary and *cyclic* (or loop), i.e., there exists an *R* such that for all *i*, the flow allocated to slot *i* is also allocated to slot *i*+*R*.

This reduces the problem to an *offline* one, where the objective is to determine a loop schedule of size *R*, given $\underline{x}$. The authors proposed a Golden Ratio Scheduler that achieves a nearly-optimal throughput under online conditions. In [6], the authors considered the case where the buffer size per input-flow is unlimited. It is shown that the mean queue size (or equivalently, the mean packet delay) is minimized with a perfectly-periodic schedule, which is not always feasible. Although the golden ratio scheduler is not perfectly periodic, it performs extremely well compared to lower bounds for expected packet delay.

1.2 Contribution of This Paper

Although loop schedulers that ensure perfect-fairness (denoted perfectly-fair loop schedulers) have been proposed in the literature, we are not aware of any work that analyzed and evaluated the extent of non-periodicity in these schedulers. In this paper, we propose a periodicity metric and compute the bounds for the metric over the class of perfectly-fair loop schedulers. By analyzing the periodicity properties of various loop schedulers, we propose a recursive loop scheduler for a class-based scheduling scenario. We demonstrate the gain in periodicity performance achieved by the recursive scheduler over known loop schedulers.

The paper is organized as follows: We define our scheduling problem, where we consider the design of perfectly-fair loop schedulers to maximize allocation smoothness while maintaining intra-class fairness in a class-based scheduling scenario, in Section 2. Then, we consider the requirements for optimal per-flow periodicity and establish the lower bound for the periodicity metric in Section 3. We describe the mechanism and periodicity properties of several loop schedulers in Section 4. In Section 5, we propose a recursive loop scheduler that achieves good periodicity performance in a class-based scheduling scenario. In Section 6, we compare the performance of various loop schedulers in terms of numerical results. We describe two applications of perfectly-fair loop schedulers in wireless networks in Section 7 and finally, some concluding remarks are given in Section 8.

2 Problem Definition

Given a set of requested shares, $\underline{x}$, we consider the design of perfectly-fair loop schedulers to maximize allocation smoothness while maintaining intra-class fairness in a class-based scheduling scenario. Let $\underline{y}$ denote a vector of elements *y*, where $y(i)$ is the i^{th} element of $\underline{y}$.

2.1 Perfectly-Fair Loop Schedulers

Let $\mathbf{F}^{\underline{x}}$ denote the class of *K*-flow perfectly-fair loop schedulers, i.e., schedulers that satisfy the perfect-fairness constraint over any interval of *R* slots, where *R* is the loop size. This implies that (for any finite *K*), the elements of $\underline{x}$ are rational. We can then define *R* and r^j, $1\leq j \leq K$, as follows:

$$\begin{aligned} R &= LCD(\underline{x}) \\ r^j &= x^j \cdot R \end{aligned}$$

where r^j is the number of slots allocated to flow *j* over any interval of *R* slots and LCD($\underline{x}$) is the lowest common divisor of the elements of $\underline{x}$.

If $n^j_\pi(m)$ denotes the duration between the $(m-1)^{th}$ and m^{th} allocation to flow *j* under a scheduler π, then we have the following property:

Property 1 *If $\pi \in \mathbf{F}^{\underline{x}}$, then for $1\leq j \leq K$,*

$$\begin{aligned} n^j_\pi(r^j+m) &= n^j_\pi(m) \\ \sum_{m=k}^{k+r^j-1} n^j_\pi(m) &= R, \text{ for any } k>0 \end{aligned}$$

Hence, each scheduler π can be uniquely characterized by the elements, $\{n^j_\pi(m)\}_{m=1}^{r^j}$ (which we denote by $\underline{n^j_\pi}$).

2.2 Class-based Scheduling Scenario

A class-based scheduling framework [9] is based on the paradigm of service classes, where flows with a common characteristic are grouped together. By considering a ‘class-aware’ scheduling paradigm that allocates slots to flows within each class *independently* of other classes (*intra*-class scheduling) and then *combines* the allocation vectors obtained in an optimal way (*inter*-class scheduling), we may achieve a gain in performance and reduction in complexity over ‘class-unaware’ schedulers. We propose a recursive implementation of such a scheduler in Section 5.

We consider a *C*-class scheduling scenario where each class *c* comprises κ^c flows whose indices are defined by $\mathbf{C}^c$, where

$$\mathbf{C}^c = \{\sum_{y=1}^{c-1}\kappa^y+1, \sum_{y=1}^{c-1}\kappa^y+2, \cdots, \sum_{y=1}^{c}\kappa^y\}$$

and $x^j = x^k \Leftrightarrow j,k \in \mathbf{C}^c$ for any $1\leq j,k \leq K$. We define the notation $\tilde{y}^c$, where $y \in \{x, r\}$, such that $\tilde{y}^c = y^j$ if $j \in \mathbf{C}^c$.

2.3 Performance Criteria

We define the following criteria to evaluate the performance of perfectly-fair loop schedulers in a class-based scheduling scenario:

2.3.1 Allocation Smoothness (Periodicity)

Per-flow periodicity : A suitable metric to evaluate the periodicity of allocation with respect to flow *j* is the variance of $\underline{n}_\pi^j$, $Var[\underline{n}_\pi^j]$ = $E[\underline{n}_\pi^j]^2$ - $(E[\underline{n}_\pi^j])^2$, where

$$E[\underline{n}_\pi^j]^x = \frac{\sum_{m=1}^{r^j}[n_\pi^j(m)]^x}{r^j}$$

However, according to Property 1, we have the following:

$$\begin{aligned} E[\underline{n}_\pi^j] &= \frac{\sum_{m=1}^{r^j} n_\pi^j(m)}{r^j} \\ &= \frac{R}{r^j} \quad \text{independent of } \pi \end{aligned}$$

Hence, the periodicity measure for any scheduler π with respect to flow *j* can be evaluated in terms of $E[\underline{n}_\pi^j]^2$: a smaller value of $E[\underline{n}_\pi^j]^2$ implies a more periodic slot allocation to flow *j* and vice versa. We note that the order of the elements in $\underline{n}_\pi^j$ is unimportant for the evaluation of $E[\underline{n}_\pi^j]^2$, and hence, we can consider $\underline{n}_\pi^j$ as a set of r^j elements.

Ensemble periodicity : We define the weighted covariance of $\{\underline{n}_\pi^j\}_{j=1}^K$, $wcov_\pi$, as follows:

$$wcov_\pi = \sum_{j=1}^{K} \frac{r^j}{R} \cdot \frac{E[\underline{n}_\pi^j]^2 - (E[\underline{n}_\pi^j])^2}{(E[\underline{n}_\pi^j])^2}$$

The metric, $wcov_\pi$, reflects the periodicity over the ensemble of all flows. With perfect periodicity, $wcov_\pi$=0 since $\underline{n}_\pi$ = $E[\underline{n}_\pi^j]$. Hence, a value close to zero is an indication of good periodicity performance over the ensemble of flows.

2.3.2 Intra-class Fairness

A desirable characteristic in class-based scheduling is the notion of intra-class fairness, i.e., all flows from the same class should possess the same periodicity characteristics.

