The impact of local policies on the quality of packet routing in paths, trees, and rings

Eric Angel [1], Evripidis Bampis [1], Fanny Pascual [2]

Abstract

We consider the packet routing problem in store-and-forward networks whose topologies are either paths, trees, or rings. We are interested by the quality of the solution produced, with respect to a global optimal solution, if each link uses a (fixed) local policy to schedule the packets which go through it. The quality of the derived solutions is measured using the worst case analysis for two global optimality criteria, namely the maximum arrival date of a packet at its destination (or makespan) and the average arrival date of the packets at their destinations.

We consider the setting where n packets, each one having a size (or length) and a destination, are released from the same source. In the case of rings, there exist two paths between the source and a destination. Each packet is owned by a user which chooses a path to its destination. We assume that users are rational: knowing the local policy used by the links and the state of the network, a user chooses the path which minimizes the arrival date of its packet at its destination. We are then interested by the quality of the Nash equilibria obtained.

1 Introduction

The classical problem of *packet routing in store-and-forward networks* can be formalized as follows. We are given a network represented by a directed graph, where the nodes are the switches and the arcs are the communication links. In general, a node can serve as the source or destination of an arbitrary number of packets (or tasks). We are also given a set of packets, each one characterized by its length and an ordered pair of nodes (source, destination) that have to be routed through the network from their sources to their destinations. At most one packet can be routed at the same time on a given link and a packet cannot be routed on several links at the same time. The time that a packet needs to cross a link is proportional to the length of the packet, and in the sequel we assume w.l.o.g. that this time is equal to the length of the packet. Before the routing starts, all the packets are stored at their sources in special buffers. During the routing, packets may wait in the buffers of intermediate nodes.

We focus on particular network topologies, namely paths, trees, and rings. In the case of paths and trees there is a unique path between the source and each destination node. Thus the routing problem in these network topologies consists in scheduling the packets on each link along the paths linking the source to the destinations of the packets. We study the quality of the solutions obtained by using different local policies for scheduling the packets on every link, instead of a centralized algorithm. In the case of rings, we assume that each packet is owned by a selfish (rational) agent who has the choice of the path to reach its destination. This situation is modeled as a non-cooperative game, in which each agent has two possible (pure) strategies which correspond to the choice of a path. We assume that each agent knows the local policy used by the links and has a complete information about the traffic in the network (the lengths and destinations of the packets owned by the other agents), when he chooses his strategy. His goal is to minimize the arrival date of the packet he owns. We are interested by solutions which are pure Nash equilibria, i.e., a combination of pure strategies (choices of paths) for the agents

[1]IBISC, Université d'Évry Val d'Essonne, 523 Place des Terrasses, 91000 Évry, France. E-mail: {angel, bampis}@ibisc.univ-evry.fr

[2]LIP6, Université Pierre et Marie Curie, 104 avenue du Président Kennedy, 75016 Paris, France. E-mail: fanny.pascual@lip6.fr

such that no agent has an incentive to unilaterally change its strategy given the strategies chosen by the other agents.

The quality of the derived solution is measured using the worst case analysis for two optimality criteria, namely the maximum arrival date (*maximum completion time* or *makespan*) and the average arrival date (*average completion time*) of the packets (*tasks*) at their destinations.

In the case of paths and trees, the measure that we use in order to evaluate the loss of performance is the classical measure in approximation algorithms, namely the *approximation ratio*, which is defined as the maximum, over all possible instances, of the ratio of the value of the objective function obtained in our setting, over the optimal value of the studied objective function.

In the case of rings, we use the notion of *price of anarchy* [8] defined over all *pure* Nash equilibria, i.e., Nash equilibria where the strategy of each task is one path and not a distribution of probabilities over several choices. Formally, let $\mathcal{I}$ be the set of all possible instances, and let us denote by $\mathcal{N}(I)$ the set of all pure Nash equilibria for instance $I \in \mathcal{I}$. We denote by OPT_I the value of the studied objective function at an optimal solution. The *price of anarchy* is defined as:

$$\text{Price of anarchy} = \max_{I \in \mathcal{I}, N \in \mathcal{N}(I)} \frac{\text{Objective function value in } N}{OPT_I}.$$

Local policies. The links of the network have a *common local policy* which allows them to give an order to the packets: when a link becomes idle, it chooses, according to the adopted policy, a packet to route (without preemption) among the waiting packets, if any, in its buffer. The policy of the links is decided once and for all and does not depend on the instance of the packets to be routed.

We study the impact of the following policies on the quality of the produced solutions:

- SPT: *Shortest Processing Time*. The packet (task) which has the smallest length (i.e. processing time) is scheduled first.
- LPT: *Longest Processing Time*. The packet which has the largest length is scheduled first.
- LRD: *Largest Remaining Distance*. The packet which has the largest remaining distance is scheduled first. The remaining distance of a packet is the number of arcs it has to cross to arrive at its destination.
- LRT: *Longest Remaining Time*. The packet which has the longest remaining time to its destination is scheduled first. The remaining time of a packet is its length times the number of arcs it has to cross to arrive at its destination.

1.1 Related works

The packet routing problem in general store-and-forward networks is an NP-hard problem [14] which has been extensively studied in the literature (see for example [1, 9, 10, 11]). However, in these works, the authors consider the setting where all the packets have the same length, and they mainly give centralized algorithms. Randomized distributed or online algorithms have been proposed in [1, 9, 11], but in these algorithms the policy of each link does not only depend on the packets that are allocated to this link, but also on the total number of packets in the network.

Another related problem, studied in [6], is the *multicommodity flow over time problem*, where flows do not travel instantaneously through a network but require a certain amount of time to travel through each arc: a flow over time specifies a flow rate entering an arc for each point in time. Notice that contrary to our model, a packet may start to be sent from an intermediate

node before having been completely received at this node. Besides showing that the problem is NP-hard, the authors propose an efficient centralized algorithm. They also give a greedy algorithm for the case where the out-degree of each node is at most one (this result is useful in Section 2.3).

A closely related work is the one of [4] in which the authors studied a scheduling problem on parallel links. In their model, they introduce a local policy for each parallel link, allowing to give an order to the tasks that are executed on the considered link. This model is similar to the one considered in our paper (especially to the case of rings), since it can be described as follows: a set of n selfish packets is initially located at a common source and has to be transmitted to a common destination through a simple network composed of a set of m parallel edges (communication links) connecting the source to the destination node. Every link has a public local policy, known to all packets, which determines the order in which the packets that are allocated to this link will be scheduled. Each packet is characterized by its length and aims to adopt a strategy –choice of a link– that minimizes its arrival (completion) time. From the network point of view, the quality of a schedule is measured in terms of its makespan, i.e., the time at which the last packet arrives at its destination. More recently, in [7], the authors compared the price of several policies (SPT, LPT, Randomized) in the case of parallel links with different speeds.

