

PERIODIC PACKET ROUTING ON TREES

BRITTA PEIS, SEBASTIAN STILLER, AND ANDREAS WIESE

ABSTRACT. In the periodic packet routing problem each of a set of tasks repeatedly emits packets over an infinite time horizon. These have to be routed along their fixed path through a common network. A schedule must resolve the resource conflicts on the arcs, such that the maximal delay any packet experiences can be bounded. We prove that such a schedule exists if and only if a simple to check criterion on the load of each arc is fulfilled. This even holds for the desirable class of template schedules. As in non-periodic packet routing we lay special emphasis on trees as underlying graphs. The scheduling policies themselves must be simple enough to be executed in real-time, i.e., with low computational overhead. We give algorithms to construct good edge-priority schedules and so-called template schedules by carefully balancing the delay among the packets. We can show that our template schedule guarantees a delay of less than $c(2 - (1/2)^{\text{diam}(G)/2})$ and our edge-priority schedule of at most $1.5c - 1$ (with c the maximal number of tasks using an arc). The latter is shown to be tight by an involved but insightful class of examples. Also for the template schedule we give a lower bound, as for a number of other positive results. All together this yields a fairly complete overview on period packet routing on trees. To compare the power of priority and template schedules we derive imitation theorems, i.e., we show that any bound achieved by a priority schedule on a specific instance can also (almost) be attained by a template schedule.

1. INTRODUCTION

Packet routing has become a standard model for data transfer. The basic model features a graph of servers as nodes and links as arcs. Each packet travels through the network along a simple path from its source node, i.e., its sender, to its sink node, i.e., its receiver. For the time a packet crosses an arc, it blocks this link (or part of its bandwidth) for other packages. Each server can store several packages at the same time. In our model we will assume even infinite node capacity. Further, we assume the paths of the packets to be given. Still, the resource conflicts on the arcs require synchronized scheduling decisions. Algorithms for these decisions are the central subject of this paper.

Usually, the data transfer between each pair of source and sink consists of a huge quantity of packets. Executing complicated, individual scheduling decisions for each of these packets would create an unacceptably high computational overhead. As typical example think of video streaming or a voice-over-IP connection.

For a suitable model we combine classical packet routing with the paradigm of real-time scheduling: In periodic packet routing one is given a graph and a set of tasks. Each task, for example a video streaming, is defined by a pair of source and sink node. Once the connection is established along a certain routing path, the task repeatedly, over a long time interval emits packets. Given several such tasks in the network, one needs a simple policy to decide at each arc which packet goes first.

This policy shall ensure a certain level of quality, i.e., that each packet is delayed by at most a certain, tolerably small time span.

As in classical real-time scheduling, we distinguish two task models, the strict periodic and the sporadic task model. A strict periodic task emits one new packet exactly every p time units. In the sporadic case the period p is only a lower bound on the separation time: a task can emit packets at any time, but the release dates of two packets of the same task are at least p time units apart. Strict periodic tasks model a steady demand of data transfer, e.g., in a video stream. For a voice-over-IP connection a sporadic task is a better model.

As the total duration of a connection is usually much larger than the transfer time for a single packet, we assume the time horizon of each task to be infinite. Note that a sporadic instance of the periodic packet routing problem consists of an infinite set of scenarios for each of which the policy must guarantee the desired level of quality. One of these scenarios is equivalent to the strict periodic instance with the same input data.

Firstly, we request as level of quality that the delay for every packet is finite. We prove a necessary and sufficient condition for this on arbitrary graphs. In the majority of this paper the graph is restricted to trees. For these, secondly, we give policies that achieve stronger levels of quality.

We say a tree is directed if its arcs have an orientation prescribing the direction in which packets can traverse them. In undirected and bi-directed trees arcs can be used in both directions. For the latter the capacities in both directions are independent while in the former case an arc cannot be used in both direction at the same point in time.

As stated before, the scheduling policies must be enough simple to be implementable. A widely used class of policies in classical real-time machine scheduling are fixed priority policies. Here each task has a (distinct) priority assigned. Whenever, jobs (in our case: packets) are in conflict, the first job of the task with highest priority is served first. Additional to this global priority policy, we discuss three more powerful variations. While a global priority schedule is based on a linear order of all tasks, a quasi global priority schedule is based on priority relation, that is not necessarily transitive. We will show that under certain, mild conditions such a policy is well-defined and advantageous. An edge priority schedule works like a global priority schedule except that the priority of the tasks can be different for each edge. Finally, we strengthen the power of edge priority schedules by allowing for an initial swing-in phase.

A more involved, but still simple class of schedules are template schedules. In a template schedule at each point in time an arc is open to transfer packets of exactly one task. This exclusive openness permutes cyclically over all tasks that send their packets along the arc. Initially, one has to find for each arc the order in which the tasks are permuted and the offsets between the cyclic permutations of all arcs. Executing a template schedule requires a global, discrete clock, while priority schedules (including global priority schedules) can be execute fully local. In a template schedule an arc may be idle although packets for which it is currently not open are waiting to traverse it. Nevertheless, we show that template schedules are provably more powerful than priority schedules.

1.1. Definitions. Let $G = (V, A)$ be a directed tree. Let T denote a set of tasks $\tau_i = (s_i, t_i)$ with $s_i, t_i \in V$ such that in G there is a directed path from s_i to t_i .

Let $p \in \mathbb{N}$ denote a period length. We call $I = (G, T, p)$ an instance of the *periodic packet routing problem* (PPRP). The intuition is that each task τ_i repeatedly creates new packets which have to be transported from s_i to t_i by a routing schedule. We assume unit transit times (i.e., each packet needs one timestep to traverse an arc), unit bandwidths (each arc can be used by at most one packet at a time) and unlimited storage in each vertex.

We denote by P_i the arcs on the path from s_i to t_i . Denote by $M_{i,j}$ the j -th packet created by task τ_i and by $\mathcal{M} = \{M_{i,j} | \tau_i \in T, j \in \mathbb{N}_0\}$ the set of all packets. We define by $\mathcal{M}_i := \{M_{i,j} | j \in \mathbb{N}_0\}$ the set of all packets created by τ_i . The release times of the packets are given by a map $r : \mathcal{M} \rightarrow \mathbb{N}_0$ which in general is not known a priori. However, we require that $r(M_{i,j+1}) \geq p + r(M_{i,j})$ for all tasks τ_i and all $j \in \mathbb{N}_0$. We call each map r satisfying this property a *realization*. We call I an instance of the *strict PPRP* if we allow only the (unique) realization which satisfies $r(M_{i,j+1}) = p + r(M_{i,j})$ and $r(M_{i,0}) = 0$ for all tasks τ_i and all $j \in \mathbb{N}_0$. Otherwise we call I an instance of the *sporadic PPRP*. Denote by \mathcal{R} the set of all realizations.

Our objective is to compute a schedule $S : \mathcal{R} \times \mathbb{N}_0 \times \mathcal{M} \rightarrow V$ with the following properties for all realizations r (we denote by $S_{r,t} : \mathcal{M} \rightarrow V$ the map $S(r, t, \bullet)$):

- For all tasks τ_i , all arcs $e = (u, v) \in P_i$, and all packets $M_{i,j} \in \mathcal{M}_i$ we have that $S(r, t, M_{i,j}) = u \Rightarrow S(r, t+1, M_{i,j}) \in \{u, v\}$ (unit transit times and packets follow path from s_i to t_i).
- For each arcs $e = (u, v) \in A$ and each $t \in \mathbb{N}_0$ we have that $|S_{r,t}^{-1}(u) \cap S_{r,t+1}^{-1}(v)| \leq 1$ (unit bandwidth).
- For all packets $M_{i,j}$ we have that $S(r, t, M_{i,j}) = s_i$ for all $t \leq r(M_{i,j})$ (release times are obeyed).

We say an arc $e = (u, v)$ is *used by a packet M at time t* if $S(r, t, M) = u$ and $S(r, t+1, M) = v$.

A directed graph $G = (V, A)$ is a *bidirected tree* if there is a directed tree $G' = (V, A')$ and $A = \{(u, v), (v, u) | (u, v) \in A'\}$. In this paper we also consider instances $I = (G, T)$ of the PPRP where G is a bidirected tree. Since a bidirected tree is strongly connected we no longer need to require explicitly for each task τ_i that there is a directed path from s_i to t_i . For the schedules we use exactly the same formal definition as above. Note that this implies that anti-parallel arcs can be used independently of each other.