2.4 Problem Formulation

If $\underline{a}_\pi$ denotes the allocation vector (of length *R*) according to scheduler π, then our scheduling problem can be formulated as follows:

K-flow Loop Scheduling Problem

Determine the allocation vector $\underline{a}_{\pi^*}$ such that for 1$\leq j \leq K$

$$E[\underline{n}_{\pi^*}^j]^2 = \min_{\pi \in \mathbf{F}^{\underline{x}}} E[\underline{n}_\pi^j]^2$$

and $\forall$ *j*,*k* $\in \mathbf{C}^c$ for 1$\leq c \leq C$

$$E[\underline{n}_{\pi^*}^j]^2 = E[\underline{n}_{\pi^*}^k]^2 \ .$$

To assess the size of the problem, let **A** = $\{\underline{a}_\pi : \pi \in \mathbf{F}^{\underline{x}}\}$. Then, we have the following:

$$|\mathbf{A}| = \frac{R!}{\prod_{j=1}^{K} r^j!}$$

We note that a large number of $\underline{a}_\pi \in$ **A** are equivalent since they are identical under rotation with respect to $E[\underline{n}_\pi^j]^2$. However, even after eliminating these, the resultant space is still non-tractable for large *R*.

A dynamic programming approach to derive an optimal scheduler requires the definition of an additive objective function, i.e., one which is computed incrementally. However, the periodicity metric is a cumulative quantity, which renders the approach unsuitable.

Therefore, our approach is to evaluate the periodicity performance of various known loop schedulers. By comparing against a lower bound (See Section 3), we can determine the 'best' scheduler that maintains intra-class fairness and also quantify its deviation from the optimal scheduler.

3 Conditions for Optimal Per-flow Periodicity for Perfectly-Fair Loop Schedulers

In this section, we determine the conditions for optimal allocation periodicity for some flow *j*. This can be expressed as a requirement on $\underline{n}_\pi^j$ in the following theorem:

Theorem 1 *If $\underline{n}_*^j$ is defined as follows:*

$$\underline{n}_*^j = \{\overbrace{\lfloor\frac{R}{r^j}\rfloor, \cdots, \lfloor\frac{R}{r^j}\rfloor}^{r^j\lfloor\frac{R}{r^j}\rfloor + r^j - R}, \underbrace{\lceil\frac{R}{r^j}\rceil, \cdots, \lceil\frac{R}{r^j}\rceil}_{R - r^j\lfloor\frac{R}{r^j}\rfloor}\}$$

then we have the following:

$$E[\underline{n}_*^j]^2 = \min_{\pi \in \mathbf{F}^{\underline{x}}} E[\underline{n}_\pi^j]^2$$

Proof. We consider the following cases:

***R* ≡ 0 (modulo r^j) :** Perfect allocation periodicity is achieved for flow *j* when the inter-allocation interval is constant, i.e., $n_*^j(k)$ = $n_*^j(m)$. This is achieved if and only if $n_*^j(k)$ = $\frac{R}{r^j}$ for 1$\leq k \leq r^j$.

***R* ≡ *y* (modulo r^j), 1$\leq y \leq r^j$-1 :** In this case, a constant inter-allocation interval for flow *j* cannot be achieved. The best one can achieve is the following for 1$\leq k \leq r^j$:

$$n_*^j(k) \in \{N, N+1\}, \text{ where } 1 \leq N \leq R - r^j.$$

For 1$\leq m \leq r^j$-1, let us assume the following:

$$\underline{n}_*^j = \{\overbrace{N, \cdots, N}^{m}, \underbrace{N+1, \cdots, N+1}_{r^j - m}\}$$

Then, since $\sum_{k=1}^{r^j} n^j(k)$ = *R*, we have the following:

$$m \cdot N + (r^j - m) \cdot (N+1) \quad = \quad R$$

from which we have

$$m \quad = \quad r^j \cdot N + r^j - R$$

However, since 1$\leq m \leq r^j$-1, we have the following constraints on *N*:

$$\frac{R}{r^j} - 1 + \frac{1}{r^j} \leq N \leq \frac{R}{r^j} - \frac{1}{r^j} \qquad (1)$$

Since $\lfloor \frac{R}{r^j} \rfloor$ -1 < $\frac{R}{r^j} - 1 + \frac{1}{r^j}$ and $\lceil \frac{R}{r^j} \rceil > \frac{R}{r^j} - \frac{1}{r^j}$, the only integer *N* that can satisfy Eq. (1) is *N* = $\lfloor \frac{R}{r^j} \rfloor$.

□

The corresponding value for $E[\underline{n}^j_*]^2$ can be used as a *lower* bound for all $\pi \in$ **F**$^{\underline{x}}$ and is given as follows:

$$E[\underline{n}^j_*]^2 \quad = \quad \frac{R(2N+1) - r^j N(N+1)}{r^j}$$

4 Description of *K*-flow Loop Schedulers

In this section, we describe the mechanism as well as the periodicity and intra-class fairness characteristics of several loop schedulers. Without loss of generality, we will assume that $r^j \leq r^k$ for *j*<*k* and $r^j \geq 2$. The case of r^j=1 is trivial since $\underline{n}^j_\pi$ = *R* for $\pi \in$ **F**$^{\underline{x}}$, i.e., perfect-periodicity is always achieved for flow *j*. We denote by $ns^{(j,k)}_\pi(m)$ the cumulative number of slots allocated to flow *k* up to the m^{th} allocation to flow *j* by scheduler π.

4.1 *K*-flow Deficit Round Robin Scheduler (DRR_K)

Fair-queueing schedulers (e.g., Weighted-Fair Queueing (WFQ)) achieve nearly-perfect fairness, but they are usually expensive to implement. DRR_K [10] is an online scheduler that is an approximation to fair-queueing which is simple to implement and yet achieves good fairness and can also be implemented as a loop scheduler. Within the scope of our scheduling problem, the DRR_K scheduler reduces to a Weighted Round Robin (WRR) policy, which simply allocates a block of r^1 slots to flow 1 followed by a block of r^2 slots to flow 2 and so on. Hence, each flow *j* is allocated slots in blocks of size r^j, with an interval of *R*-r^j slots between successive blocks. Therefore, we have the following:

$$\underline{n}^j_{DRR_K} \quad = \quad \{1, \cdots, 1, R - r^j + 1\} \qquad (2)$$

$$E[\underline{n}^j_{DRR_K}]^2 \quad = \quad \frac{r^j + (R-r^j)^2 + 2(R-r^j)}{r^j}$$

The periodicity performance of the DRR_K is given by the following theorem:

Theorem 2 *Although the DRR_K ensures intra-class fairness, it exhibits the worst periodicity amongst $\pi \in$ **F**$^{\underline{x}}$, i.e., for 1$\leq j \leq K$,*

$$E[\underline{n}^j_{DRR_K}]^2 \quad = \quad \max_{\pi \in \mathbf{F}^{\underline{x}}} E[\underline{n}^j_\pi]^2$$

Proof. According to Eq. (2), the periodicity metric for flow *j* is a function of r^j only and hence, the DRR_K scheduler ensures intra-class fairness.

Let us consider an arbitrary scheduler $\pi \in$ **F**$^{\underline{x}}$ with $\underline{n}^j_\pi$ given as follows:

$$\underline{n}^j_\pi \quad = \quad \{1+z_1, \cdots, 1+z_{r^j-1}, R - r^j + 1 - \sum_{y=1}^{r^j-1} z_y\}$$

where $z_y \in \mathbf{Z}^+$, 1$\leq y \leq r^j$-1. We note that for z_y=0, 1$\leq y \leq r^j$-1, π = DRR_K.