We also have to mention two other models that consider the selfish packet routing problem in networks. The model introduced in [13] consists of routing in a network a very large number of packets of negligible size, giving rise to a flow problem where the aim of each packet is to minimize its arrival date, and the global objective function is the maximum arrival date. The second model considers *congestion games*, introduced in [12] and studied in [3]: the cost of each packet i is the sum of the costs of the links crossed by i, where the cost of a link is a function of the number of packets that are using this link. The main difference with respect to our model is that the cost of a link does not take into account the time at which the packets effectively cross this link. If the network is a parallel links network, like in the case of [8] or [4], the price of anarchy of the congestion game corresponds to the price of anarchy considered in [8], where the tasks are scheduled in a round-and-robin way. However, in the case of congestion games if the network topology is different, each link used by two tasks will add a cost for each of these two tasks, even if no task has to wait for the other one to cross the link.

1.2 Terminology and notations

For clarity reasons, in the sequel we adopt the scheduling terminology, e.g., we use the term task instead of packet. In the setting that we consider, n tasks are available (present) on the same node (called the source) at time 0 and have the same or different destinations. Furthermore, the links of the network have the same local policy which is either SPT, LPT, LRD, or LRT.

We consider *store-and-forward networks*: a task must be received completely by a node before it can be sent on. Tasks wait on the buffer of the link they wish to take until this link becomes free. We assume that the buffers are infinitely large. As soon as a link becomes free, the task with the highest priority in the buffer is scheduled. Notice that this way a large packet may make a short packet wait several times on its path, if the large packet has a priority higher than the short packet.

In the sequel, we will denote the length of task i by l_i ($l_i > 0$), and the number of arcs in the shortest path between the source and the destination of i will be denoted by x_i. C_i denotes the completion time of task i, and C_{max} denotes the completion time of the last task completed. A_i denotes the outgoing arc taken by task i from the source, and $\mathcal{B}_i$ is the set of tasks which went through arc A_i before task i.

We consider the two following problems: the *Maximum Completion Time Problem*, in which we wish to minimize the date at which all the tasks have been completed, and the *Average Completion Time Problem*, in which we wish to minimize the average completion time of the tasks. This last problem is equivalent to the one of minimizing the sum of the completion times of all the tasks.

1.3 Outline of the paper

In Section 2, we study the approximation ratio of the four studied policies when the network is a path or a tree for the two considered objective functions. In Section 3, we study the price of anarchy in the case where the network is a ring. In Section 4, this loss of performance is studied in the case where all the tasks have the same source and the same destination, in path and ring networks.

Table 1 (resp. Table 2) shows the approximation ratio (or price of anarchy in rings) when there are several destinations (resp., one destination), for the Maximum Completion Time problem (columns denoted by C_{max}) and the Average Completion Time problem (columns $\sum C_i$). We give either tight bounds, or a lower bound and an upper bound. In this latter case, r denotes the approximation ratio (or price of anarchy for rings).

Each policy (e.g., SPT) may have a *sub-policy* (another policy) to schedule its tasks when there is a tie. For example, SPT may schedule the tasks which have the same length with the LRD policy. It is important to note that the results given by the following tables hold *whatever* the sub-policy of each policy is. Therefore, no result is given for the LRD policy for the Average Completion Time in the case where there is a single destination in paths or trees (Table 2), since the ratio of LRD depends in this case of its sub-policy (it is 1 if the sub-policy is SPT, whereas it is in $\Theta(n)$ if it does not have any sub-policy or if its sub-policy is LPT).

Policy	Path		Tree		Ring	
	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}
SPT	1	$2-\varepsilon < r < 2$	1	$2-\varepsilon < r < 2$	$1.34 < r \leq 2$	$2-\varepsilon < r < 3$
LPT	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
LRT	$\Theta(n)$	$2-\varepsilon < r \leq n$	$\Theta(n)$	$2-\varepsilon < r \leq n$	$\Theta(n)$	$2-\varepsilon < r \leq n$
LRD	$\Theta(n)$	1	$\Theta(n)$	$2-\varepsilon < r < 2$	$\Theta(n)$	$1.5-\varepsilon < r < 3$

Table 1: Summary of the results when the tasks have several destinations. Value ε is a small positive number which tends towards 0. The result for the maximum Completion Time Problem with the LPT policy in trees is obtained with a network whose size is exponentially large in n.

Policy	Path and Tree		Ring	
	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}
SPT	1	1	$1.17 < r \leq 2$	$1.66 < r \leq 2$
LPT, LRT	$\Theta(n)$	1	$\Theta(n)$	$1.16 < r \leq 2$
LRD	(see sub-policy)	1	(see sub-policy)	$r \leq 2$

Table 2: Summary of the results when the tasks have the same destination.

2 Paths and Trees: several destinations

Before starting the analysis of the different policies, let us give some general observations. We consider here out-trees where the source is at the root. Note, however, that considering general trees, where tasks can go through links in both sides, would not allow a task to decrease its completion time, since with the four above mentioned policies no task would have incentive to cross a link which does not lead to its destination, and then to take again this link in the other way to eventually be able to reach its destination.

Notice also that with the SPT, LPT, and LRD policies, since all the links use the same policy, and since there is a unique path from the source to a destination, the order of priority of the tasks in all the buffers will be the same for all the links (e.g. the largest task will always be the largest task, and the task which has the longest remaining distance to its destination will always be the same one). Thus, all the buffers (except the buffer at the source node) will schedule their tasks in a FIFO (first in first out) order.

With the LRT policy this is not always the case, as shown by the following example, where the order of priority of the tasks changes during the time (i.e., a task may "overtake" another task which left the source before). Consider the following instance: a path made of four nodes s, n_1, n_2, n_3, where s is the source of the tree, and the three following tasks (or packets): a task of length 3 (denoted by P_3) which wants to go to node n_3; a task of length 5 (denoted by P_5) which wants to go to node n_2; and a task of length 9 (denoted by P_9) which also wants to go to node n_2. At time 0, P_9 has the highest priority (its remaining time is 18), P_5 has the second highest priority (its remaining time is 10), and P_3 has the lowest priority (its remaining time is 9). Thus P_9 is scheduled first, followed by P_5 and P_3 in this order. As soon as P_9 arrives at node n_1, since it is the only task in the buffer, it is scheduled between nodes n_1 and n_2. While P_9 is scheduled between these two nodes (this takes 9 time units), tasks P_5 and P_3 arrive at node n_1 and wait that the link is free. At this time P_3 has the highest priority (its remaining time is 6, whereas the remaining time of P_5 is 5). Thus, P_3 will be scheduled before P_5 on edge (n_1, n_2), although it was scheduled after P_5 on edge (s, n_1).