When considering instances of the PPRP on *undirected trees* we assume joint capacity of each edge in both directions. Formally, for each edge $e = \{u, v\}$ and each timestep $t \in \mathbb{N}_0$ we require that $|S_{r,t}^{-1}(u) \cap S_{r,t+1}^{-1}(v)| + |S_{r,t}^{-1}(v) \cap S_{r,t+1}^{-1}(u)| \leq 1$. In particular, we assume that each edge can be used in both directions (and hence again we do not need to require explicitly the existence of a path from s_i to t_i for a task τ_i).

We say a *limit for a task τ_i* in a schedule S is a value k such that each packet which is ever created by τ_i needs at most k timesteps to reach t_i after it has been created. We denote by c the *congestion*, that is the maximum number of tasks which use an arc (or the maximum number of tasks which use an undirected edge in one direction). We say an instance I is *feasible* if there is a schedule for I such that for each task there is a finite limit. We will see in the sequel that an instance is feasible if and only if $c \leq p$. This holds also if we have an instance on a general graph where the path for each task is given explicitly. For a task τ_i we define D_i to be the length of P_i . For an arc/edge e let T_e denote the set of tasks which use e .

We call a schedule S a *direct* schedule if in S no packet waits in a vertex different from its start vertex (and its destination vertex). For ease of notation we say that a schedule is *undirect* if it is not necessarily direct.

We investigate three types of schedules, *template schedules*, *global-priority* and *edge-priority schedules*. We define them below. The problem can of course also be studied on general graphs (rather than only on trees). However, we will show in Section 9 that even on chain graphs (which have a quite simple structure) there are instances on which no global-priority schedule can guarantee a non-trivial bound for the travelling times for the packets of each task.

1.2. Template Schedules. One of the types of schedules studied in this paper are template schedules. Intuitively, for each arc e we have a pin-wheel with \bar{p} slices (for an integer $\bar{p} \leq p$). Each slice represents a congruence class of the timesteps modulo \bar{p} . There is a hand which points at a slice of the pin-wheel. At time t , the hand points at the $(t \bmod \bar{p})$ -th slice. Each slice of the wheel belongs to a task which uses e . Packets created by a task τ are only allowed to traverse e when the hand of the wheel points at a slice belonging to τ . Now we give a formal definition.

Definition 1.1 (Template Schedule). Let $I = (G, T, p)$ be an instance of the sporadic PPRP or the strict PPRP. A periodic schedule S for I is a *template schedule* if there exists an integer $\bar{p} \leq p$ and a map $task : E \times \{0, \dots, \bar{p} - 1\} \rightarrow T \cup \{undefined\}$ such that an arc $e = (u, v)$ is used at time t by a packet M created by a task τ if and only if

- (1) M is located on u at time t ,
- (2) $task(e, t \bmod \bar{p}) = \tau$, and
- (3) no packet created by τ before M is located on u .

Note that each map $task : E \times \{0, \dots, \bar{p} - 1\} \rightarrow T \cup \{undefined\}$ yields a template schedule by following the above definition. Also note that a template schedule might delay a packet even though there is no other packet waiting for using the next arc on its path. This is inevitable in a direct periodic schedule.

1.3. Global-priority and Edge-priority Schedules. The other types of schedules studied in this paper are global-priority and edge-priority schedules. In global-priority schedules there is a global ordering for the tasks which determines the priority of the tasks. The packets are then given priority according to this ordering. In edge-priority schedules we have the same paradigm. However, there we have an individual ordering of the tasks for each arc. Now we give formal definitions:

Definition 1.2 (Edge-priority schedule). Let $I = (G, T, p)$ be an instance of the sporadic or strict PPRP. A periodic schedule S for I is an *edge-priority schedule* if there exists a total order $\prec_e \subseteq T \times T$ for each arc e in G such that

- Whenever there are several packets waiting to use e , a packet moves first whose corresponding task τ has highest priority according to \prec_e . Formally, $\tau \preceq_e \tau'$ for any other task τ' that created a packet waiting to use e .
- If the task with highest priority according to \prec_e has created several packets which are waiting to use e then the packet moves first which was created first.

A global-priority schedule is simply an edge-priority schedule in which all arcs e have the same ordering \prec_e .

Definition 1.3 (Global-priority schedule). Let $I = (G, T, p)$ be an instance of the sporadic or strict PPRP. A periodic schedule S for I is a *global-priority schedule* if it is an edge-priority schedule and all total orders \prec_e for the arcs are identical.

1.4. Related Work. The packet routing problem in the non-periodic case is widely studied, e.g. [4, 9, 10, 11, 14]. Peis et al. [12] study the packet routing problem on trees. Busch et al. [2] present algorithms computing direct schedules, i.e., schedules which delay packets only in their start vertex. In a celebrated paper Leighton et al. [7] show that there is always a schedule of length $O(C + D)$ (where C denotes the maximum number of packets using an edge and D is the length of the longest path of a packet). In [8] Leighton et al. also present an algorithm which finds such schedules. This is extended to the periodic setting by Andrews et al. [1] guaranteeing a bound of $O(D_i + 1/r_i)$ for each session i with a packet injection rate of r_i (corresponding to a task with a period length in our notation). In particular, they introduce the template schedules which are also studied in this paper.

The task models are borrowed from classical real-time scheduling. Here one studies real-time executable algorithms for distributing and scheduling (computational) jobs on a processor platform. However, as there is no routing aspect techniques are quite different. For an overview cf. [3].

1.5. Our Contributions. The most basic question for an instance of PPRP is whether any schedule can ensure a finite upper bound for all packets.

► We derive a theorem (Section 2) that allows for a simple and general answer for *any graph*: such a schedule exists if and only if $c \leq p$. The affirmative part of the theorem rests on a strong insight into the behavior of template schedules. This allows to bound the maximal backlog of packets such a schedule produces at a vertex.

For the three different types of *trees* we give algorithms to efficiently construct priority and template schedules, and analyze the QoS these schedules guarantee. Hereby we further distinguish classes of schedules, e.g., between direct and indirect schedules. We match these results by lower bounds on the QoS any schedule of the prescribed type can achieve on the respective class of trees. We summarize these results in the tables below which draw a comprehensive landscape of what can be achieved for the PPR problem by the considered classes of schedules. Apart from the complete picture, three results stand out for themselves:

- For bidirected trees we give (Section 3.2) an algorithm to construct template schedules that guarantee a maximal delay for each packet of $2c - c(1/2)^{\lceil \text{diam}(G)/2 \rceil - 1} - 1$. This is achieved by carefully distributing the necessary delays among the tasks.
- Global-priority schedules are rather weak and easy to analyze. But it is natural to extend this class to edge-priority schedules, which can behave much more complex: Such schedules can balance the delay that tasks incur at different arcs. Thereby the maximal delay for the packets of all tasks is kept low. Under mild conditions on the congestion we can (Section 5) on the one hand construct such a carefully balanced priority schedule that achieves $1.5c - 1$ as maximal delay. On the other hand we give a non trivial construction that yields lower bound tightly matching the quality achieved by the algorithm.
- Finally, in Section 6 we follow a more direct approach to compare the power of template schedules and priority schedules. We show that whenever a priority schedule achieves a certain quality of service, one can construct a template schedule

Template Schedules				
	Indirect Schedules		Direct Schedules	
	Limit	Bound on c	Limit	Bound on c
Directed Trees	$c + D_i - 1$	p	$c + D_i - 1$	p
Bidirected Trees	$2c - c(1/2)^{\lceil \text{diam}(G)/2 \rceil - 1} + D_i - 1$	p	$2c + D_i - 2$	$p/2$
Undirected Trees	$4c - 2c(1/2)^{\lceil \text{diam}(G)/2 \rceil - 1} + D_i - 1$	$p/2$	$4c + D_i - 3$	$p/4$

TABLE 1. Overview of our results for template schedules.

Priority Schedules				
	Type	Limit	Bound on c	Lower Bound
Bidirected Trees	Global	$2c + D_i - 1$	$p/3$	–
Directed Trees	Global	–	–	$2c + D_i - 1$
Directed Trees	Edge	$1.5c + D_i - 1$	$2p/5$	$1.5c + D_i - 1$

TABLE 2. Overview of our results for priority schedules. All bounds hold for both the strict periodic and the sporadic setting.

imitating the priority schedule well enough to achieve the same or almost the same quality. Key to these results is to prove that priority schedules after some time show a periodic behavior.

To round up the picture on PPRP we also state minor results for which we defer the proofs to the Appendix. Among these are complexity results and results on two other ways to strengthen the concept of priority schedules: First, we allow the priority relation to be not necessarily transitive. Under certain conditions this gives a well-defined and fruitful class of so called quasi-priority schedules. Secondly, we briefly mention the effect of an initial swing-in period in which the schedule can deviate from its priority rules.