Using Eq. (2), $E[\underline{n}^j_\pi]^2$ can be expressed in terms of $E[\underline{n}^j_{DRR_K}]^2$ as follows:

$$E[\underline{n}^j_\pi]^2 \quad = \quad E[\underline{n}^j_{DRR_K}]^2 + \frac{\sum_{y=1}^{r^j-1} z_y^2 + [\sum_{y=1}^{r^j-1} z_y]^2 - 2(R-r^j)\sum_{y=1}^{r^j-1} z_y}{r^j} \qquad (3)$$

Since $\underline{n}^j_\pi$ corresponds to inter-allocation intervals, we have the following constraint:

$$R - r^j + 1 - \sum_{y=1}^{r^j-1} z_y \quad \geq \quad 1$$

In addition, according to the triangular inequality, we have:

$$\sum_{y=1}^{r^j-1} z_y^2 \quad \leq \quad [\sum_{y=1}^{r^j-1} z_y]^2$$

Substituting into Eq. (3), we have the following:

$$\begin{aligned} E[\underline{n}^j_\pi]^2 &\leq E[\underline{n}^j_{DRR_K}]^2 \\ &+ \frac{\sum_{y=1}^{r^j-1} z_y^2 + [\sum_{y=1}^{r^j-1} z_y]^2 - 2[\sum_{y=1}^{r^j-1} z_y]^2}{r^j} \\ &= E[\underline{n}^j_{DRR_K}]^2 + \frac{\sum_{y=1}^{r^j-1} z_y^2 - [\sum_{y=1}^{r^j-1} z_y]^2}{r^j} \\ &\leq E[\underline{n}^j_{DRR_K}]^2 \end{aligned}$$

□

4.2 *K*-flow Credit Round Robin Scheduler (CRR_K)

The motivation to design the CRR_K scheduler [11] was to reduce the latency of the DRR_K scheduler. As with the DRR_K scheduler, the CRR_K scheduler can be implemented as a loop scheduler, and the pseudo-code is given as follows:

_K_-flow Credit Round Robin Scheduler (CRR_K)

```
Initialize $y^j = \frac{r^j}{r^K}$, 1≤ $j \leq K$
Set i=1, SP=K, count=0
while $i \leq R$
        if count < K
           if $y^{SP} < 1$
              count = count + 1
           else
              $a_{CRR_K}(i)$ = SP,
              $y^{SP} = y^{SP}$ - 1, i = i + 1, count = 0
           SP = SP - 1 (modulo K)
        else $y^j = y^j + \frac{r^j}{r^K}$ $\forall$ j, count = 0
```

The CRR_K scheduler possesses the following property for 1≤ $j \leq K$-1:

Property 2 *The m^{th} allocation of flow j always occurs between the $\lceil\frac{mr^K}{r^j}\rceil$ th and $\lceil\frac{mr^K}{r^j}\rceil$ - 1 th allocation of flow K, 1≤ m≤ r^j, i.e.,*

$$ns^{(j,K)}_{CRR_K}(m) = \lceil\frac{mr^K}{r^j}\rceil - 1$$

Proof. With the CRR_K scheduler, the first slot is always allocated to flow *K*. We can consider subsequent allocations in blocks, where each block terminates with the next flow *K* allocation, as illustrated in Fig. 1, where $ns^{(j,K)}_{CRR_K}(m)$ is the number of slots allocated to flow *K* up to the m^{th} allocation to flow *j*.

According to the transmission heuristics given by the pseudo-code, $ns^{(j,K)}_{CRR_K}(m)$ has to satisfy the following conditions:

$$\frac{[ns^{(j,K)}_{CRR_K}(m)+1]r^j}{r^K} \geq m \text{ and}$$
$$\frac{ns^{(j,K)}_{CRR_K}(m)\cdot r^j}{r^K} < m$$

Hence, we obtain $ns^{(j,K)}_{CRR_n}(m)$ = $\lceil\frac{mr^K}{r^j}\rceil$ -1 □

Property 2 can be generalized for a class-based scenario as follows:

Property 3 *For the CRR$_K$ scheduler, flows within each class are allocated in blocks, where the order within class **C**^{u} is $\sum_{m=1}^{u-1}\kappa^m+1$, $\sum_{m=1}^{u-1}\kappa^m$+2, $\cdots$, $\sum_{m=1}^{u}\kappa^m$ for 1≤ $u \leq C$. In addition, the m^{th} block of **C**^{u} will reside between the $\lceil\frac{m\cdot\tilde{r}^C}{\tilde{r}^u}\rceil^{th}$ and $\lceil\frac{m\cdot\tilde{r}^C}{\tilde{r}^u}\rceil - 1^{th}$ block of **C**^{C}, where 1≤ $m \leq \tilde{r}^u$.*

We note from Property 3 that flows within each class are always transmitted in blocks, where each flow from that class is allocated exactly once and the order within each block is constant. Hence, the periodicity characteristics for flows belonging to the same class are identical, i.e., intra-class fairness is maintained.

4.3 *K*-flow Weighted Round Robin with WFQ-like spreading Scheduler (*WRR-sp_K*)

The *WRR-sp_K* scheduler [12] is a variant of the standard WRR scheduler, in which the service order amongst the flows is identical to WFQ. The algorithm for the *WRR-sp_K* scheduler is described as follows:

_K_-flow WRR with spreading Scheduler (*WRR-sp_K*)

```
Let the array $\underline{y}$ contain the sequence
$< \frac{m}{r^j}, j >$: $m \in \{1, \cdots, r^j\}$, 1≤ $j \leq K$
sorted in lexicographic order.
The vector $\underline{a}_{WRR-sp_K}$ is constructed as follows:
    $a_{WRR-sp_K}(i)$ = j if $y(i) =< \frac{m}{r^j}, j >$
```

The *WRR-sp_K* scheduler possesses the following property for 1≤ $j \leq K$-1:

Property 4 *The m^{th} allocation of flow j always occurs between the $\lceil\frac{mr^k}{r^j}\rceil$ th and $\lceil\frac{mr^k}{r^j}\rceil$ - 1 th allocation of flow k, where k>j, 1≤ m≤ r^j, i.e.,*

$$ns^{(j,k)}_{WRR-sp_K}(m) = \lceil\frac{mr^k}{r^j}\rceil - 1$$

Proof. According to the algorithm, the m^{th} allocation to flow *j* is characterized by the parameter $\frac{m}{r^j}$. If *y* denotes the cumulative number of slots allocated to flow *k* up to the m^{th} allocation of flow *j* and $j < k$, then *y* must satisfy the following conditions:

$$\frac{y}{r^k} < \frac{m}{r^j} \text{ and}$$
$$\frac{y+1}{r^k} \geq \frac{m}{r^j}$$

Hence, we obtain $ns^{(j,k)}_{WRR-sp_K}(m)$ = *y* = $\lceil\frac{mr^k}{r^j}\rceil$ -1 □

Property 4 can be generalized for a class-based scenario as follows:

Property 5 *For the WRR-sp_K scheduler, flows within each class are allocated in blocks, where the order within class **C**^{u} is $\sum_{m=1}^{u-1}\kappa^m$+1, $\sum_{m=1}^{u-1}\kappa^m$+2, $\cdots$, $\sum_{m=1}^{u}\kappa^m$ for 1≤ $u \leq C$. In addition, the m^{th} block of **C**^{u} will reside between the $\lceil\frac{m\cdot\tilde{r}^y}{\tilde{r}^u}\rceil^{th}$ and $\lceil\frac{m\cdot\tilde{r}^y}{\tilde{r}^u}\rceil - 1^{th}$ block of **C**^{y}, where y>u and 1≤ $m \leq \tilde{r}^u$.*

As with the CRR_K scheduler, we can deduce from Property 5 that intra-class fairness is maintained for the *WRR-sp_K* scheduler.