2.1 The SPT policy

Theorem 2.1 *The approximation ratio of the SPT policy in a path or a tree, for the Maximum Completion Time problem, is smaller than 2.*

Proof: Let OPT be the completion time of the last task in an optimal solution of this problem. Let us consider any task i, and let us show that the completion time of this task, C_i, is smaller that $2\,OPT$. Since we are in a tree (or a path), for each couple (source, destination), there is only one possible path. Moreover, since the smaller a task is, the earlier it left the source; once a task is gone, it will not be caught up by another task and will not catch up with any task. Thus, the completion time of task i is equal to the time that i waits before its departure from the source, plus the time it needs to go to its destination. Let W_i denote the time before the departure of i from the source. We have: $C_i = W_i + (x_i\, l_i)$.
W_i is smaller than the sum of all the tasks (including task i) which go through the same outgoing arc as i from the source, and OPT has to be larger than or equal to this sum. Thus, $W_i < OPT$. Likewise, OPT has to be larger than or equal to the time needed by task i to reach its destination (without waiting time): $x_i\, l_i \leq OPT$. Hence, $C_i < 2\,OPT$, and then $C_{max} < 2\,OPT$. □

Theorem 2.2 *Let ε be any positive number. The approximation ratio of the SPT policy in a path or a tree, for the Maximum Completion Time problem, is larger than $2 - \varepsilon$.*

Proof: Let $0 < \varepsilon < 1$. Let us show that the approximation ratio is larger than $2 - \varepsilon$ by considering the following instance: we have a tree which is a single path of $n = \lceil \frac{2}{\varepsilon} \rceil$ arcs. The source is node 0 and each node $i < n$ has an outgoing arc towards node $i + 1$. There are $(n - 1)$ tasks of length $1 - 1/n$ and a task t of length 1. The destination of the tasks of length $1 - 1/n$ is node 1, and the destination of task t is node n. In the optimal solution, t is scheduled first and the maximum completion time is equal to $C_{max} = n$. If the policies of each arc are SPT, then task t will be scheduled at the last position, and its completion time will be its waiting time before its departure ($(n-1)(1-1/n)$) plus the time that it needs to reach its destination (n), that is $(n-1)(1-1/n)+n$. Thus, the approximation ratio is equal to $\frac{(n-1)(1-1/n)+n}{n} = 2-\frac{2}{n}+\frac{1}{n^2} > 2-\varepsilon$.□

We deduce from Theorems 2.1 and 2.2 the following corollary:

Corollary 2.1 *The approximation ratio of the SPT policy in a path or a tree, for the Maximum Completion Time problem, tends towards 2.*

Let us now show that the SPT policy is optimal for the Average Completion Time problem.

Theorem 2.3 *The approximation ratio of the SPT policy in a path or a tree, for the Average Completion Time problem, is* 1.

Proof: Since a path is also a tree, let us prove this in the case of a tree. Since the routings on each outgoing arc of the source are independent from each other, we consider each arc a from the source and show that the SPT policy minimizes the sum of completion times of the tasks which must go through this arc.

Let us assume w.l.o.g. that the tasks $1, 2, \ldots, n$ go through the arc a. Let us fix a policy $\mathcal{P}$ and let $W_i(\mathcal{P})$ denote the waiting time of task i before it leaves the source, i.e., it is the sum of the lengths of the tasks scheduled on the arc a before task i according to the policy $\mathcal{P}$. Since a task may need to wait during its travel because it is blocked by another task, in general the sum of the completion times can be larger than $\sum_{i=1}^{n} W_i(\mathcal{P}) + \sum_{i=1}^{n} x_i\, l_i$.

Observe, however, that for the SPT policy, after the packets left the source they do not block each other anymore, thus the sum of the completion times is $\sum_{i=1}^{n} W_i(SPT) + \sum_{i=1}^{n} x_i\, l_i$. Moreover, the SPT policy clearly minimizes the sum of the waiting times $\sum_{i=1}^{n} W_i(\mathcal{P})$. Thus the sum of the completion times of tasks crossing the arc a is minimized when using the SPT policy. □

2.2 The LPT policy

Theorem 2.4 *Let n be the number of tasks we have to route. The approximation ratio of the LPT policy in a path or a tree, for the Maximum Completion Time problem, is in $\Theta(n)$.*

Proof: Let us consider the following instance: we have n tasks, and a tree which is a single path of $m = 2^{n-1}$ arcs. The source is node 0 and each node $i < m$ has an outgoing arc towards node $i+1$. There are n tasks $\{1, \ldots, n\}$, and each task i has a length $1/2^{i-1}$, and its destination is node 2^{i-1}.
Let OPT be the maximum completion time in an optimal solution, and $C_{max}(SPT)$ (resp., $C_{max}(LPT)$) the maximum completion time if the policies of the arcs are SPT (resp., if the policies of the arcs are LPT). We have: $OPT \leq C_{max}(SPT)$. Let us show that $C_{max}(SPT) < 2$. We will then show that the maximum completion time when the policies are LPT is in $\Theta(n)$.
If the policies are SPT, then the only waiting time for each task is the waiting time before its departure from the source. The maximum waiting time is the one of the largest task (task 1):

it is $\sum_{i=2}^{n}(1/2^{i-1}) < 1$. We know that $C_{max}(SPT)$ is smaller than or equal to the maximum waiting time, plus the maximum time that a task needs to reach its destination (cf. Proof of Theorem 2.1). The maximum travel time is $\max_{1\leq i\leq n} l_i\, x_i = \max_{1\leq i\leq n}(2^{i-1})/2^{i-1} = 1$. Thus $C_{max}(SPT)$ is here smaller than 2.
In the case where the policies of the arcs are LPT, a task cannot overtake a task larger than it. Thus, when task i has arrived at its destination, task $i+1$ still has $2^{i-1}+1$ arcs to cross (it had 2^i arcs to cross and has already crossed $2^{i-1}-1$ arcs). This represents a travel time larger than $2^{i-1}\, l_{i+1} = 2^{i-1} \times (1/2^i) = 1/2$. Since task 1 reaches its destination at time 1, we have $C_{max}(LPT) > 1 + (1/2)\,(n-1)$. The approximation ratio of the LPT policy is then $\frac{C_{max}(LPT)}{OPT} > \frac{1+(1/2)\,(n-1)}{2} \geq \frac{1}{4} + \frac{n}{4}$.
We showed that the approximation ratio of LPT is in $\Omega(n)$. Let us show that it is in $O(n)$. Indeed, the solution obtained with this policy is not worse than the solution we would obtain by releasing tasks from the source in order of decreasing lengths and by releasing a task from the source only when the previous released tasks have reached their destinations. Since the travel time of each task (without waiting time) is smaller than or equal to OPT, the completion time of the last task in this solution is at most $n\,OPT$. Thus, the approximation ratio of the LPT policy is in $\Theta(n)$. □

Theorem 2.5 *Let n be the number of tasks we have to route. The approximation ratio of the LPT policy in a path or a tree, for the Average Completion Time problem, is in $\Theta(n)$.*

Proof: Let us consider the following instance: we have a tree which is a single path of two arcs, $n-1$ tasks $\{1,\ldots,n-1\}$ of length 1, and a task n of length n^2. The source of these tasks is node 0; it has an outgoing arc towards node 1, which is the destination of the small tasks; and node 1 has an outgoing arc towards node 2, which is the destination of task n. In the optimal solution, task n is scheduled after the small tasks, and the sum of the completion times is then $OPT = (\sum_{i=1}^{n-1} C_i) + C_n = n(n-1)/2 + (n-1+2\,n^2) = 5\,n^2/2 + n/2 - 1$. Let us now consider the case where the policy of the arcs is LPT: task n is scheduled first and the sum of the completion times is then $C_n + \sum_{i=1}^{n-1} C_i = 2\,n^2 + (n-1)\,n^2 + n(n-1)/2 = n^3 + 3\,n^2/2 - n/2$. The approximation ratio is $\frac{n^3+3\,n^2/2-n/2}{5\,n^2/2+n/2-1}$, which tends towards $2n/5$ when n gets large.