2. NECESSARY BOUND ON CONGESTION

In this section we prove that an instance of the periodic packet routing problem has a schedule with a finite limit for each task if and only if $c \leq p$. Hence, for the instances studied in the remainder of this paper we will always require that $c \leq p$. Note that this result does not only hold on instances on trees but also for instances on general graphs where the paths of the tasks are given as part of the input.

Theorem 2.1. *An instance I of the strict or sporadic periodic packet routing problem is feasible if and only if $c \leq p$.*

Before we can prove the theorem we need some preparation. We say that a template schedule is *cheap* if for every arc e and every task τ there is at most one value k such that $\text{task}(e, k) = \tau$. In order to simplify the analysis we consider only cheap template schedules for the remainder of this subsection. Before we can prove Theorem 2.1 we prove some properties of cheap schedules.

Lemma 2.2. *Let S be a cheap template schedule for a strict or sporadic periodic packet routing instance I . Assume that for each arc e we have $\{\text{task}(e, k) \mid 0 \leq k < \bar{p}\} = T_e$. Let τ_i be a task. For each timestep t and each arc $e = (u, v) \in P_i$ it holds that on u there is at most one packet created by τ_i .*

Proof. Assume on the contrary that at timestep t there are two packets M and M' created by τ_i which are located on a vertex u waiting to use an arc $e = (u, v) \in P_i$. Also assume that this has not occurred in S before timestep t . W.l.o.g. assume that M arrived on u before M' and assume that M arrived on u at time t' (with $t' \leq t$). Since S is a cheap schedule it follows that $t' + \bar{p} = t$. Since $\tau_i \in T_e = \{\text{task}(e, k) \mid 0 \leq k < \bar{p}\}$ there must be a time t'' with $t' \leq t'' < t$ such that $\text{task}(e, t'' \bmod \bar{p}) = \tau_i$. But this is a contradiction since M is still located on u at time t . \square

In the following lemma we prove a universal limit for cheap template schedules.

Lemma 2.3. *Let S be a cheap template schedule for a strict or sporadic periodic packet routing instance I . Assume that for each arc e we have $\{\text{task}(e, k) \mid 0 \leq k < \bar{p}\} = T_e$. Then, for each task τ_i it holds that $D_i \cdot \bar{p}$ is a valid limit.*

Proof. From Lemma 2.2 it follows that there can be no two packets created by the same task τ_i waiting for an arc $e \in P_i$. From the definition of template schedules it follows that a packet M has to wait at most $\bar{p} - 1$ timesteps before it can be transferred over the next arc on its path. Thus, each packet created by τ_i needs at most $D_i \cdot \bar{p}$ steps to reach t_i . \square

Now we prove that $c \leq p$ is a necessary and sufficient condition for the existence of a periodic routing schedule.

Proof of Theorem 2.1. First assume on the contrary that I is feasible but there is an arc $e = (u, v)$ which is used by more than p tasks. Since I is feasible there is a limit k_i for each task τ_i . We define $k := 1 + \max_i k_i$.

Assume that for each $k \in \mathbb{N}_0$ each task τ_i creates a new packet at timestep $k \cdot p$. Hence, during the time interval $L = [0, p)$ each task $\tau_i \in T_e$ has created one packet. This implies that in L all tasks in T_e together have created $c > p$ packets. Since c and p are integral, this implies that $c \geq p + 1$. Since at most p packets can be transferred over e in p timesteps, there is at least one packet which has been created within L which has not been transferred over e . Within the time interval $L_k = [0, k \cdot p)$ each task $\tau_i \in T_e$ has created k packets. This implies that within L_k all tasks in T_e together have created $k \cdot c \geq k \cdot (p + 1)$ packets. However, at most $k \cdot p$ packets could possibly have used e . That means that there are at least k packets in the network which still need to use the arc e . Thus, there is one of those packets $M_{i,j}$ created by a task τ_i which does not reach v before time $t = k \cdot p + k$. Let $c_{i,j}$ be the time when $M_{i,j}$ was created. Since $c_{i,j} \leq k \cdot p$ we conclude that $k \cdot p + k \geq c_{i,j} + k > c_{i,j} + k_i$ and thus $M_{i,j}$ reaches t_i after strictly more than k_i timesteps. Thus, k_i is not a valid limit for τ_i which is a contradiction.

Now assume that $c \leq p$. We create a template schedule as follows: First, we define $\bar{p} := p$. Now let e be an arc. W.l.o.g. assume that $T_e = \{\tau_0, \tau_1, \dots, \tau_{|T_e|-1}\}$. Note that $|T_e| \leq c \leq p$. We define $\text{task}(e, k) := \tau_k$ for $0 \leq k < |T_e|$ and $\text{task}(e, k') := \text{undefined}$ for $|T_e| \leq k' < \bar{p} = p$. We do this procedure for each arc e . Then, for each arc e we have $\{\text{task}(e, k) \mid 0 \leq k < \bar{p}\} = T_e$ since $|T_e| \leq \bar{p} = p$. Denote by S the (cheap) template schedule resulting from task . From Lemma 2.3 it follows that in S each task has a finite limit. Thus, I is feasible. \square

3. TEMPLATE SCHEDULES

In this section we study template schedules on directed trees, bidirected trees, and undirected trees. Moreover, we distinguish between direct and not necessarily direct (undirect) schedules. For each algorithm we prove the limit that it guarantees for each task. Note that for each algorithm we require an upper bound on the congestion c . Our algorithms are summarized in Table 1.

We showed in Theorem 2.1 that an instance is feasible if and only if $c \leq p$. For some of our algorithms we require a smaller bound on c , like $p/2$ or $p/4$. This raises the question whether one can still obtain schedules with similar limits if c is larger (though still bounded by p). We answer this question by giving examples for the respective settings with higher congestion where no template schedule can achieve the limits guaranteed by our algorithms. For the case of direct schedules we even give counterexamples where there can be no direct template schedule at all.

3.1. Directed Trees. In the sequel we will study template schedules on directed trees, bidirected trees and undirected trees. We start with directed trees, presenting a general technique which we will adapt later for the other two tree classes. Note that the case of directed trees is a special case of bidirected trees.

Let $I = (G, T, p)$ be an instance of the sporadic periodic packet routing instance on a directed tree G . We present an algorithm which constructs a template schedule which guarantees a limit of $c + D_i - 1$ for each task τ_i . This bound is best possible: There are instances where there can be no better limit for every task (e.g., consider an instance on a directed path in which all tasks have identical paths). Also, we will show in Section 8 that it is *NP*-hard to determine whether there is a template schedule which guarantees a limit of $c + D_i - 2$ for an instance on a directed tree. Our algorithm transfers ideas from [12] to the periodic setting.

Algorithm *DTREE*(I)

- (1) Find feasible path-coloring $f : T \rightarrow \{0, \dots, c - 1\}$ for paths of tasks.
- (2) Define time-dependent edge-coloring $g : E \times \{0, \dots, c - 1\} \rightarrow \{0, \dots, c - 1\}$ with consecutive property (the latter ensures limit and direct routing).
- (3) Define map *task* from path-coloring and time-dependent edge-coloring.

Now we present the algorithm. We compute a *feasible path-coloring* for the paths of the tasks. I.e., we compute a map $f : T \rightarrow \{0, \dots, c - 1\}$ such that if the paths of two tasks τ_i, τ_j share an arc then $f(\tau_i) \neq f(\tau_j)$. Such a coloring can be obtained by first solving the problem for the subgraphs induced by each vertex together with its neighbors. This can be reduced to edge-coloring on bipartite multigraphs. Then, the solutions for the subproblems can be combined to a global coloring. The whole procedure can be accomplished in polynomial time, for details we refer to [12, Section 2.1].

Then we compute a time-dependent edge-coloring $g : E \times \{0, \dots, c - 1\} \rightarrow \{0, \dots, c - 1\}$ as follows: We start with an arbitrary arc e^* and define its coloring $g(e^*, i) := i$ for $0 \leq i < c$. Then, we define the coloring of the remaining arcs such that the *consecutive property* holds:

- For two consecutive arcs $e = (u, v)$ and $e' = (v, w)$ we require that $g(e, i) = g(e', (i + 1) \bmod c)$ for $0 \leq i < c$.

- For two adjacent arcs $e = (u, v)$ and $e' = (u, v')$ (or $e = (u, v)$ and $e' = (u', v)$) we require that $g(e, i) = g(e', i)$ for $0 \leq i < c$.