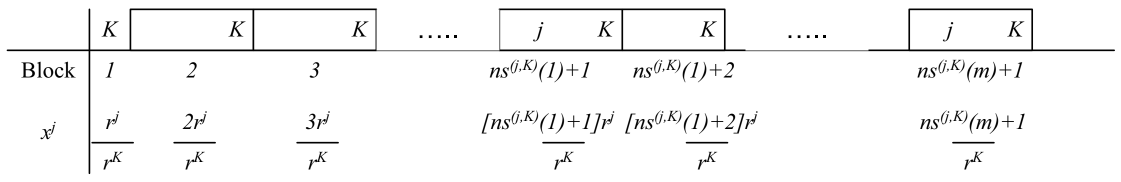


Figure 1: Illustration of allocation to flow K relative to allocation to flow j with the CRR_K scheduler.

$$\underline{n}^j_{WRR-sp_K} = \begin{cases} \{\overbrace{\kappa^1+\kappa^2\lfloor\frac{\tilde r^2}{\tilde r^1}\rfloor,\cdots,\kappa^1+\kappa^2\lfloor\frac{\tilde r^2}{\tilde r^1}\rfloor}^{\tilde r^1\lceil\frac{\tilde r^2}{\tilde r^1}\rceil-\tilde r^2},\underbrace{\kappa^1+\kappa^2\lceil\frac{\tilde r^2}{\tilde r^1}\rceil\cdots\kappa^1+\kappa^2\lceil\frac{\tilde r^2}{\tilde r^1}\rceil}_{\tilde r^2-(\lceil\frac{\tilde r^2}{\tilde r^1}\rceil-1)\tilde r^1}\}, & j\in\mathbf{C}^1; \\ \{\overbrace{\kappa^2,\cdots,\kappa^2}^{\frac{\tilde r^2K-R}{\kappa^1}},\underbrace{K,\cdots,K}_{\frac{R-\kappa^2\tilde r^2}{\kappa^1}}\}, & j\in\mathbf{C}^2. \end{cases} \tag{4}$$

$$\underline{n}^j_{*} = \begin{cases} \{\overbrace{\kappa^1+\lfloor\frac{\kappa^2\tilde r^2}{\tilde r^1}\rfloor,\cdots,\kappa^1+\lfloor\frac{\kappa^2\tilde r^2}{\tilde r^1}\rfloor}^{\tilde r^1(\kappa^1+\lfloor\frac{\kappa^2\tilde r^2}{\tilde r^1}\rfloor)+\tilde r^1-R},\underbrace{\kappa^1+\lceil\frac{\kappa^2\tilde r^2}{\tilde r^1}\rceil,\cdots,\kappa^1+\lceil\frac{\kappa^2\tilde r^2}{\tilde r^1}\rceil}_{R-\tilde r^1(\kappa^1+\lfloor\frac{\kappa^2\tilde r^2}{\tilde r^1}\rfloor)}\}, & j\in\mathbf{C}^1; \\ \{\overbrace{\kappa^2+\lfloor\frac{\kappa^1\tilde r^1}{\tilde r^2}\rfloor,\cdots,\kappa^2+\lfloor\frac{\kappa^1\tilde r^1}{\tilde r^2}\rfloor}^{\tilde r^2(\kappa^2+\lfloor\frac{\kappa^1\tilde r^1}{\tilde r^2}\rfloor)+\tilde r^2-R},\underbrace{\kappa^2+\lceil\frac{\kappa^1\tilde r^1}{\tilde r^2}\rceil,\cdots,\kappa^2+\lceil\frac{\kappa^1\tilde r^1}{\tilde r^2}\rceil}_{R-\tilde r^2(\kappa^2+\lfloor\frac{\kappa^1\tilde r^1}{\tilde r^2}\rfloor)}\}, & j\in\mathbf{C}^2. \end{cases} \tag{5}$$

4.3.1 Optimal *WRR*-sp_K-based Scheduler (*C*=2)

For a two-class scheduling scenario, using Property 5 and Theorem 1, $\underline{n}^j_{WRR-sp_K}$ and $\underline{n}^j_*$ are given in Eq. (4) and Eq. (5) respectively. Comparing these equations, we note that $\underline{n}^j_{WRR-sp_K} \neq \underline{n}^j_*$ for 1$\leq j \leq K$ and hence, the *WRR*-sp_K scheduler is not optimal in terms of per-flow periodicity. However, we note that when κ^1=1 (κ^1=*K*-1), the *WRR*-sp_K scheduler offers optimal periodicity for flows in $\mathbf{C}^2$ ($\mathbf{C}^1$). However, if $\tilde r^1$=1, then optimal (worst-case) periodicity is achieved for flows in $\mathbf{C}^1$ ($\mathbf{C}^2$).

Furthermore, we observe that for the special case of *K*=2, since κ^1=1 =*K*-1, the *WRR*-sp_2 scheduler offers optimal periodicity for *all* flows. Hence, if we define a two-class scheduler, OPT_2, that employs the $WRR-sp_2$ as a class-aware scheduler (Section 2.2), then it can be shown that $\underline{n}^j_{OPT_2}$ = $\underline{n}^j_*$ as given in Eq. (5) for 1$\leq j \leq K$. Hence, the OPT_2 scheduler is optimal for two-class scheduling, and the pseudo-code is given below, assuming $\kappa^1\cdot\tilde r^1 \leq \kappa^2\cdot\tilde r^2$ (the corresponding scheduler for $\kappa^1\cdot\tilde r^1 > \kappa^2\cdot\tilde r^2$ can be obtained by interchanging the indices 1 and 2):

Optimal Two-Class Scheduler (OPT_2)

```
Set $\underline{r}$ = [$\kappa^1\cdot\tilde r^1$, $\kappa^2\cdot\tilde r^2$], $K = \kappa^1 + \kappa^2$
Define $\mathbf{C}^1 = [\overbrace{1,2,\cdots,\kappa^1}^{1},\overbrace{1,2,\cdots,\kappa^1}^{2},\cdots,\cdots,\overbrace{1,2,\cdots,\kappa^1}^{\tilde r^1}]$
Define $\mathbf{C}^2 = [\underbrace{\kappa^1+1,\kappa^1+2,\cdots,K}_{1},\cdots,\cdots,\underbrace{\kappa^1+1,\kappa^1+2,\cdots,K}_{\tilde r^2}]$
Compute $\underline{a}_{WRR-sp_2}$ = WRR-$sp_2(\underline{r})$
for $y$ = 1:2
    index = find($\underline{a}_{WRR-sp_2}$==$y$)
    $a_{OPT_2}$(index) = $\mathbf{C}^y$
```

4.3.2 Enhancement to WRR-sp_K Scheduler

For *C* = 2, according to Section 4.3.1), the *WRR*-sp_K scheduler results in worst-case periodicity for $\mathbf{C}^2$ flows when κ^1=*K*-1 and $\tilde r^1$=1. This is due to the default lexicographic ordering in the scheduling mechanism, which can be circumvented by introducing a parameter, ϱ, $1\leq \varrho \leq K$, to the *WRR-sp_K* scheduler (denoted *WRR-$sp_K(\varrho)$*). With the *WRR-$sp_K(\varrho)$* scheduler, the ordering priority in the event of a tie in the elements $\{\frac{m}{r^j}\}_{m=1}^{r^j}$ for $1\leq j \leq K$ is given by [ϱ,ϱ+1,ϱ+2,$\cdots$,K,1,2,$\cdots$,ϱ-1]. We note that the scheduler reduces to the original *WRR-sp_K* scheduler when ϱ = 1.

We define an optimal $WRR-sp_K$ scheduler (denoted $WRR-sp^*_K$), where $WRR-sp^*_K$ = *WRR-$sp_K(\varrho^*)$* such that ϱ^* = arg $\min_{1\leq\varrho\leq K} wcov_{WRR-sp_K(\varrho)}$.