We showed that the approximation ratio of LPT is in $\Omega(n)$. Let us show that it is in $O(n)$. Indeed, the completion time of each task with this policy is not worse that the completion time we would have in the solution $\mathcal{S}$, where we release the tasks from the source in order of decreasing lengths, and we release a task from the source only when the previous released tasks have reached their destinations. Let us suppose that we have k tasks $\{1,\ldots,k\}$ of lengths $l_1 \leq \ldots, \leq l_k$, which take the same outgoing arc from the source. Since the travel time of each task i is equal to $l_i\, x_i$, the sum of the completion times in solution $\mathcal{S}$ is $\sum_{i=1}^{k} i\, l_i\, x_i$, whereas the sum of the completion times in an optimal solution is at least $\sum_{i=1}^{k} l_i\, x_i$. Each task is counted less than $k \leq n$ times more in the sum of the completion times in $\mathcal{S}$ rather than in an optimal solution. Since the completion time of each task scheduled with the LPT policies is smaller than or equal to the completion time of this task in $\mathcal{S}$, the approximation ratio of the LPT policy is in $\Theta(n)$. □

2.3 The LRD policy

2.3.1 Paths

Theorem 2.6 *The approximation ratio of the LRD policy for the Maximum Completion Time problem in a path is equal to 1.*

Proof: This result is a corollary of a result of [6]. In this paper, the authors study multicommodity flows in oriented graphs where each commodity has one source node and one sink node, and where the out degree of each node is at most 1, which includes the case of paths. Their goal is to have a quickest flow, i.e., to minimize the date at which all the flows arrive to their sinks. They give the following greedy algorithm: whenever there is a conflict between several commodities using the same arc, the algorithm gives top priority to the commodity which is the furthermost from its sink node. They prove that this algorithm is optimal.

The solution obtained with the LRD policy in our setting is a solution which could have been obtained with this multicommodity flow algorithm on the following instance: the path is the same as in our setting, all the commodities have the same source nodes (the source in our setting), and each commodity corresponds to a task (the sink node of the commodity corresponds to the destination of its corresponding task, and the flow between the source node and the sink node of each commodity is equal to the length of its corresponding task). Indeed, with the LRD policy, each arc schedules the task (the flow) whose remaining distance to its destination (sink) is the largest one. Thus the approximation ratio is 1 for the Maximum Completion Time problem in the path if the policy is LRD. □

2.3.2 Trees

Theorem 2.7 *The approximation ratio of the LRD policy in a tree, for the Maximum Completion Time problem, is smaller than 2.*

Proof: Let i be a task which has the largest completion time if the policy of the arcs is LRD, and let OPT be the maximum completion time in an optimal solution. Let us show that $C_i < 2\,OPT$. If i does not have any waiting time after its departure, then C_i is equal to the travel time of i, $x_i\, l_i \leq OPT$, plus the waiting time before its departure, $\sum_{j \in \mathcal{B}_i} l_j < OPT$ (the sum of the lengths of the tasks which go through the same arc is necessarily smaller than or equal to OPT), and thus $C_i < 2\,OPT$.
Let us now consider the case where task i need to wait after its departure. Recall that $\mathcal{B}_i$ is the set of tasks which went through the same arc than task i and which left the source before task i. Since i has to wait after it left the source, this means that it has caught up with a task of $\mathcal{B}_i$, and that there is in $\mathcal{B}_i$ a task which is larger than l_i (otherwise i would not have caught any task up). Let g be the largest task of $\mathcal{B}_i$. We are going to show that the time C_i that task i needs to reach its destination is smaller than or equal to $l_g\, x_i + \sum_{j \in \mathcal{B}_i \setminus g} l_j + l_i$. Indeed, the completion time C_i of i is maximized if every task of $\mathcal{B}_i$ has the same destination as i. This is equivalent to a routing in a path, and in that case LRD is an optimal policy according to Theorem 2.6. Notice now that, since there is a unique destination, the SPT policy is also an LRD policy, and is therefore optimal, too. So to calculate the completion time of i with the LDR policy, we can calculate the maximum completion time of tasks $i \cup \mathcal{B}_i$ with the SPT policy, since they are equal. This last quantity is equal to the waiting time of task g: $\sum_{j \in \mathcal{B}_i \setminus g} l_j + l_i$ plus its travel time $l_g\, x_i$.

The sum of the tasks which go through the same arc is smaller than or equal to OPT, and so $\sum_{j \in \mathcal{B}_i \setminus g} l_j + l_i < OPT$. Moreover, since g left the source before i with the LRD policy, we know that $x_g \geq x_i$, and so $l_g\, x_i \leq l_g\, x_g \leq OPT$. Hence, $C_i < 2\,OPT$, and the approximation ratio of the LRD policy is smaller than 2. □

Theorem 2.8 *Let ε be any fixed positive number. The approximation ratio of the LRD policy in a tree, for the Maximum Completion Time problem, is larger than $2 - \varepsilon$.*

Proof: Let $k = \lceil \frac{2}{\varepsilon} \rceil$. Note that this is a constant number, since ε is fixed. Let n be a positive integer such that n is a multiple of k, and $n >> k$. Let us consider the following

instance: a tree, as drawn in Figure 1, $n-(n/k)$ tasks $\{1,\ldots,(n-n/k)\}$ of length 1, and a task t of length n/k. In Figure 1 the source is S, and d_i is the destination of task i. For $i \in \{1,\ldots,(n-n/k)\}$, the distance between S and d_i is $k+1$, whereas the distance between S and d_t is k. In an optimal solution, task t is scheduled first and the maximum completion time, OPT, is equal to $n+k$ (it is the completion time of the last task of length 1). If the policies of the links are LRD, then task t is scheduled after the other tasks and the maximum completion time is $C_{max}(LRD) = 2\,n-(n/k)$. The approximation ratio of LRD is then $\frac{C_{max}(LRD)}{OPT} = \frac{2\,n-(n/k)}{n+k} = \frac{(2-1/k)\,(n+k)-(2-1/k)\,k}{n+k} = (2-\frac{1}{k}) - \frac{2\,k-1}{n+k}$.
Since $k = \lceil \frac{2}{\varepsilon} \rceil$, the approximation ratio of LRD is equal to $(2-\frac{1}{\lceil 2/\varepsilon \rceil}) - \frac{2\,\lceil 2/\varepsilon \rceil - 1}{n+\lceil 2/\varepsilon \rceil}$, which tends towards $2-\frac{1}{\lceil 2/\varepsilon \rceil} \geq 2-\frac{\varepsilon}{2} > 2-\varepsilon$ when n tends towards the infinity. □

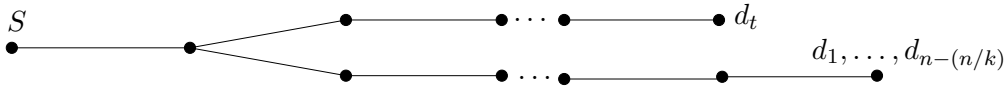

Figure 1: Example where the approximation ratio tends towards 2 for the Maximum Completion Time problem with the LRD policy.