Note that after having defined the coloring of e^* the consecutive property implies the coloring of the other arcs. From the colorings f and g we compute the map *task*: We define $\bar{p} := c$. Let e be an arc and let $k \in \{0, \dots, c-1\}$. If there is a task $\tau_i \in T_e$ such that $f(\tau_i) = g(e, k)$ we define $\text{task}(e, k) := \tau_i$. Since f is a valid path coloring there can be at most one such task. If $g(e, k) \notin f(T_e)$ we define $\text{task}(e, k) := \text{undefined}$. We do this for all arcs e and all timesteps t . Denote by $DTREE(I)$ the resulting schedule.

Theorem 3.1. *Let I be an instance of the sporadic periodic packet routing problem on a directed tree with $c \leq p$. The schedule $DTREE(I)$ is a direct template schedule which guarantees a limit of $c + D_i - 1$ for each tasks τ_i .*

Proof. The (first) consecutive property of g is passed on to *task*: For two consecutive arcs $e = (u, v)$ and $e' = (v, w)$ we have that $\text{task}(e, k) = \text{task}(e', k + 1 \bmod c)$ for all k . Therefore, once a packet has left its start vertex it is never delayed until it reaches its destination vertex. Each packet has to wait for at most $c - 1$ timesteps in its start vertex. We conclude that for each task τ_i it holds that $c + D_i - 1$ is a valid limit. \square

3.2. Bidirected Trees. In this section, we show how we can adapt the technique introduced for template schedules on directed trees for (direct and undirect) template schedules on bidirected trees. First, we describe the schedule $BTREE(I)$ which is undirect and guarantees a limit of $2c - c \left(\frac{1}{2}\right)^{\lceil \text{diam}(G)/2 \rceil - 1} + D_i - 1$ ($\text{diam}(G)$ denotes the diameter of G). Then, we show that at least for the case $c = 2$ this limit cannot be improved by giving a suitable example instance. However, we give a randomized algorithm which guarantees a limit of $1.5c + D_i - 1$ *in expectation* for each task τ_i .

Then we discuss direct schedules on bidirected trees. We show that one needs to require a certain bound on $\frac{c}{p}$ because there is an instance with $c = \frac{3}{4}p$ for which there exists no direct schedule. However, we prove that given an instance with $c \leq \frac{p}{2}$ we can find a direct schedule $BTREE_{dir}(I)$ which guarantees a limit of $2c + D_i - 2$.

First, we introduce some structure on the tree which we will use for all algorithms in the sequel.

Definition 3.2 (Tree-structure). Let G be a directed or bidirected tree. We define an arbitrary vertex v_r to be the root vertex. We call an arc e an *up*-arc if it is oriented towards v_r and a *down*-arc if it is oriented away from v_r . For a vertex v let $h(v)$ be the distance between v_r and v . For each task τ_i we define the vertex v_i which is closest to v_r to be the *peak vertex* of τ_i . For a task τ_i we define its *height* $h(\tau_i)$ by $h(\tau_i) := h(v_i)$. We say a packet *moves up* if it is using an up-arc. A packet *moves down* if it uses a down-arc.

Now we describe our undirect schedule $BTREE(I)$ which guarantees a limit of $2c - c \left(\frac{1}{2}\right)^{\lceil \text{diam}(G)/2 \rceil - 1} + D_i - 1$. Intuitively, the schedule works as follows: on the way towards the root, the packets of each task are delayed only in their start vertex. This results in a delay of at most $c - 1$ in the start vertex. The delay of a task τ_i on a down-arc e depends on the height e and on whether e is the first down-arc of τ_i .

If e is the first down-arc on P_i then τ_i is delayed up to $c \left(1 - \left(\frac{1}{2}\right)^{h(e)}\right)$ times before it can use e . If e is not the first down-arc on P_i then τ_i is delayed at most $c \left(\frac{1}{2}\right)^{h(e)}$ times before it can use e . In total, on its way down a task τ_i can be delayed only up to $c \left(1 - \left(\frac{1}{2}\right)^{h(e)} + \sum_{k=h(e)+1}^{\ell} \left(\frac{1}{2}\right)^k\right) = c \cdot \sum_{k=1}^{\ell} \left(\frac{1}{2}\right)^k \leq c \cdot \left(1 - \left(\frac{1}{2}\right)^{h(G)}\right)$ times, assuming that e is the first down-arc on P_i and P_i has $\ell + 1$ down-arcs in total (we denote by $h(G)$ the maximum height of a vertex in G). By choosing the root v_r such that $h(G) \leq \lceil \text{diam}(G)/2 \rceil$ we achieve the bound on the maximum delay stated above.

Algorithm *BTREE(I)*

- (1) Define values for *task* from top to bottom. Start with root vertex v_r such that in v_r no task is delayed, like in *DTREE(I)*
- (2) On an intermediate vertex v
 - (a) Tasks moving up from v are not delayed on v
 - (b) Tasks moving down from v with peak vertex above v are delayed at most $c \left(\frac{1}{2}\right)^{h(v)}$ times
 - (c) Tasks with peak vertex v are delayed at most $c \left(1 - \left(\frac{1}{2}\right)^{h(v)}\right)$ times on v
- (3) Gives at most $2c - c \left(1 - \left(\frac{1}{2}\right)^{h(G)-1}\right) - 1$ delays for each task (with geometric sequence)

Now we describe the algorithm in detail. We choose the root v_r such that $h(G) \leq \lceil \text{diam}(G)/2 \rceil$. For our template schedule we define $\bar{p} := c$. We define the values of the map $\text{task}(e, k)$ for all arcs e adjacent to v_r such that no task is delayed in v_r . The problem of finding such values can be reduced to finding a direct schedule in a directed tree, see Section 3.1.

We can assume by induction that for all arcs e adjacent to a vertex v with $h(v) \leq n$ the respective values of the map $\text{task}(e, k)$ have been defined. Now consider a vertex v with $h(v) = n + 1$. Denote by G_v the subtree rooted at v . Let e_v^\uparrow and e_v^\downarrow be the two arcs on the two directed paths between v_r and v which are adjacent to v . Denote by $T_v^{(up)}$ the tasks whose path uses v and e_v^\uparrow and by $T_v^{(down)}$ the tasks whose path uses v and e_v^\downarrow . Let $T_v^{(peak)}$ be the tasks for which v is the peak vertex. For a down-arc $e = (u, v)$ we define the height $h(e)$ by $h(e) := h(u)$.

Due to the induction we have already defined offsets $o_i, o_j \in \{0, \dots, c - 1\}$ for each task $\tau_i \in T_v^{(down)}$ and $\tau_j \in T_v^{(up)}$ such that $\text{task}(e_v^\uparrow, o_i) = \tau_i$ and $\text{task}(e_v^\downarrow, o_j) = \tau_j$. Assume these offsets are fixed, we need to determine offsets for the tasks $T_v^{(up)} \cup T_v^{(down)} \cup T_v^{(peak)}$ to use the child arcs of v .

We do this as follows: ignoring the tasks $T_v^{(down)}$ for a moment, for the tasks in $T_v^{(up)} \cup T_v^{(peak)}$ we define a schedule task' such that none of them is delayed on v . Note that this problem can be reduced to finding a direct schedule on a directed tree, see Section 3.2. By some permutation one can ensure that $\text{task}'(e, k) = \text{task}(e, k)$ for all child arcs e of v and e_v^\uparrow and all $k \in \{0, \dots, c - 1\}$.

For all up-arcs e which are adjacent to v we define $\text{task}(e, k) := \text{task}'(e, k)$ for all $k \in \{0, \dots, c - 1\}$. Now we need to define the values $\text{task}(e, k)$ for down-arcs e which are adjacent to v . In general, for the tasks which use these arcs we need to

define some delay on v . Let $e = (v, w)$ be a down-arc. Let $T_e^{(down)}$ denote all tasks which use e and which also use e_v^\downarrow . Let $T_e^{(peak)}$ denote all tasks which use e and for which v is their peak vertex. There are two cases: If $|T_e^{(down)}| \geq c \left(1 - \left(\frac{1}{2}\right)^{h(v)}\right)$ then we define the map *task* such that the tasks $T_e^{(peak)}$ are not delayed in v . Since $|T_e^{(peak)}| < c \left(\frac{1}{2}\right)^{h(v)}$ this can be achieved by delaying each task $\tau \in T_e^{(down)}$ at most $c \left(\frac{1}{2}\right)^{h(v)}$ times. Similarly, if $|T_e^{(down)}| < c \left(1 - \left(\frac{1}{2}\right)^{h(v)}\right)$ we define *task* such that tasks $T_e^{(down)}$ are not delayed in v and each task $\tau \in T_e^{(peak)}$ is delayed at most $c \left(1 - \left(\frac{1}{2}\right)^{h(v)}\right)$ times. Denote by $BTREE(I)$ the schedule resulting from the map *task*.