4.4 *K*-flow Golden Ratio (GR_K) Scheduler

The Golden Ratio Scheduler was proposed in [8] and is described as follows:

```
K-flow Golden Ratio Scheduler ($GR_K$)

Let $z$ = 0.6180339887 and $w(m) = frac(m \cdot z)$
where $frac(y) = y - \lfloor y \rfloor$
Let the array $\underline{y}$ contain the sequence $w(m)$,
$0\leq m \leq R$-1, sorted in ascending order.
The vector $\underline{a}_{GR_K}$ is constructed as follows:
    $a_{GR_K}(i) = j$
if $\sum_{k=1}^{j-1} x^k \leq y(i) \leq \sum_{k=1}^{j} x^k$, $1\leq j \leq K$
```

It was established in [6] that if R is a Fibonacci number, then $\underline{n}^j_{GR_K}$ comprises at most three values for each j; otherwise, more values are generated.

4.5 *K*-flow Short-term Fair Scheduler (STF_K)

We can characterize the *fairness* performance of any loop scheduler in terms of the *cumulative service-deficit*, $sd^j(i)$, which measures the discrepancy between the requested and allocated share for flow *j* up to slot *i*, $1\leq i \leq R$. If $y^j(i)$ denote the cumulative number of slots allocated to flow *j* up to and including slot *i*, then we have the following:

$$sd^j(i) = x^j - \frac{y^j(i)}{i}$$

A *positive* (negative) value of $sd^j(i)$ implies that flow *j* has received *less* (more) than its requested share up to slot *i*. Hence, we consider a scheduler that allocates each slot to the flow with maximum instantaneous service-deficit so as to achieve maximum fairness (Short-term Fair or STF_K scheduler). Whenever there is a tie, priority for allocation is given to the flow with the highest flow index. The pseudo-code for the STF_K scheduler is given as follows:

```
K-flow Short-term Fair Scheduler ($STF_K$)

Initialize $y^j(0)$ = 0, $1\leq j \leq K$
for $i$=1:$R$
    $y^j(i) = y^j(i-1)$, $1\leq j \leq K$
    $sd^j(i) = \frac{r^j}{R} - \frac{y^j(i)}{i}$, $1\leq j \leq K$
    $a_{STF_K}(i)$ = arg $\max_{1\leq j\leq K} sd^j(i)$
    $y^{a_{STF_K}(i)}(i) = y^{a_{STF_K}(i)}(i)+1$
```

This scheduler was first suggested in [8], where the authors conjectured, based on numerical calculations, that it is a promising scheduler. However, no analysis of the scheduler was provided in terms of periodicity properties.

4.6 *K*-flow Random (RND_K) Scheduler

The loop schedulers considered so far are *deterministic* since the allocation vector $\underline{a}_\pi \in \mathbf{A}$ is fixed. In this section, we define a *random* scheduler, RND_K, whose allocation vector, $\underline{a}_{RND_K}$, is *uniformly* selected from **A**. We note that $RND_K \in \mathbf{F}^{\underline{x}}$ because the selected $\underline{a}_{RND_K}$ is used for allocation in each loop.

Let us refer to an allocation sequence based on the RND_K scheduler, and consider a particular loop that begins with the $(m-1)^{th}$ allocation to flow *j*, as illustrated in Fig. 2. Since r^j slots must be allocated to flow *j* in any loop, the total number of ways the $\{k^{th}\}_{k=m}^{m+r^j-2}$ allocations to flow *j* can occur within *R*-1 slots is $\binom{R-1}{r^j-1}$. However, the corresponding expression that ensures that $n^j_{RND_K}(m)$ = *N* is given by $\binom{R-N-1}{r^j-2}$. Hence, for $1\leq N\leq R-r^j+1$,

$$\text{Prob}(n^j_{RND_K}(m) = N) = \frac{\binom{R-N-1}{r^j-2}}{\binom{R-1}{r^j-1}}$$

The periodicity metric for each flow *j* is evaluated as follows:

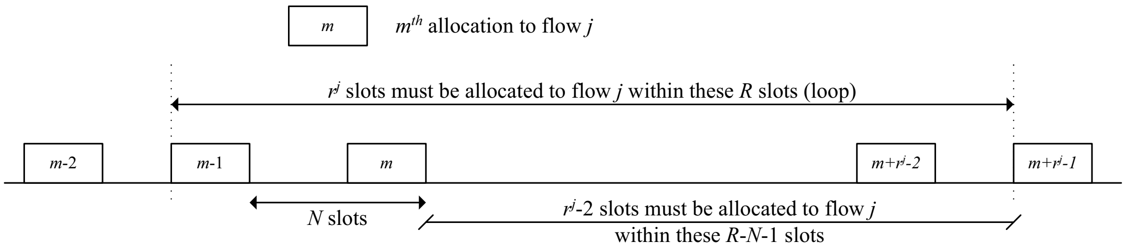
$$\begin{aligned} E[\underline{n}^j_{RND_K}]^2 &= \sum_{N=1}^{R-r^j+1} \text{Prob}(n^j_{RND_K}(m) = N) \cdot N^2 \\ &= \frac{R(2R-r^j+1)}{r^j(r^j+1)} \qquad (6) \end{aligned}$$

By comparing Eq. (6) with Eq. (2), we have the following theorem:

Theorem 3 *The periodicity of the RND_K scheduler is lower-bounded by that of the DRR_K scheduler, i.e., for $1\leq j \leq K$,*

$$E[\underline{n}^j_{RND_K}]^2 \leq E[\underline{n}^j_{DRR_K}]^2$$

We note that unlike the DRR_K, CRR_K and *WRR-sp_K* schedulers, the GR_K, STF_K and RND_K schedulers *do not* ensure intra-class fairness in a class-based scheduling scenario.



m
m^{th} allocation to flow j
r^j slots must be allocated to flow j within these R slots (loop)
m-2
m-1
m
$m+r^j$-2
$m+r^j$-1
N slots
r^j-2 slots must be allocated to flow j within these R-N-1 slots

Figure 2: Evaluation of Prob($n^j_{RND_K}(m)$=N) for the random scheduler.

5 A Recursive Class-aware Perfectly-Fair Loop Scheduler

In this section, we propose a class-aware recursive scheduler (denoted *REC*($\underline{C}$), where $\underline{C}$=$\{$1,2,$\cdots$,$C\}$) that achieves good periodicity performance in a *C*-class scheduling scenario.

The mechanism of the $REC(\underline{C})$ scheduler comprises partitioning the problem into *C*-2 levels of sub-problems (*Forward*), solving each level *C*-2 sub-problem (*Solution*) and substituting the solutions in the return path (*Return*) to obtain the required allocation vector for $REC(\underline{C})$. We describe each of the 3 components as follows:

Forward : We begin by partitioning the $REC(\underline{C})$ scheduler (level 0 problem) into sub-problems. At the first level, we obtain *C* level 1 sub-problems, $REC(\underline{I}_1), REC(z_1)$, $\forall\ z_1 \in \underline{C}$ and $\underline{I}_1$=$\underline{C}\backslash z_1$. In the same manner, each level 1 sub-problem is further partitioned into *C*-1 level 2 sub-problems, $REC(\underline{I}_2), REC(z_2)$, where $|\underline{I}_2|$=*C*-2. This continues until the (*C*-2)th level, where we obtain level *C*-2 sub-problems, $REC(\underline{I}_{C-2}), REC(z_{C-2})$, where $|\underline{I}_{C-2}|$=2.