We deduce from Theorems 2.7 and 2.8 the following corollary:

Corollary 2.2 *The approximation ratio of the LRD policy in a tree, for the Maximum Completion Time problem, tends towards 2.*

Let us now show that the average completion time is unbounded when the policy of every arc is LRD.

Theorem 2.9 *Let n be the number of tasks we have to route. The approximation ratio of the LRD policy in a path or a tree, for the Average Completion Time problem, is in $\Theta(n)$.*

The proof is the same as the one of Theorem 2.5.

2.4 The LRT policy

Theorem 2.10 *Let n be the number of tasks we have to route, and let ε be any small positive number. The approximation ratio of the LRT policy in a path or a tree, for the Maximum Completion Time problem, is larger than $2-\varepsilon$ and smaller than or equal to n.*

Proof: Let $0 < \varepsilon < 1$. Let us show that the approximation ratio is larger than $2-\varepsilon$ by considering the following instance: we have a tree which is a single path of $n = \lceil \frac{2}{\varepsilon} \rceil$ arcs. The source is node 0 and each node $i < n$ has an outgoing arc towards node $i+1$. There are two tasks: one of length 1, whose destination is node n, and one of length $n+\varepsilon$, whose destination is node 1. In the optimal solution, the small task is scheduled first and the maximum completion time is equal to $n+1+\varepsilon$. If the policies of each arc are LRT, then the large task is scheduled first and the maximum completion time is $2\,n+\varepsilon$. Thus, the approximation ratio is equal to $\frac{2\,n+\varepsilon}{n+1+\varepsilon} = 2-\frac{2+\varepsilon}{n+1+\varepsilon} > 2-\varepsilon$.

The solution obtained with the LRT policy is not worse than the solution we would obtain by releasing the tasks from the source in any order, and by releasing a task from the source only when the previous released tasks have reached their destinations. Since the travel time of each task (without waiting time) is smaller than or equal to OPT, the completion time of the last task in this solution is at most $n\,OPT$. Thus the approximation ratio of the LPT policy is at

most n. □

Note that it is an open question whether the approximation ratio of the LRT policy in a tree is in $\Theta(n)$ or not.

Theorem 2.11 *Let n be the number of tasks we have to route. The approximation ratio of the LRT policy in a path or a tree, for the Average Completion Time problem, is in $\Theta(n)$.*

The proof is the same as the one of Theorem 2.5.

3 Rings: several destinations

We consider here a ring with $m \geq 2$ nodes and in which there are between two neighbor nodes u and v an arc (u, v) and an arc (v, u). Thus, in this setting, at the source each task has two possible strategies: either it takes the right side of the ring, or it takes the left side of the ring. Both ways lead to its destination. We assume that each task knows the setting of the problem, i.e. the number of edges in the ring and the characteristics of all the tasks (their lengths and destinations), as well as the strategies chosen by the other tasks (the path that each task chooses to take to reach its destination). Since the aim of each task is to reach its destination as soon as possible, it will choose the strategy which, given the characteristics and strategies of the other tasks, will minimize its completion time.
We thus assume that tasks will converge towards a stable solution in which no task can decrease its completion time by unilaterally changing strategy. Such a situation is, by definition, a Nash equilibrium. We are interested in the price of anarchy of *pure* Nash equilibria.

With the policies we study (SPT, LPT, LRT, and LRD), no task has an incentive to cross an arc (u, v) and then to go back to u, the node where it comes from: once each task has chosen the direction it will take to its destination (i.e., go on the left side or the right side of the ring), then this task will not change direction. With the LRD policy, each arc schedules tasks in decreasing order of remaining distance to destination. The remaining distance of a task crossing an arc is equal to the number of arcs this task has to cross to arrive at its destination if it does not change direction.

Notice that the best strategy of a task does not always consist in choosing the shortest path to its destination. Consider, for example, a ring made of three links, linking the source s to nodes n_1 and n_2. Suppose that each link schedules first the longest tasks (its policy is LPT) and that there are two tasks of lengths 1 and 3, which both want to reach node n_1. The task of length 3 will take the link leading to n_1, where it will arrive three time units after its start. The task of length 1 does not have an incentive to take its shortest path, otherwise it will have to wait for the other task, which has the priority, and it would reach its destination at time 4. By taking path s, n_2, n_1, this task arrives earlier (at time 2) at its destination.

The price of anarchy of a given policy in a ring is larger than or equal to the approximation ratio of this policy in a path. Indeed, in the case where we have a very large ring and if the destinations of the tasks are, for example, in the beginning of the right side of the ring, then, in the optimal solution as well as in the Nash Equilibrium, all the tasks will choose the outgoing arc on the right of the source, and the ring can then be seen as a path (this path would be the path made of all the arcs of the ring except the arc from the source to the left side of the ring). Thus we can conclude that the price of anarchy is in $\Omega(n)$ for the Average Completion Time problem for the LPT, LRD and LRT policies (cf. Theorem 2.5). This price of anarchy is also in $O(n)$ because the travel time of each task is smaller than or equal to the maximum completion time in an optimal solution. Likewise, the price of anarchy is in $\Theta(n)$ for the Maximum Completion Time problem for the LPT policy (cf. Theorem 2.4), and it is larger than or equal to 2 for the Maximum Completion Time problem for the SPT and LRT policies (cf. Theorem 2.1 and 2.10)

3.1 The SPT policy

Theorem 3.1 *The price of anarchy of the SPT policy, for the Maximum Completion Time, problem is smaller than 3.*

Proof: Once a task is gone from the source, it cannot catch up a task which took the same direction before it, since the policy is SPT (the smallest task, which is the fastest to go through a link, is scheduled first). Thus a task will not have any waiting time after its departure. Let i be a task whose completion time is equal to the maximum completion time C_{max}, and let us show that C_{max} is smaller than $3\,OPT$, where OPT is the maximum completion time in an optimal solution. In the worst case, the waiting time of i is equal to the sum of the lengths of all the other tasks, which is smaller than $\sum_{j=1}^{n} l_j \leq 2\,OPT$ because in the optimal solution the amount of tasks scheduled on one of the two outgoing links from the source is at least $(\sum_{j=1}^{n} l_j)/2$ and so $OPT \geq (\sum_{j=1}^{n} l_j)/2$. Each task, knowing the strategies of the other tasks, chooses to go to its destination by the right side of the ring, or by the left side of the ring, and decides to take the direction which will minimize its completion time. If it takes its shortest path, of length x_i, then its travel time will be $x_i\, l_i \leq OPT$, and so its completion time will be smaller than $3\,OPT$. Since i wants to minimize its completion time it will choose to take the other side of the ring only if this does not increase its completion time, and so $C_{max} < 3\,OPT$. □

Theorem 3.2 *The price of anarchy of the SPT policy for the Average Completion Time problem is smaller than or equal to 2.*