Theorem 3.3. *Let I be an instance of the sporadic PPRP on a bidirected tree with $c \leq p$. The schedule $BTREE(I)$ is a template schedule which guarantees a limit of $2c - c \left(\frac{1}{2}\right)^{\lceil diam(G)/2 \rceil - 1} + D_i - 1$ for each task τ_i .*

Proof. Let τ_i be a task with peak vertex v_i . Let M_i be a packet created by τ_i . From the definition of *task* it follows that on its way up M_i is delayed only in its start vertex (at most $\bar{p} - 1 = c - 1$ times). On v_i the packet M_i is delayed at most $c \left(1 - \left(\frac{1}{2}\right)^{h(v_i)}\right)$ times. Now let $e = (v, w)$ be a down-arc on P_i which is not adjacent to v_i (i.e., $v \neq v_i$). By construction, M_i is delayed at most $c \left(\frac{1}{2}\right)^{h(v)}$ times on v . Denote by P_i^\downarrow all vertices on the way down of τ_i , excluding v_i and t_i . Note that $h(v) \leq h(G) - 1$ for each vertex $v \in P_i^\downarrow$. We calculate that in total M_i is delayed at most

$$\begin{aligned} c \left(1 - \left(\frac{1}{2}\right)^{h(v_i)}\right) + \sum_{v \in P_i^\downarrow} c \left(\frac{1}{2}\right)^{h(v)} &= c \left(\sum_{k=1}^{h(t_i)-1} \left(\frac{1}{2}\right)^k \right) \\ &\leq c \left(\sum_{k=1}^{h(G)-1} \left(\frac{1}{2}\right)^k \right) \\ &= c \left(1 - \left(\frac{1}{2}\right)^{h(G)-1}\right) \\ &= c \left(1 - \left(\frac{1}{2}\right)^{\lceil diam(G)/2 \rceil - 1}\right) \end{aligned}$$

times. This proves a limit of $2c - c \left(\frac{1}{2}\right)^{\lceil diam(G)/2 \rceil - 1} + D_i - 1$ for each packet created by τ_i . \square

For the case $c = 2$ and $diam(G) \geq 3$ our analysis of $BTREE(I)$ guarantees a bound of $2c + D_i - 2$. Now we show that this is indeed best possible.

Proposition 3.4. *There is an instance I of the strict PPRP on a bidirected tree with $c = p = 2$ such that in any template schedule there is a task τ_i such that the packets created by τ_i reach t_i after at least $2c + D_i - 2$ steps.*

Proof. We say a task τ is *delayed* by a schedule if the packets created by τ are delayed. In our construction we first ensure that there is one task which is delayed

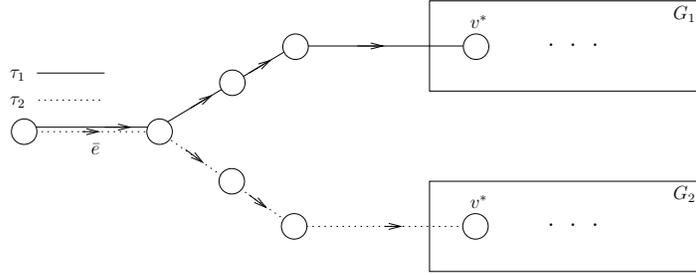


FIGURE 3.1. The graph used for the proof of Proposition 3.4. The boxes G_1 and G_2 denote the two copies of the gadget G shown in Figure 3.2.

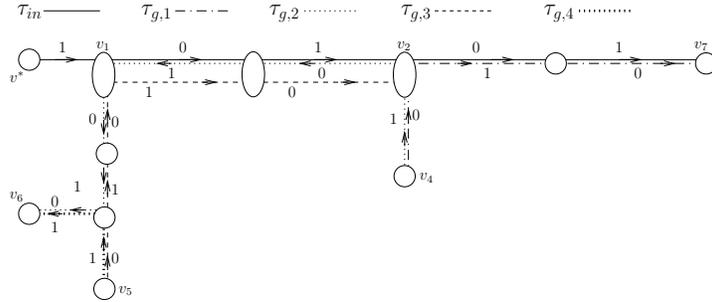


FIGURE 3.2. The gadget G used in the proof of Proposition 3.4. The numbers 0 and 1 represent whether the tasks use the corresponding arcs in even or odd timesteps, respectively.

at least once: Define two tasks τ_1 and τ_2 such that $s_1 = s_2$ and P_1 and P_2 share the first arc \bar{e} on their path. Starting with this setup, we let P_1 continue in a gadget G_1 and P_2 in a gadget G_2 . See Figure 3.1 for a sketch. The gadgets are defined below. We say the tasks τ_1 and τ_2 are the *incoming tasks* of the respective gadgets. The gadgets G_1 and G_2 are identical and ensure the following: If the incoming task τ was delayed before entering the gadget, then either τ is delayed again, or another task which is defined inside the gadget is delayed twice. This ensures that in the overall instance there is a task which is delayed at least twice and thus its packets reach their destination after at least $2c + D_i - 2$ steps.

In our construction an incoming task τ arrives on the first vertex v^* inside the gadget on even timesteps if it was delayed before entering the gadget. Now we describe the gadgets G_i . Their graph is depicted in Figure 3.2. Let τ_{in} denote the incoming task of G_i . We introduce the tasks $\tau_{g,1}, \tau_{g,2}, \tau_{g,3}$ and $\tau_{g,4}$. The paths of these tasks are shown in Figure 3.2. Assume that τ_{in} was delayed once before entering the gadget. We show that then there is at least one task which is delayed twice. We assume on the contrary that there is no task which is delayed twice or more. Since τ_{in} was delayed once $\tau_{g,1}$ is never delayed (if it was delayed before reaching v_2 then it would be delayed again at v_2 by τ_{in}). Thus, $\tau_{g,2}$ is delayed once (by $\tau_{g,1}$). Since τ_{in} is not delayed inside G_i we conclude that $\tau_{g,3}$ is not delayed

either. However, this implies that $\tau_{g,4}$ is delayed by $\tau_{g,3}$ and $\tau_{g,2}$ which yields a contradiction. \square

Now we present a randomized algorithm for periodic packet routing on a bidirected tree. It guarantees a limit of $c + D_i - 1$ in expectation for each task for the strict PPRP and a limit of $1.5c + D_i - 1$ in expectation for each task τ_i for the sporadic PPRP.

Algorithm *RTREE*(I)

- (1) Split instance I into instance I_{up} (parts of the paths towards v_r) and instance I_{down} (parts of the paths away from v_r).
- (2) Compute maps $task_{up}$ for *DTREE*(I_{up}) and $task_{down}$ for *DTREE*(I_{down}).
- (3) Choose offsets for $task_{up}$ and $task_{down}$ uniformly at random.
- (4) Combine resulting maps to schedule.

Let I be an instance of the strict or sporadic PPRP on a bidirected tree with $c \leq p$. We denote by G_{up} and G_{down} the directed trees obtained by taking G and considering only the up- or the down-arcs, respectively. For each task $\tau_i = (s_i, t_i)$ we define two tasks $\tau_{up,i} = (s_i, v_i)$ and $\tau_{down,i} = (v_i, t_i)$. We define $T_{up} = \{\tau_{up,i} | \tau_i \in T\}$ and $T_{down} = \{\tau_{down,i} | \tau_i \in T\}$. Let $I_{up} = (G_{up}, T_{up}, p)$ and $I_{down} = (G_{down}, T_{down}, p)$. W.l.o.g. we assume that I_{up} and I_{down} have the same congestion c . We compute the schedules *DTREE*(I_{up}) and *DTREE*(I_{down}) and the respective maps $task_{up}$ and $task_{down}$ (see Section 3.1).

We observe that for each k the map $task_{up}^{(k)}$ defined by $task_{up}^{(k)}(e, t) := task_{up}(e, t + k \bmod c)$ yields a valid task assignment. We define the maps $task_{down}^{(k)}$ (for the different values of k) similarly. Our randomized algorithm works as follows: we choose integers k and k' independently and uniformly at random from the set $\{0, \dots, c-1\}$. Then, we compute the map $task$ from the maps $task_{up}^{(k)}$ and $task_{down}^{(k')}$: for each up-arc e we define $task(e, t) = task_{up}^{(k)}(e, t)$ for all $t \in \{0, \dots, c-1\}$. For each down-arc e' we define $task(e', t) = task_{down}^{(k')}(e', t)$ for all $t \in \{0, \dots, c-1\}$. Using the map $task$ we define the template schedule *RTREE*(I).