Solution : For each level *C*-2 sub-problem, we perform two-stage scheduling (intra-class scheduling to obtain $\underline{a}_{REC(\underline{I}_{C-2})}$ and $\underline{a}_{REC(z_{C-2})}$, followed by inter-class scheduling to obtain $\underline{a}_{REC(\underline{I}_{C-2}),REC(z_{C-2})}$. The respective functions, *intra-c*() and *inter-c*(), are defined as follows:

intra-$c(\underline{I}, z)$ **:** This function allocates slots to flows $\in$ $\mathbf{C}^{\underline{I}}$ and flows $\in$ $\mathbf{C}^z$ independently to obtain $\underline{a}_{REC(\underline{I})}$ and $\underline{a}_{REC(z)}$ respectively. Since the flows $\in$ $\mathbf{C}^z$ are homogeneous with respect to their requested shares, a simple RR allocation is sufficient to obtain $\underline{a}_{REC(z)}$. For $|\underline{I}|$=2, we can exploit the optimality property of the OPT_2 scheduler to obtain $\underline{a}_{REC(\underline{I})}$.

inter-$c(\underline{a}_{REC(\underline{I})}, \underline{a}_{REC(z)})$ **:** This function combines $\underline{a}_{REC(\underline{I})}$ and $\underline{a}_{REC(z)}$ to obtain $\underline{a}_{REC(\underline{I}),REC(z)}$.

Let $\underline{v}$=$\underline{a}_{REC(z)}$, $\underline{w}$=$\underline{a}_{REC(\underline{I})}$, assuming that $|\underline{a}_{REC(z)}| \leq |\underline{a}_{REC(\underline{I})}|$. Our approach is to insert the elements of $\underline{w}$ into the vector $\underline{v}$ so that successive elements of $\underline{v}$ are as uniformly-spaced as possible in the combined vector, i.e., we attempt to maximize allocation smoothness with respect to $\underline{v}$. This is illustrated in Fig 3, where P = $\lceil\frac{|\underline{w}|}{|\underline{v}|}\rceil$ and Q is given as follows:

$$Q \;=\; \begin{cases} P, & |\underline{w}| = |\underline{v}|; \\ P-1, & \text{otherwise.} \end{cases}$$

For the case where $|\underline{a}_{REC(z)}| > |\underline{a}_{REC(\underline{I})}|$, we simply swap the definition of $\underline{v}$ and $\underline{w}$.

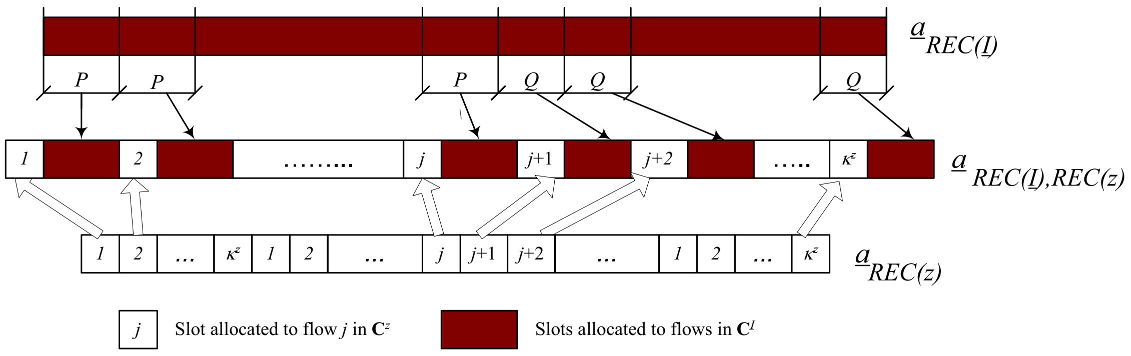
Return : Once we solve all the level *C*-2 sub-problems, we begin the return path. For each level *C*-3 sub-problem, $REC(\underline{I}_{C-3}), REC(z_{C-3})$, we deduce the 'best' allocation vector, $\underline{a}^*_{REC(\underline{I}_{C-3})}$, according to the ensemble periodicity metric by considering the allocation vectors of all its child sub-problems at level *C*-2. We apply the inter-class scheduling function, *inter-c*(), to obtain $\underline{a}_{REC(\underline{I}_{C-3}),REC(z_{C-3})}$ from $\underline{a}^*_{REC(\underline{I}_{C-3})}$ and $\underline{a}_{REC(z_{C-3})}$. We repeat the procedure for level *C*-4,*C*-5,$\cdots$, and eventually, we will obtain the allocation vector of the original level 0 problem (*REC*($\underline{C}$)).

5.1 Illustration of Mechanism for $REC(\underline{C})$ (*C*=4)

We illustrate the mechanism of $REC(\underline{C})$ for the case of *C*=4, where the original problem is partitioned into 2 levels as shown in Fig. 4.

Forward : At the first level, the *REC*(1,2,3,4) scheduler is partitioned into 4 sub-problems *REC*($\underline{C}\backslash z$),*REC*($z$), for each $z \in \underline{C}$. Let us consider the level 1 sub-problem, *REC*(2,3,4),*REC*(1). As with *REC*(1,2,3,4), *REC*(2,3,4) can be partitioned into 3 level 2 sub-problems, given by *REC*(3,4),*REC*(2), *REC*(2,4),*REC*(3) and *REC*(2,3),*REC*(4).

Solution : We consider the problem *REC*(3,4),*REC*(2). We apply *intra-c*() to obtain $\underline{a}_{REC(3,4)}$ and $\underline{a}_{REC(2)}$, and then apply *inter-c*() to compute $\underline{a}_{REC(2,3),REC(4)}$. The allocation vectors,



$\underline{a}_{REC(\underline{I})}$
P
P
P
Q
Q
Q
1
2
………
j
j+1
j+2
…..
κᶻ
$\underline{a}_{REC(\underline{I}),REC(z)}$
1
2
…
κᶻ
1
2
…
j
j+1
j+2
…
1
2
…
κᶻ
$\underline{a}_{REC(z)}$
Slot allocated to flow j in Cᶻ
Slots allocated to flows in Cᴵ

Figure 3: Illustration of the inter-class scheduling mechanism of the $REC(\underline{C})$ scheduler.

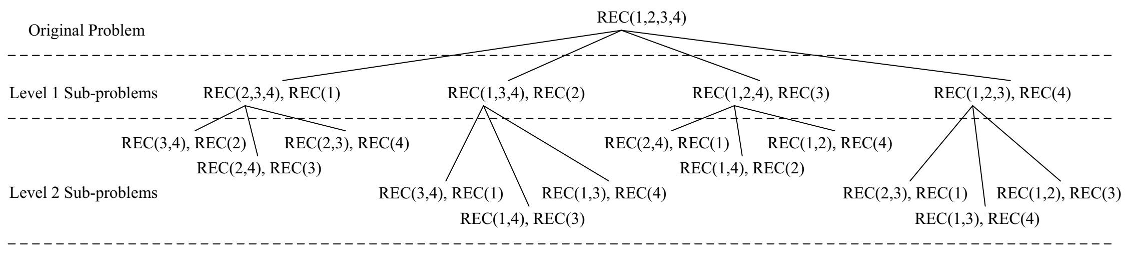


Figure 4: Illustration of the mechanism of the $REC(\underline{C})$ scheduler for C=4.

$\underline{a}_{REC(2,4),REC(3)}$ and $\underline{a}_{REC(2,3),REC(4)}$ are obtained in a similar manner.