Proof: Let us suppose that we have n tasks $T = \{1, 2, \ldots, n\}$ of lengths $l_1 \leq l_2 \leq \cdots \leq l_n$ to schedule. Let $\mathcal{O}$ be an optimal solution of these tasks for the Average Completion Time problem. Let (A, B) be a partition of tasks of T such that, in $\mathcal{O}$, the tasks which are in A are scheduled on the left outgoing arc from the source, and the tasks which are in B are scheduled on the right outgoing arc from the source. Let $\mathcal{S}$ be a Nash Equilibrium when the policies of the arcs are SPT. Let $\mathcal{S}'$ be the solution obtained when all the tasks take the same outgoing arc from the source (e.g., the one at the left of the source), and when the policies of the arcs are SPT. Let $W_i(\mathcal{S}')$ (resp., $W_i(\mathcal{O})$) be the time needed for task i to cross the first arc from the source in $\mathcal{S}'$ (resp., in $\mathcal{O}$). In $\mathcal{S}$, the completion time of each task i is $C_i(\mathcal{S}) \leq W_i(\mathcal{S}') + (x_i - 1)l_i$. Indeed in $\mathcal{S}$, each task i chooses the outgoing arc on which it will reduce its completion time, and by choosing the outgoing arc on its shortest path, i would have a completion time equal to the time needed to cross the first arc (i.e., its waiting time plus its length l_i), which is smaller than or equal to $W_i(\mathcal{S}')$, plus its travel time once it crossed the first arc, which is equal to $(x_i - 1)l_i$ (because once a task is gone it does not have any waiting time, since the policy is SPT).
In $\mathcal{O}$, the completion time of task i is equal to the time it needs to cross the first arc, $W_i(\mathcal{O})$, plus its remaining travel time to go to its destination, which is larger than or equal to $(x_i - 1)\, l_i$. Since $C_i(\mathcal{S}) \leq W_i(\mathcal{S}') + (x_i - 1)l_i$ and the completion time of i in $\mathcal{O}$ is $C_i(\mathcal{O}) \geq W_i(\mathcal{O}) + (x_i - 1)l_i$, if $\sum_{i=1}^{n} W_i(\mathcal{S}') \leq 2 \sum_{i=1}^{n} W_i(\mathcal{O})$, then we can deduce that the sum of the completion times in $\mathcal{S}$ is smaller than or equal to twice the sum of the completion times in $\mathcal{O}$. Let us now show that $\sum_{i=1}^{n} W_i(\mathcal{S}') \leq 2 \sum_{i=1}^{n} W_i(\mathcal{O})$.

The waiting time of task i in $\mathcal{S}'$ is equal to $l_1 + \cdots + l_{i-1}$, and so the time $W_i(\mathcal{S}')$ that task i needs to cross the first arc is $\sum_{j=1}^{i} l_j$. Therefore, the sum of the times needed to cross the first arc in $\mathcal{S}'$ is equal to $\sum_{i=1}^{n} W_i(\mathcal{S}') = \sum_{i=1}^{n} \sum_{j=1}^{i} l_j = n\, l_1 + (n-1)\, l_2 + \cdots + 2\, l_{n-1} + l_n$.

We saw that there are two queues A and B in $\mathcal{O}$. Let us see the smallest value that $s = \sum_{i=1}^{n} W_i(\mathcal{O})$ can take. This problem is equivalent to the scheduling problem $P2||\sum_j C_j$ for which we have two machines and we want to schedules jobs in order to minimize the sum of completion times. In [2] it has been proved that an LPT list scheduling algorithm gives the optimal

solution. More precisely, an optimal solution can be obtained by sorting the jobs in non decreasing lengths, and schedule each job in a round-and-robin way on each machine. Therefore, if n is even (resp., odd) the queue A will be composed of jobs $1, 3, \ldots, n-1$ (resp. $1, 3, \ldots, n-2, n$) and the queue B will be composed of jobs $2, 4, \ldots, n$ (resp., $2, 4, \ldots, n-1$), in this order. This way $l_{n-(2\,i)}$ and $l_{n-(2\,i+1)}$ will be counted $i+1$ times in s, whereas they are counted $(2i+1)$ and $(2i+2)$ times in $\sum_{i=1}^{n} W_i(\mathcal{S}')$. Thus, each task i is at most counted twice more in $\sum_{i=1}^{n} W_i(\mathcal{S}')$ than in the minimum value of $\sum_{i=1}^{n} W_i(\mathcal{O})$. So $\sum_{i=1}^{n} W_i(\mathcal{S}') \leq 2 \sum_{i=1}^{n} W_i(\mathcal{O})$ and then the sum of the completion times in $\mathcal{S}$ is smaller than or equal to twice the sum of the completion times in $\mathcal{O}$. □

Theorem 3.3 *The price of anarchy of the SPT policy for the Average Completion Time problem is larger than or equal to* $55/41 \approx 1.34$.

Proof: Let us consider the instance shown on Figure 2: we have a ring of length 10, and 9 tasks: 6 tasks $\{1, \ldots, 6\}$ of length 1, 2 tasks $\{7, 8\}$ of length $1-\varepsilon$, and one task 9 of length $1-2\,\varepsilon$, where ε is a small value. In the sum of the completion times we will neglect the ε, since we can fix ε as small as we wish.

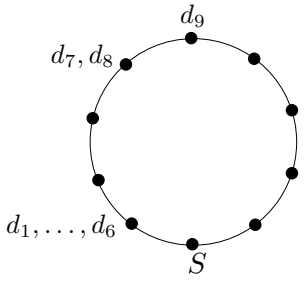

Figure 2: Example where the price of anarchy tends towards 55/41 for the Average Completion Time problem with the SPT policy. The source is S, and d_i is the destination of task i.

If all the tasks take the left side of the ring we have a Nash equilibrium and the sum of the completion times is 55. In the optimal solution, tasks $7, 8$ and 9 take the right side of the ring and the sum of the completion times is 41. Thus, the price of anarchy is at least 55/41 when the policies of the links are SPT. □

3.2 The LRD policy

Theorem 3.4 *The price of anarchy of the LRD policy, for the Maximum Completion Time problem, is smaller than 3.*

Proof: Let T be the set of tasks to be routed, and (X_1, X_2) a partition of T. Let (X_1, X_2, Pol) denote the solution obtained when the policy of each arc is Pol and when the tasks which choose to take the left outgoing arc from the source are in X_1 and tasks which choose to take the right outgoing arc from the source are in X_2. Let $C_{max}(X_1, X_2, Pol)$ denote the last completion time in this solution, and let $C_{max}(X, Pol)$ be the last completion time in this solution of a task belonging to the set X (where X is either X_1 or X_2) . Let $\mathcal{S}$ be a Nash equilibrium when the policies are LRD, and let A (resp., B) be the set of tasks which choose to take the left (resp. the right) outgoing arc from the source in $\mathcal{S}$.