Theorem 3.5. *Let I be an instance of the PPRP on a bidirected tree with $c \leq p$. The schedule *RTREE*(I) guarantees a limit of*

- $c + D_i - 1$ in expectation for each task τ_i if I is an instance of the strict PPRP.
- $1.5c + D_i - 1$ in expectation for each task τ_i if I is an instance of the sporadic PPRP.

Proof. In the strict periodic setting, once k and k' are fixed all packets created by a task τ_i have the same delays at s_i and v_i . The values k and k' were chosen uniformly at random from the set $\{0, \dots, c-1\}$. Thus, the packets of each task τ_i have an expected delay of $\frac{c-1}{2}$ in s_i and an expected delay of $\frac{c-1}{2}$ in v_i . This yields an expected limit of $c + D_i - 1$ in the strict periodic setting.

In the sporadic setting the packets created by τ_i have an expected delay of $\frac{c-1}{2}$ in v_i . However, they might be delayed up to $c-1$ times in s_i (depending in their release time). This yields an expected limit of $1.5c + D_i - 1$ in the sporadic setting. \square

The schedules for bidirected trees presented so far are not direct, i.e., packets might wait in vertices different from their start vertex. Now we present an algorithm for computing a direct schedule $BTREE_{dir}(I)$ for a packet routing instance on a bidirected tree. It requires that $c \leq p/2$. This might look restrictive at first glance. However, we will show afterwards that there are instances with $c = \frac{3}{4}p$ for which there does not exist a direct periodic schedule at all. Thus, we need a certain upper bound on the congestion c in order to guarantee that there actually is a direct periodic schedule which we can compute.

Algorithm $BTREE_{dir}(I)$

- (1) Sort task descendingly by the height of the peak vertex of their path.
- (2) Consider the tasks in this order. Assign each task the smallest initial waiting time such that its packets do not collide with packets of previously considered tasks.

Let $I = (G, T)$ be an instance of the sporadic PPRP on a bidirected tree G such that $c \leq p/2$. We present our algorithm which finds a direct template schedule $BTREE_{dir}(I)$ with limit $D_i + 2c - 1$ for each task τ_i . It is based on the concepts presented in [2, Theorem 3.4].

We sort the tasks descendingly by their height $h(\tau_i)$ (the distance between v_r and P_i). W.l.o.g. let $\tau_1, \tau_2, \dots, \tau_{|T|}$ be this order. Our schedule is a direct schedule, i.e., each packet is delayed for a certain number of steps and then moves to its destination without being delayed any further. We define the schedule via a template schedule with periodicity $\bar{p} := 2c$. We iterate over the tasks with $i = 1$ to $|T|$. Consider the i -th iteration. Let $P_i = \{v_0, v_1, \dots, v_{|P_i|-1}\}$ be the path of τ_i and let $e_j = (v_j, v_{j+1})$ for all $j \in \{0, \dots, |P_i| - 2\}$. Let w_i be the smallest positive integer such that $task(e_j, w_i + j \bmod \bar{p}) = \text{undefined}$ for all relevant values for j . We assign τ_i the initial waiting time w_i and define $task(e_j, w_i + j \bmod \bar{p}) = \tau_i$ for all respective values for j . We will show in Theorem 3.6 that there is always a value for w_i with $0 \leq w_i \leq 2c - 2 < \bar{p}$. We denote by $BTREE_{dir}(I)$ the resulting schedule.

Theorem 3.6. *Let I be an instance of the sporadic PPRP on a bidirected tree with $c \leq p/2$. The schedule $BTREE_{dir}(I)$ is a direct template schedule which guarantees a limit of $2c + D_i - 2$ for each task τ_i .*

Proof. Let τ_i be task. First of all we show that we can always find a value w_i with $w_i \leq \bar{p} - 1$ which does not interfere with any previous assignment for tasks $\tau_{i'}$ with $0 \leq i' < i$. Let e and e' be the two arcs in P_i which are incident to v_i . (The case that there is only one such arc can be proven with a similar reasoning as below.) Recall that we considered the tasks in an order given by the values $h(\tau_i)$. Thus, when assigning the initial delay w_i to τ_i only tasks which use e or e' could possibly interfere. We say a task $\tau_{i'}$ with $0 \leq i' < i$ blocks a timeslot m if there is an arc e_j such that $task(e_j, m + j \bmod \bar{p}) = \tau_{i'}$. Since G is a bidirected tree from the definition of *task* it follows that each task which uses e or e' can block at most one timeslot m with $0 \leq m < c$.

Excluding τ_i there are at most $2 \cdot (c - 1)$ tasks whose path uses e or e' . This implies that there is a value for w_i with $0 \leq w_i \leq 2c - 2$ such that w_i is not blocked and thus $w_i \leq 2c - 2$. By definition of *task* each packet $M_{i,j}$ waits in s_i for w_i timesteps and then moves to t_i without being delayed any further. Thus,

$BTREE_{dir}(I)$ is a direct periodic schedule and $2c + D_i - 2$ is a valid limit for each task τ_i . \square

Now we present an instance with $c = \frac{3}{4}p$ for which there can be no direct periodic schedule. This justifies our required bound on the congestion for the algorithm $BTREE_{dir}(I)$.

There exists an instance $I = (G, \mathcal{P})$ of the path coloring problem (i.e., the problem of coloring given paths such that two paths which share an arc are colored with different colors) with the following properties (see [6]):

- the underlying graph G is a bidirected tree
- for each arc e there are at most three paths in \mathcal{P} which use e
- any feasible path coloring for I needs at least five colors

Starting from I we construct an instance $\bar{I} = (\bar{G}, T, p)$ of the periodic packet routing problem. The graph \bar{G} is constructed as follows: starting with G , we replace each pair of anti-parallel arcs by a path with twelve anti-parallel arcs. We call all vertices which exist in both G and \bar{G} the *old vertices*. For each path $P_i \in \mathcal{P}$ starting in a vertex u and ending in a vertex v we add a task $\tau_i = (u, v)$. Finally, we define $p := 4$.

Proposition 3.7. *There is an instance \bar{I} of the strict PPRP with $c \leq \frac{3}{4}p$ for which there is no direct periodic template schedule.*

Since in I each arc is used by at most three paths and $p = 4$ we have that $c \leq \frac{3}{4}p$ for our instance \bar{I} . We show that there can be no direct template schedule for \bar{I} by using the fact that any path coloring for I needs at least five colors. Assume on the contrary that there is a direct template schedule for \bar{I} . We know that $3 \leq \bar{p} \leq 4$. Denote by $leave(\tau_i, u, j)$ the time when a packet $M_{i,j}$ leaves a vertex $u \in P_i$. Since for each arc in G we have a directed path of length twelve in \bar{G} we have that there is a value $w_i \in \{0, 1, 2, 3\}$ for each task τ_i with $leave(\tau_i, u, j) \equiv w_i \pmod{\bar{p}}$ for each old vertex $u \in P_i \setminus \{t_i\}$ and each packet $M_{i,j}$. Thus, for two tasks τ_i and τ_j whose paths share an arc it holds that $w_i \neq w_j$. This implies that the map $c(P_i) := w_i$ defines a path coloring for I with four colors. But this is a contradiction since each path coloring for I needs at least five colors.

3.3. Undirected Trees. Now we adjust the techniques introduced above for bidirected trees to undirected trees. Here, we need to take care that two packets which move towards each other do not interfere. Let I be an instance of the sporadic PPRP on an undirected tree G such that $c \leq p/2$. We present an algorithm which finds a (not necessarily direct) schedule $UTREE(I)$ with limit $4c - 2c \left(\frac{1}{2}\right)^{\lceil diam(G)/2 \rceil - 1} + D_i - 1$ for each task τ_i . We will prove below that the bound for the congestion is really necessary. In particular, we will show that for any $\alpha, \beta > 0$ there is an instances I of the strict PRP with $p/2 < c \leq (1 + \epsilon)p/2$ for which no template schedule can guarantee a limit of $\alpha p + D_i + \beta$ for each task τ_i .

Then we study the problem of finding directed schedules for the sporadic PPRP on undirected trees. We will show an algorithm $UTREE_{dir}(I)$ which finds a direct template schedule with limit $4c + D_i - 3$ for each task τ_i if $c \leq \frac{p}{4}$. We will justify the bound on the congestion by giving an instance on an undirected path with $p/2 < c \leq (1 + \epsilon)p/2$ for which there exists no direct template schedule at all.