Return : Next, we determine the 'best' allocation vector (denoted $\underline{a}^*_{REC(2,3,4)}$) amongst $\underline{a}_{REC(2,3),REC(4)}$, $\underline{a}_{REC(2,4),REC(3)}$ and $\underline{a}_{REC(2,3),REC(4)}$ in terms of the ensemble periodicity metric.

For the level 1 sub-problem *REC*(2,3,4),*REC*(1), we obtain the corresponding allocation vector, $\underline{a}_{REC(2,3,4),REC(1)}$ from $\underline{a}^*_{REC(2,3,4)}$ and $\underline{a}_{REC(1)}$ using *inter-c*(). In a similar manner, the allocation vectors for the remaining level 1 sub-problems, i.e., $\underline{a}_{REC(1,3,4),REC(2)}$, $\underline{a}_{REC(1,2,4),REC(3)}$ and $\underline{a}_{REC(1,2,3),REC(4)}$, can be computed. The final step involves determining the 'best' allocation vector, $\underline{a}^*_{REC(1,2,3,4)}$, in the same way as for the level 1 sub-problem *REC*(2,3,4),*REC*(1).

5.2 Computational Requirement of *REC*($\underline{C}$) scheduler

According to the description in Section 5.1, we require $\binom{C}{2}$ runs of *intra-c*() and $C(2^{C-1}$-$C)$ runs of *inter-c*() and computation of the ensemble periodicity metric to obtain $\underline{a}_{REC(\underline{C})}$. The fact that C is usually small makes the problem tractable in practical cases.

6 Numerical Results

We consider the following broadband applications with the corresponding typical bandwidth requirements in kbps [13]: Streaming Video (Internet Quality) (128), Residential Voice (300), Video Telephony (400), Interactive Games (500) and Streaming video (Video-on-Demand Quality) (3700). We define various C-class scheduling scenarios (where each class comprises flows from a particular application) and compare $wcov_\pi$ obtained for each scenario for the perfectly-fair loop scheduler defined in Section 4 and 5. For example, if we consider Residual Voice, Video Telephony and Interactive Games, then we have $\tilde{\underline{r}}$=[300,400,500] ≡ [3,4,5]. We assume that the flow composition is uniform, i.e., $\kappa^c = \kappa$ for $1 \le c \le C$.

We compare the periodicity performance corresponding to various schedulers for different flow configurations and C=3 in Fig. 5. The corresponding results for C=4 and 5 are shown in Fig. 6. Numerical results for other flow configurations can be found in [14].

Although not depicted in the figures, we note that the RND_K scheduler performs significantly worse than the deterministic schedulers (excluding the DRR_K scheduler). In addition, the performance for each scheduler is relatively invariant with κ for κ >1 for a given $\tilde{\underline{r}}$. Hence, we consider the following cases:

κ **>1 :** Between the *WRR-sp*$_K$ and CRR_K schedulers, the CRR_K performs *worse*. In addition, an enhancement of the *WRR-sp*$_K$ always exists and the

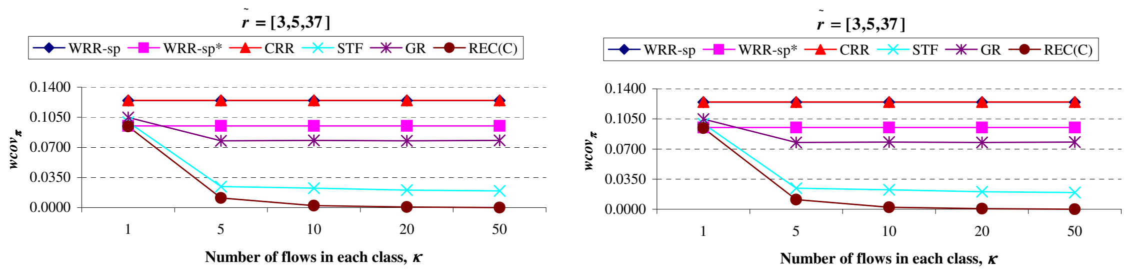


Figure 5: Comparison of $wcov_\pi$ corresponding to various $\pi \in \mathbf{F}^{\underline{x}}$ for $\underline{\tilde{r}}$ = [3,4,5] (left) and [3,5,37] (right) in 3-class scheduling.

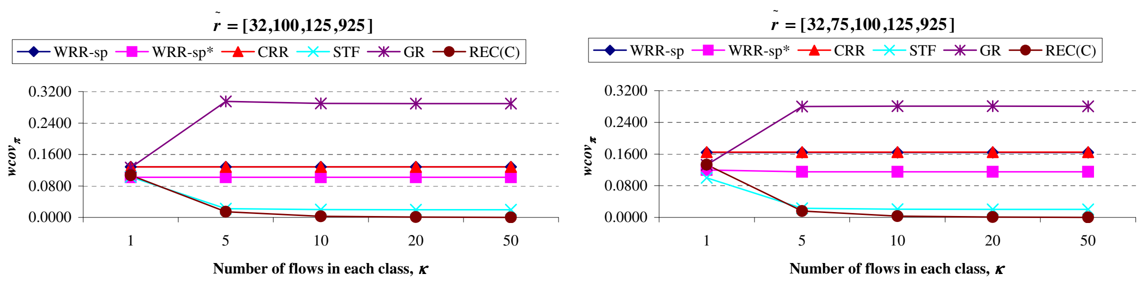


Figure 6: Comparison of $wcov_\pi$ corresponding to various $\pi \in \mathbf{F}^{\underline{x}}$ for $\underline{\tilde{r}}$ = [32,100,125,925] (left) and [32,75,100,125,925] (right) in 4-class and 5-class scheduling.

gain in terms of the weighted covariance is significant. Amongst the $REC(\underline{C})$, STF_K and GR_K schedulers, the relative performance is always according to the above order, with the $REC(\underline{C})$ scheduler achieving the *best* performance. In fact, $wcov_{STF_K}$ and $wcov_{REC(\underline{C})} \approx 0$ for scenarios where $\tilde{r}^C >> \tilde{r}^c$, 1$\leq c \leq C$-1.

While the *WRR-sp*$_K$, CRR_K and *WRR-sp*$^*_K$ schedulers ensure intra-class fairness for any scheduling scenario, it is not enforced by the STF_K and $REC(\underline{C})$ schedulers for certain scenarios, and is never enforced by the GR_K scheduler for any scenario. Hence, there is a trade-off between achieving good periodicity performance and ensuring intra-class fairness. If the latter needs to be guaranteed for any class-scheduling scenario, then the *WRR-sp*$^*_K$ scheduler should be used; otherwise, the $REC(\underline{C})$ scheduler should be used.

κ=1 : For an easier comparison of the periodicity performance of the schedulers, we plot the results for κ=1 in Fig. 7. The STF_K and *WRR-sp*$^*_K$ schedulers offer the best overall periodicity performance, while the GR_K and CRR_K schedulers offer the worst performance. We note that intra-class fairness is irrelevant in this case.

7 Applications in Wireless Networks

7.1 Channel-aware Wireless Schedulers

Most wireless schedulers proposed recently can be mapped onto a Unified Wireless Fair Queuing Framework (UWFQF) [15]. In this framework, a wireless scheduler comprises a wireline scheduler as well as a wireless adaptation scheme. While scheduling is performed using the wireline algorithm under error-free conditions, the wireless adaptation scheme takes over when these conditions no longer prevail.

The perfectly-fair loop schedulers considered in this study can be employed as the wireline algorithm in wireless schedulers since they are simple to implement and also analytically tractable. We perform a Quality of Service (QoS) analysis of this class of wireless schedulers in [16].