Let us suppose w.l.o.g. that the source node is node 1, which has then two neighbors: node 2 at its left and node m at its right. Once tasks are partitioned into sets A and B, the routing

can be viewed as two parallel routings in two paths: one path starting at node 1 which has an outgoing arc towards node 2, which has an outgoing arc towards node 3, ..., which has an outgoing arc towards node m, and where tasks belonging to A are routed; and one path starting at node 1, which has an outgoing arc towards node m, ..., which ends at node 2, and where tasks belonging to B are routed. According to the Theorem 2.6, LRD is an optimal strategy in a path so, given an assignment of the tasks into two buffers left and right, LRD is not worse than the other policies: $C_{max}(A, B, LRD) \leq C_{max}(A, B, SPT)$. Let us now compare $C_{max}(A, B, SPT)$ to the maximum completion time OPT in an optimal solution. Let i be the task which has the largest completion time in (A, B, SPT), and let us suppose w.l.o.g. that i belongs to A. Its completion time is equal to $C_{max}(A, B, SPT) = W_i + l_i\, D$, where W_i is the waiting time before the departure of i from the source (one has $W_i < 2\, OPT$, see the proof of Theorem 3.1), and D is the number of arcs that i has to cross to arrive at its destination. If the path that i takes to go to its destination is its shortest path, then $l_i\, D \leq OPT$ and $C_{max}(A, B, LRD) \leq C_{max}(A, B, SPT) < 3\, OPT$. Otherwise the shortest path from the source to the destination of i is the path on the right side of the ring, and i would have taken its shortest path if it would have been in B. In this case its completion time would have been smaller than $3\, OPT$ (less than $2\, OPT$ of waiting time, and a travel time of at most OPT). Since we have a Nash equilibrium, i minimizes in (A, B, LRD) its completion time by going on the left side of the ring, and then this completion time is smaller than or equal to the completion time it would have by going in B, and so is smaller than $C_{max}(B, LRD) \leq C_{max}(B, SPT) < 3\, OPT$. □

Theorem 3.5 *The price of anarchy of the LRD policy, for the Maximum Completion Time problem, is larger than or equal to 1.5.*

Proof: Let us consider the instance shown in Figure 3: we have a ring of length $2\, n + 2$, and n tasks: $n - 1$ tasks $\{2, \ldots, n\}$ of length 1 and a task 1 of length $2\, n$. The destination of task i is the i^{th} node on the left from the source.

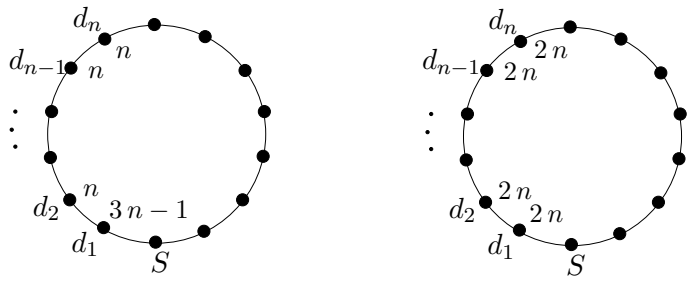

Figure 3: Example where the price of anarchy is 3/2 for Maximum Completion Time problem with the LRD policy. *Left*: Nash equilibrium. *Right*: Optimal solution. The source is S, and d_i is the destination of task i. There are $n - 1$ tasks $\{2, \ldots, n\}$ of length 1 and a task 1 of length $2\, n$. The number inside a ring at each node d_i is the completion time of task i in this ring.

In the Nash equilibrium shown on Figure 3 *Left*, all the tasks take the left side of the ring and the maximum completion time is $3\, n - 1$. In the optimal solution (shown on Figure 3 *Right*), task 1 takes the left side of the ring, and the other tasks the right side of the ring, and the maximum completion time is $2\, n$. Thus the price of anarchy when the policies of the links are LRD is $(3\, n - 1)/(2\, n)$, which tends towards 3/2 when n gets large. □

4 Case where there is only one destination

We are now interested in the price of anarchy (for rings), or approximation ratio (for trees) when all the tasks have the same source and the same destination. In this case the LRD policy does not make sense if it does not have a sub-policy to give a priority to tasks which have the same destination: any policy which does not introduce idle times is indeed an LRD policy. We will show that, in a ring, all these policies have a price of anarchy of at most two for the Maximum Completion Time problem. Likewise, since there is one single destination, if a task is larger than another, then it has a larger remaining travel time, and so the results for the LPT and LRT policies are the same.

4.1 Paths and Trees

The cases of paths and trees here are the same, since there is only one path between a source and a destination in a tree. For the Average Completion Time problem, since the SPT policy is optimal when there are several destinations, it is also optimal when there is only one destination. For the LPT and LRT policies, the proof of Theorem 2.5 can be used, slightly modified, in the sense where there is a very large task followed by many small tasks, and the path is a single arc: the approximation ratio is then in $\Theta(n)$. The approximation ratio of the LRD policy depends of its sub-policy: if it does not have a sub-policy then LRD can be the worst possible policy, e.g., LPT, and its approximation ratio is then in $\Theta(n)$ for this problem.

For the Maximum Completion Time problem, we know that the LRD policy is optimal in a path (see Section 2.3). Since there is a single destination, the SPT, LPT and LRT policies are also LRD policies and thus are optimal for this problem.

4.2 Rings

The upper bounds obtained for rings in the case where tasks can have multiple destinations are, of course, upper bounds in the case of a unique destination; thus, the price of anarchy for the Average Completion Time problem is at most 2 for the SPT policy. As we saw in Section 3, the approximation ratio of a policy in a path is a lower bound of the price of anarchy of this policy in a ring. Thus, the price of anarchy is in $\Omega(n)$ for the LPT and LRT policies for the Average Completion Time problem in a ring. Likewise, it is in $O(n)$ because the travel time of each task is smaller than or equal to the maximum completion time in an optimal solution.

If the ring has only two arcs ($m = 2$), then the Maximum Completion Time problem is a routing problem on two identical parallel links ($P2||C_{max}$). The only possible Nash equilibrium when the policy is LPT is equivalent to the solution obtained by a centralized LPT algorithm, which greedily schedules tasks from the largest one to the smallest one. This can easily be shown by contradiction: let us suppose that in a Nash equilibrium a task i starts later than a task j smaller than it ($l_i > l_j$). Since the policy of each arc is LPT, task i cannot be scheduled on the link on which task j is scheduled. By going via this link, task i would start earlier and thus decrease its completion time: task i has an incentive to change its strategy, and this situation is then not a Nash equilibrium. On the contrary, the solution obtained by a centralized LPT schedule is clearly a Nash equilibrium. Therefore, the approximation ratio of the LPT algorithm for ($P2||C_{max}$), which is 7/6 [5], is a lower bound of the price of anarchy of the LPT policy for the Maximum Completion Time problem.

Let us now prove the remaining bounds of the results in Table 1.3. The following theorem shows that the price of anarchy for the Maximum Completion Time problem is at most 2 for the SPT, LPT, LRT, and LRD policies.

Theorem 4.1 *The price of anarchy of any policy which does not introduce idle time, for the Maximum Completion Time problem, is smaller than or equal to 2.*

Proof: Let us consider that we have a ring of length $m \geq 2$, n tasks, a unique source and a unique destination. Let OPT be the maximum completion time in an optimal solution for our problem. Let (A, B) be a partition of the tasks such that, in the optimal solution, the tasks of A take the left path, of length L_a, and the tasks of B take the right path, of length L_b. Let us suppose that the length of the shortest path is $L = \min\{L_a, L_b\}$. Task n is the largest task, and let us suppose w.l.o.g. that task n belongs to A.