Algorithm $UTREE(I)$

- (1) Adapt algorithm $BTREE(I)$ to undirected trees.
- (2) Always ensure that moving packets are either all on vertices with even level or all on vertices with odd level (parity property).

Now we present our algorithm for computing the schedule $UTREE(I)$. It adapts the concepts of $BTREE(I)$ to undirected trees. Our strategy to avoid collisions is that for the packets which have left their start vertex located on a vertex v at a time t we have that $h(v) + t$ is always even (implying that these packets are either all located on a vertex with even height or all located on a vertex with odd height). We call this the *parity property*. The parity property can be understood to halve the bandwidth. However, since $c \leq p/2$ there is still enough bandwidth available.

We define $\bar{p} := 2c$. Like in $BTREE(I)$ we start with the root v_r (chosen such that $h(G) \leq \lceil \text{diam}(G)/2 \rceil$). We define the map $task$ such that no packet needs to wait in v_r and such that for an edge $e = \{v, v_r\}$ we have that $task(e, k) = \tau$ only if the packets of τ move from v to v_r and k is odd or the packets of τ move from v_r to v and k is even. Such values for $task$ can be found since $c \leq p/2$. We continue with the same iterative procedure as in $BTREE(I)$. However, in order to ensure the parity property now if a packet is delayed it needs to wait twice as many times as in $BTREE(I)$. Again, the fact that $c \leq p/2$ ensures that there is enough available bandwidth.

Since $\bar{p} = 2c$ each packet is delayed at most $2c - 1$ times in its start vertex. After that, each packet is delayed at most $2c \left(1 - \left(\frac{1}{2}\right)^{\lceil \text{diam}(G)/2 \rceil - 1}\right)$ times. This gives a limit of $4c - 2c \left(\frac{1}{2}\right)^{\lceil \text{diam}(G)/2 \rceil - 1} + D_i - 1$ for each task τ_i . Denote by $UTREE(I)$ the schedule resulting.

Theorem 3.8. *Let I be an instance of the sporadic PPRP on an undirected tree with $c \leq p/2$. The schedule $UTREE(I)$ guarantees a limit of $4c - 2c \left(\frac{1}{2}\right)^{\lceil \text{diam}(G)/2 \rceil - 1} + D_i - 1$ for each task τ_i .*

Proof. Similar proof as for Theorem 3.3. □

For $UTREE(I)$ we required that $c \leq p/2$. This might look restrictive. However, we can show with following proposition that this bound is really necessary.

Proposition 3.9. *For any $\alpha, \beta, \epsilon > 0$ there is an instances I of the strict PPRP with $p/2 < c \leq (1 + \epsilon)p/2$ on an undirected path for which there is no template schedule which guarantees a limit of $\alpha p + D_i + \beta$ for every task τ_i .*

Note that this implies that also no template schedule can guarantee a limit of $\alpha c + D_i + \beta$ for each task τ_i for the respective instances. Also, the same statement is implied for instances of the sporadic PPRP (since the strict periodic setting is a special case of the sporadic setting). We will proof the proposition in the sequel.

Now, we study the problem of computing a direct schedule for an undirected tree. We show how to adapt the algorithm $BTREE_{dir}(I)$ to undirected trees. Given an instance I of the sporadic PPRP on an undirected tree with $c \leq \frac{p}{4}$. We show how to compute the schedule $UTREE_{dir}(I)$ which guarantees a limit of $4c + D_i - 3$ for each task τ_i . Afterwards, we will justify the required bound of $c \leq \frac{p}{4}$: we give an

example of an instance with $\frac{p}{2} < c \leq (1 + \epsilon)\frac{p}{2}$ on an undirected path for which there can be no direct schedule at all.

Algorithm $UTREE_{dir}(I)$

- (1) Sort task descendingly by the height of the peak vertex of their path.
- (2) Consider the tasks in this order. Assign each task the smallest initial waiting time such that its packets do not collide with packets of previously considered tasks.

Now we describe the algorithm for computing $UTREE_{dir}(I)$. We sort the tasks descendingly according to $h(\tau_i)$. W.l.o.g. let $\tau_1, \dots, \tau_{|T|}$ be this order. Our schedule is a direct schedule, i.e., each packet is delayed for a certain number of steps and then moves to its destination without being delayed any further. We define the template schedule with periodicity $\bar{p} := 4c$. We iterate over the tasks from $i = 1$ to $|T|$. Let τ_i be the task considered in the i -th iteration. Like in $BTREE_{dir}(I)$ we call a value $w_i < \bar{c}$ valid if $task(e_j, w_i + j \bmod 4c) = \text{undefined}$ for all edges e_j on P_i . We assign τ_i the smallest valid value w_i such that $h(s_i) + w_i$ is even. We set $task(e_j, w_i + j \bmod 4c) = \tau_i$ for the respective edges e_j . We will show in Theorem 3.10 that there is always a valid value for w_i such that $h(s_i) + w_i$ is even with $w_i \leq 4c - 3$. Denote by $UTREE_{dir}(I)$ the resulting schedule.

Theorem 3.10. *Let I be an instance of the sporadic PPRP on an undirected tree G with $c \leq p/4$. The schedule $UTREE_{dir}(I)$ is a direct template schedule which guarantees a limit of $4c + D_i - 3$ for each task τ_i .*

Proof. Let τ_i be task. First we show that there is always a valid value w_i with $w_i \leq 4c - 3$. Let e and e' be the two edges in P_i which are incident to v_i . (The case that there is only one such edge can be proven with a similar reasoning as below.) When assigning the initial delay w_i to τ_i only tasks which use e or e' could possibly interfere with τ_i . We say a task $\tau_{i'}$ with $0 \leq i' < i$ blocks a timeslot m if there is an edge e_j such that $task(e_j, m + j \bmod 4c) = \tau_{i'}$.

Since G is a tree it follows that if two paths P_i and $P_{i'}$ share an edge then either on all edges which they have in common they point in the same direction or on all these edges they point in opposite directions. For ease of notation we say that τ_i and $\tau_{i'}$ move in the same direction or they move in opposite direction.

First we discuss tasks $\tau_{i'}$ which move in the opposite direction of τ_i . For the analysis we define a time-dependent edge-coloring $c : E \times \mathbb{N} \rightarrow \{\text{red}, \text{green}\}$. Denote by $d(u, v)$ the length of a shortest path between two vertices u and v . Let $e = \{u, v\}$ be an edge such that $d(u, v_r) = i$ and $d(v, v_r) = i + 1$. For t such that $t + i$ is even we define $c(e, t) = \text{green}$, for t such that $t + i$ is odd we define $c(e, t) = \text{red}$. From the algorithm it follows that $h(s_{i'}) + w_{i'}$ is even. Since the periodicity \bar{p} is even this implies that on their way up (in direction of v_r) packets created by $\tau_{i'}$ use only reds edges, and on their way down they use only green edges. If $h(s_i) + w_i$ is even the same holds for packets created by τ_i . Since τ_i and $\tau_{i'}$ move in opposite directions, if two packets created by these tasks meet then one of them moves up and the other one moves down. Thus, they do not interfere since then they use differently colored edges. This implies that tasks which move in opposite direction of τ_i do not block any timeslots m such that $h(s_i) + m$ is even.

Now let $\tau_{i'}$ be a task which moves in the same direction as τ_i . Since G is a tree $\tau_{i'}$ can block at most one timeslot m with $0 \leq m < \bar{c}$. Excluding τ_i there are at

most $2 \cdot (c - 1)$ tasks which use e or e' in the same direction as τ_i . Since $c \leq p/4$ this implies that there are at most $2 \cdot (c - 1)$ blocked timeslots m such that $m + h(s_i)$ is even. This implies that $w_i \leq 4c - 3 < \bar{p}$ and thus $UTREE_{dir}(I)$ guarantees a limit of $4c + D_i - 3$ for each task τ_i . From its definition it follows that $UTREE_{dir}(I)$ is direct. \square

Now we show that for any $\epsilon > 0$ there are instances on an undirected path with $c \leq (1 + \epsilon)p/2$ for which no direct schedule exists. Thus, we have to give some upper bound on the congestion of an instance in order to guarantee that an algorithm can compute a direct schedule for it (as we did for $UTREE_{dir}(I)$).