7.2 Energy-efficient MAC Protocols

Energy efficiency is an important issue in wireless networks, particularly in wireless sensor networks [17]. In such networks, most nodes are likely to be battery-powered, and it is difficult to change batteries due to the remote deployment under harsh environments of such networks. Hence, one of the major design goals of most wireless sensor networks is to prolong the lifetime of sensor nodes by minimizing energy waste.

The radio is a major energy consumer in any sensor node platform, where the major sources of energy waste

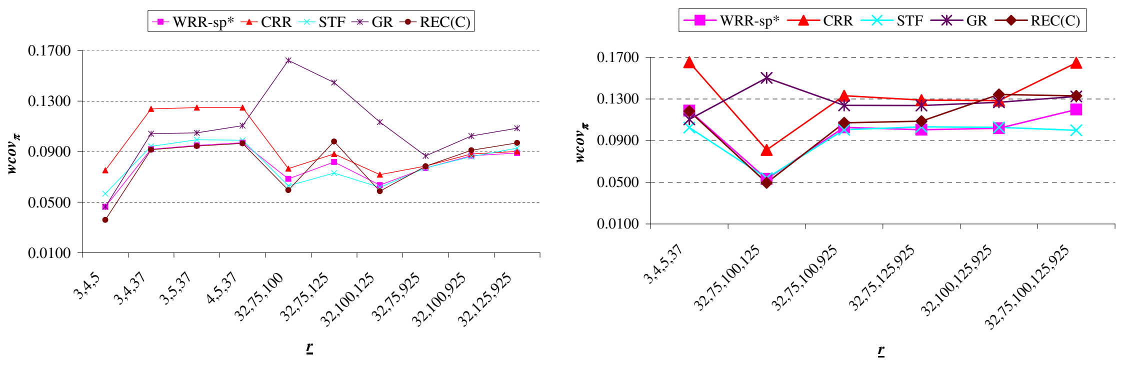


Figure 7: Comparison of $wcov_\pi$ for various $\pi \in \mathbf{F}^{\underline{x}}$ for K-flow scheduling where K=3 (left) and K=4,5 (right).

are collision and idle listening. Medium Access Control (MAC) protocols determine when and how each node accesses the wireless channel and can be broadly classified into two groups: scheduled and contention-based. Although scheduled protocols have limited scalability and adaptivity, they are energy-efficient since collisions are avoided and nodes can go to sleep in intervals where they are not allocated channel access.

In future work, we plan to investigate the trade-offs amongst QoS, fairness and energy-efficiency by deploying perfectly-fair loop schedulers in wireless sensor networks.

8 Conclusions

In this paper, we consider the problem of finding a weighted time-division multiplexed loop scheduler for K flows that minimizes the average packet delay. The optimization criteria translates to finding a loop scheduler that allocates slots to flows as periodically as possible. We use the second moment of the inter-allocation distance for a flow i as the periodicity metric of the scheduler with respect to that flow. We derive the conditions for optimal per-flow periodicity for any K-flow loop scheduler.

We consider a class-based scheduling scenario where flows can be grouped according to their relative bandwidth demands. We analyze the periodicity properties of a weighted round robin with spreading (*WRR-*sp_K) scheduler for a two-class scenario. Based on these properties, we establish an optimal scheduler that employs the *WRR-*sp_K scheduler as an inter-class scheduler, and also suggest an enhancement to the *WRR-*sp_K scheduler. We then propose a recursive class-aware scheduler based on the two-class optimal scheduler.

We define a metric that reflects the periodicity performance over the ensemble of all flows. We then compare the performance of the above schedulers in terms of numerical results. Although the recursive scheduler achieves the best periodicity performance, it fails to guarantee intra-class fairness, which is desirable for class-based scheduling. On the other hand, the enhanced *WRR-*sp_K scheduler gives the best periodicity performance amongst those schedulers that maintain intra-class fairness. Hence, there is a trade-off between periodicity and fairness performance in the design of loop schedulers.

References

[1] S. Baruah, G. Buttazzo, S. Gorinsky, and G. Lipari, “Scheduling periodic task systems to minimize output jitter,” *Proc. of the IEEE International Conference on Real-Time Computing Systems and Applications*, pp. 62–69, December 1999.

[2] A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber, “Minimizing Service and Operation Cost of Periodic Scheduling,” *Proc. of the ACM Symposium on Discrete Algorithms*, pp. 11–20, January 1998.

[3] Z. Brakerski, A. Nisgav, and B. Patt-Shamir, “General Perfectly Periodic Scheduling,” *Proc. of the ACM Symposium on Principles of Distributed Computing*, pp. 163–172, July 2002.

[4] A. Bar-Noy, A. Nisgav, and B. Patt-Shamir, “Nearly Optimal Perfectly-Periodic Schedules,” *Proc. of the ACM Symposium on Principles of Distributed Computing*, pp. 107–116, August 2001.

[5] A. Bar-Noy, V. Dreizin, and B. Patt-Shamir, “Efficient Periodic Scheduling by Trees,” *Proc. of the IEEE INFOCOM*, vol. 2, pp. 791–800, June 2002.

[6] M. Hofri and Z. Rosberg, “Packet Delay under the Golden Ratio Weighted TDM Policy in a Multiple-Access Channel,” *IEEE Trans. Information Theory*, vol. 33, no. 3, pp. 341–349, May 1987.

[7] Z. Rosberg, “Optimal Decentralized Control in a Multiaccess Channel with Partial Information,” *IEEE Transactions on Automatic Control*, vol. 28, no. 2, pp. 187–193, February 1983.

[8] A. Itai and Z. Rosberg, “A Golden Ratio Control Policy for a Multiple-Access Channel,” *IEEE Trans.*

Information Theory, vol. 33, no. 3, pp. 341–349, May 1987.

[9] Y. Guo and H. Chaskar, "Class-based Quality of Service over Air Interfaces in 4G Mobile Networks," *IEEE Communications Magazine*, vol. 40, no. 3, pp. 132–137, March 2002.

[10] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," *IEEE/ACM Transanctions on Networking*, vol. 4, no. 3, pp. 375–385, June 1996.

[11] V. Do and K. Yun, "An Efficient Frame-Based Scheduling Algorithm: Credit Round Robin," *Proc. of the IEEE Workshop on HPSR*, pp. 103–110, June 2003.

[12] H. Zhang, "Service disciplines for guaranteed performance service in packet - switching networks," *Proceeding of the IEEE*, vol. 83, no. 10, pp. 1374–1399, October 1995.

[13] S. Viswanathan, "Future View of Broadband Demand," FCC TAC Meeting, Intel, April 2003, Available at `http://www.fcc.gov/oet/tac/TAC_III_04_17_03/Future_View_of_Broadband_De%mand.ppt`.

[14] R. Rom, M. Sidi, and H. P. Tan, "Performance Analysis of a Recursive Cyclic Scheduler for Class-based Scheduling," CCIT Tech Report 470, Technion, Israel Institute of Technology, February 2004, Available at `http://www.ee.technion.ac.il/CCIT/info/Publications/Sceintific_e.asp`.

[15] T. Nandagopal, S. Lu, and V. Bharghavan, "A Unified Architecture for the Design and Evaluation of Wireless Fair Queuing Algorithms," *Proc. of the ACM MOBICOM*, pp. 132–142, October 1999.

[16] R. Rom and H. P. Tan, "Stochastic Analysis and Performance Evaluation of Wireless Schedulers," *Wiley Journal of Wireless Communications and Mobile Computing Special Issue: Performance Evaluation of Wireless Networks*, vol. 4, no. 1, pp. 19–41, February 2004.

[17] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks (Elsevier) Journal*, vol. 38, no. 4, pp. 393–422, March 2002.