Observe that the problem of scheduling the tasks of A to their unique destination is a scheduling problem on a path, where LRD is optimal according to Theorem 2.6. Since the destination is unique, we can conclude that all the policies which do not introduce idle times are optimal, and their maximum completion times are the same as the one obtained with the SPT policy, that is the sum of all the tasks before task n plus the travel time of task n: $\sum_{i \in A \setminus n} l_i + l_n L_a \geq \sum_{i \in A \setminus n} l_i + l_n L$. We have, therefore, $OPT \geq \sum_{i \in A \setminus n} l_i + l_n L$. Moreover, we also know that $OPT \geq \sum_{i \in B} l_i$, since the tasks of B have at least one arc to cross.

We introduce the solution $\mathcal{S}$ (which is not necessarily a Nash equilibrium) in which all the tasks take the shortest path from the source to the destination. The maximum completion time for solution $\mathcal{S}$ is equal to $\sum_{i=1}^{n-1} l_i + l_n L$. Observe now that no Nash equilibrium $\mathcal{S}_{nash}$ is worse than the solution $\mathcal{S}$. The proof is by contradiction. If in solution $\mathcal{S}_{nash}$ a task has a completion time larger than $\sum_{i=1}^{n-1} l_i + l_n L$, then this task is necessarily scheduled on the path of the longest length, and it would decrease its completion time by choosing to go via the shortest path, meaning that $\mathcal{S}_{nash}$ is not a Nash equilibrium.

Thus, the maximum completion time in a Nash equilibrium is $\sum_{i=1}^{n-1} l_i + l_n L = (\sum_{i \in B} l_i) + (\sum_{i \in A \setminus n} l_i + l_n L) \leq 2\,OPT$. □

Theorem 4.2 *The price of anarchy of the SPT policy, for the Maximum Completion Time problem, is larger than or equal to 5/3.*

Proof: Let us consider the instance in which we have a ring of length 3, and 3 tasks: two tasks of length 1 and a task of length 3. The destination is the node at the right of the source: there are from the source to this node a path of length 2 and a path of length 1. The solution in which the three tasks take the path of length 1 is a Nash equilibrium, and the maximum completion time is 5. In an optimal solution, the tasks of length 1 take the path of length 2 and the task of length 3 take the path of length 1: the maximum completion time is then 3. Thus, the price of anarchy of the SPT policy for the Maximum Completion Time problem is larger than or equal to 5/3. □

Theorem 4.3 *The price of anarchy of the SPT policy, for the Average Completion Time problem, is larger than or equal to* $219/187 \approx 1.17$.

Proof: Let us consider the instance in which we have a ring of length 4, and 6 tasks: three tasks of length 1, a task of length 1.5, a task of length 2.25, and a task of length 3.375. The destination is the first node at the right of the source: there are from the source to this node a path of length 3, and a path of length 1. The solution in which all the tasks take the path of length 1 is a Nash equilibrium, and the sum of the completion times is 27.375. In an optimal solution, two tasks of length 1 take the path of length 3 and the other tasks take the path of length 1: the sum of the completion times is then 23.375. Thus, the price of anarchy of the SPT policy for the Average Completion Time problem is larger than or equal to $27.375/23.375 = 219/187$. □

5 Concluding remarks

We analyzed the worst case performance of decentralized policies which route tasks released from the same source in networks whose underlying topology is a path, a tree, or a ring. We showed significant differences between the four studied policies, since the use of some policies (for example, SPT) provides solutions which are in the worst case 2 or 3 times worse than the solutions obtained with an optimal centralized algorithm, whereas some other policies (for example, LPT) do not have such a bounded worst case ratio.

In the case of rings, we did not mention in this paper the convergence time needed in order to reach a Nash equilibrium. We assumed that the situation where tasks choose their destinations in a ring is a Nash equilibrium. However, this would not be realistic if the time to reach (or compute) a Nash equilibrium is too long. For the SPT and LPT policies, the tasks easily converge towards a Nash equilibrium: for the SPT policy, for example, the smallest task chooses its shortest path; given this, the smallest remaining task chooses the path which will minimize its completion time, and so forth. Each task, in increasing order of lengths, makes its choice according to the strategies of the other tasks. The convergence time towards a Nash equilibria in rings is an open question for the LRD and LRT policies.
We focused on the worst case quality of *pure* Nash equilibria. Note that it is possible to avoid mixed Nash equilibria (i.e. Nash equilibria in which the strategy of each task consists in choosing the side path of the ring with a probability $0 < p < 1$ and the right side of the ring otherwise), by fixing a negligible delay ε on one of the two outgoing links of the source (this means that there is an idle time of length ε before each task taking this link). This way the travel time of a task which takes the left side of the ring will never be the same as the travel time of this task if it takes the right side of the ring, and there will not exist Nash equilibria which are not pure.

We only considered paths, trees, and rings network topologies in this paper. In the future it would be interesting to study the price of anarchy induced by local policies in general graphs. The case where tasks are released from the same source and have to go to the same destination in general graphs would certainly be a good starting point in this direction.

References

[1] F.M. auf der Heide and B. Vöcking. A packet routing protocol for arbitrary networks. In *Proceedings of the 12th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 291–302, 1995.

[2] J. Bruno, E.G. Coffman Jr., and R. Sethi. Algorithms for minimizing mean flow time. In *Proceedings of IFIP Congress*, pages 504–510, 1974.

[3] G. Christodoulou and E. Koutsoupias. The price of anarchy of finite congestion games. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC)*, pages 67–73, 2005.

[4] G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. In *Proceedings of the 31st International Colloquium on Automata, Languages, and Programming (ICALP), LNCS 3142*, pages 345–357, 2004.

[5] R. Graham. Bounds on multiprocessor timing anomalies. *SIAM Jr. on Appl. Math*, 17(2):416–429, 1969.

[6] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science (short version in ICALP 2003)*, 379(3):387–404, 2007.

[7] N. Immorlica, L. Li, V.S. Mirrokni, and A. Schulz. Coordination mechanisms for selfish scheduling. In *Proceedings of the 1st Workshop on Internet and Network Economics (WINE), LNCS 3828*, pages 55–69, 2005.

[8] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 1563*, pages 404–413, 1999.

[9] F.T. Leighton, B.M. Maggs, and S.B. Rao. Packet routing and job-shop scheduling in o(congestion + dilatation) steps. *Combinatorica (preliminary version in FOCS 88)*, 14(2):167–186, 1994.

[10] F.T. Leighton, B.M. Maggs, and A.W. Richa. Fast algorithms for finding o(congestion + dilatation) packet routing schedules. *Combinatorica*, 19(3):375–401, 1999.

[11] R. Ostrovsky and Y. Rabani. Universal o(congestion + dilatation + $\log^{1+\varepsilon} n$) local control packet switching algorithms. In *Proceedings of the 29th ACM Symposium on Theory of Computing (STOC)*, pages 644–653, 1997.

[12] R. W. Rosenthal. A class of games possessing pure strategy nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[13] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

[14] D.B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. In *Proceedings of the 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 148–15, 1991.