Proposition 3.11. *For any $\epsilon > 0$ there are instances of the strict PPRP on an undirected path with $p/2 < c \leq (1 + \epsilon)p/2$ for which there is no ‘direct template schedule.’*

Proof. Let $\epsilon > 0$. We describe how to construct an instance with the described property. Let k be an integer such that $\frac{2}{k} \leq \epsilon$. The graph G is an undirected path with $2k + 1$ vertices. Denote by v_A and v_E the vertices at the ends of G . We define $p := 2k$. We introduce $k + 1$ tasks $\tau_1, \dots, \tau_{k+1}$ with $\tau_i = (v_A, v_E)$ for all $i \in \{1, 2, \dots, k + 1\}$ and one task $\tau_{k+2} = (v_E, v_A)$. We call the former tasks the *good tasks*. Since each edge is used by all tasks we have that $c = k + 2 = \frac{p}{2} + \frac{p}{2} \cdot \frac{2}{k} \leq (1 + \epsilon) \cdot \frac{p}{2}$.

Assume on the contrary that there is a direct template schedule for I . Denote by w_i the initial delay for the first packet of each task. We assume that w_{k+1} is even (the case that w_{k+1} is odd can be proven similarly). Then each good task τ_i must have an odd initial delay w_i with $0 \leq w_i < 2k$. Since there are $k + 1$ good tasks but only k odd integers in the interval $[0, \dots, 2k - 1]$ this yields a contradiction. \square

Note that Proposition 3.11 implies the same statement for instances of the sporadic PPRP. Now we can prove Proposition 3.9.

Proof of Proposition 3.9: Let $\alpha, \beta, \epsilon > 0$. We construct an instance $I = (G, T, p)$ of the periodic packet routing problem on an undirected path such that $\frac{p}{2} < c \leq (1 + \epsilon) \cdot \frac{p}{2}$ but there can be no periodic schedule which guarantees a limit of $D_i + \alpha \cdot p + \beta$ for each task τ_i .

Let k be an integer such that $\frac{2}{k} \leq \epsilon$. The graph G is an undirected path with $(2k - 1)(k + 2)(\alpha \cdot p + \beta) + 2$ vertices. Denote by v_A and v_E the vertices at the two ends of G . We introduce $k + 1$ tasks $\tau_1, \dots, \tau_{k+1}$ with $\tau_i = (v_A, v_E)$ for each $i \in \{1, 2, \dots, k + 1\}$ and one task $\tau_{k+2} = (v_E, v_A)$. We set $p := 2k$. Since each edge is used by all tasks we have $c = k + 2 = \frac{p}{2} + \frac{p}{2} \cdot \frac{2}{k} \leq (1 + \epsilon) \frac{p}{2}$.

Assume on the contrary that there is a template schedule S for I in which each task is delayed at most $\alpha \cdot p + \beta$ times. Thus, there can be at most $(k + 2)(\alpha \cdot p + \beta)$ edges on which packets are delayed. Since G has $(2k - 1)(k + 2)(\alpha \cdot p + \beta) + 1$ edges by the pigeon hole principle there must be $2k$ consecutive edges on which no packet is delayed. Thus, the schedule S is a direct schedule on this subpath. However, with a similar reasoning as in the proof of Proposition 3.11 we conclude that this is impossible. \square

4. GLOBAL- AND EDGE-PRIORITY SCHEDULES, SPORADIC SETTING

In this section we study global-priority and edge-priority schedules on directed and bidirected trees in the sporadic setting. We present schedules and prove limits

for them. We also give lower bounds which show the limitations of global- and edge-priority schedules.

First we present a schedule $GPRIO(I)$ for bidirected trees. The schedule assigns the priorities to the tasks according to their height. Then we study an edge-priority schedule $EPRIO(I)$ for directed trees. It is based on $GPRIO(I)$ but uses a more sophisticated tie-breaking procedure for tasks with the same height. For both schedules we prove bounds for their limits and show that they are best possible within their respective schedule classes. Table 2 summarizes the results presented in this section.

Algorithm $GPRIO(I)$

- (1) Sort tasks by height $h(\tau_i)$ of peak vertex.
- (2) Define $\tau_i \prec \tau_j$ if and only if $h(\tau_i) < h(\tau_j)$.
- (3) Ties in prioritization are broken arbitrarily.

Now we present the algorithm for computing the schedule $GPRIO(I)$. Let I be an instance of the strict or sporadic packet routing problem on a bidirected tree. Note that here we do not impose any conditions on the congestion or whether I is a strict or sporadic instance. However, when we prove our limits for $GPRIO(I)$ in Theorem 4.1 we will require a bound on c . We first order the tasks by the height $h(\tau_i)$ of their peak vertex. A task τ_i has a higher priority than a task τ_j if its peak vertex is closer to the root than the peak vertex of τ_j . Formally, $\tau_i \prec \tau_j$ if and only if $h(\tau_i) < h(\tau_j)$. This yields a partial order \prec . We complete \prec to a total order arbitrarily. In particular, this implies that the prioritization of tasks with the same peak vertex are chosen arbitrarily.

Theorem 4.1. *Let I be a instance of the sporadic PPRP on a bidirected tree with $c \leq p/3$. The schedule $GPRIO(I)$ guarantees a limit of $2c + D_i - 2$ for each task τ_i .*

We will prove the theorem in the sequel. It is easy to see that there are instances – even in the strict periodic case – where $GPRIO(I)$ delays a packet $2c - 2$ times. We will show later that $GPRIO(I)$ is best possible in the sense that no global-priority schedule can guarantee a better limit for every task in every instance.

Algorithm $EPRIO(I)$

- (1) Sort tasks by height $h(\tau_i)$ of peak vertex.
- (2) Define $\tau_i \prec \tau_j$ if and only if $h(\tau_i) < h(\tau_j)$.
- (3) Ties in prioritization are broken such that each task is delayed at most $1.5c$ times in total.

Now we study our edge-priority schedule $EPRIO(I)$ for directed trees. Let I be an instance of the strict or sporadic packet routing problem on a directed tree. Again, we do not require a bound on c here in the definition of the schedule but later in the theorem. We need to define a prioritization for each arc separately. First, like in $GPRIO(I)$, we define that $\tau \prec \tau'$ if $h(\tau) < h(\tau')$ for each arc prioritization. So now we focus on the tie-breaking for tasks τ, τ' with $h(\tau) = h(\tau')$. Consider a vertex v . Let e_1, \dots, e_r be the ingoing arcs of v which are up-arcs and let e'_1, \dots, e'_s be the outgoing arcs of v which are down-arcs. Denote by E_v the set of all these arcs. In case that v_r is not the root vertex let \bar{e} be the remaining arc which is adjacent to

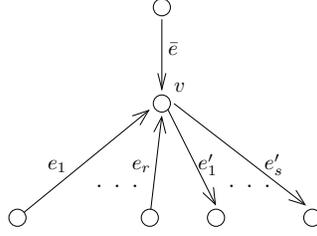


FIGURE 4.1. Sketch for the definition of $EPRIO(I)$.

v . See Figure 4.1 for a sketch. We define the prioritization for all tasks with peak vertex v .

We define T_v to be the set of tasks which use v . Let T'_v be the tasks in T_v which do not use \bar{e} . We compute a minimum path coloring for the paths of the tasks T_v . This can be done by a reduction to edge-coloring of a bipartite multigraph. Exactly c colors are needed, for details see [12]. Assume that the paths are colored with colors $\{1, 2, \dots, c\}$. If there is an arc $\bar{e} \in E_v$ such that more than $c/2$ paths use both \bar{e} and \bar{e} then we assume w.l.o.g. that the paths which use \bar{e} and \bar{e} use the colors $1, \dots, m$ if \bar{e} is an up-arc and the colors $c - m + 1, \dots, c$ if \bar{e} is a down-arc (assuming that there are m such paths). Note that there can be at most one arc \bar{e} with this property. Denote by $f(\tau)$ the color of the path of a task τ . Let $\tau, \tau' \in T'_v$ be a pair of tasks. Then, on each up-arc in E_v we define $\tau \prec \tau'$ if $f(\tau) < f(\tau')$. For each down-arc in E_v we define $\tau \prec \tau'$ if $f(\tau) > f(\tau')$. Denote by $EPRIO(I)$ the resulting schedule.

Theorem 4.2. *Let I be an instance of the sporadic PPRP on a directed tree with $c \leq 2p/5$. The schedule $EPRIO(I)$ guarantees a limit of $\frac{3}{2}c + D_i - 1$.*

We will prove the theorem later. We will also show that $EPRIO(I)$ is best possible.

Now we prove properties of $GPRIO(I)$ and $EPRIO(I)$ which we need in order to prove Theorems 4.1 and 4.2. First, we analyze the maximum number of delays which a packet can encounter on its way up (in the direction of the root).

Lemma 4.3. *Let I and \bar{I} be instances of the sporadic PPRP on a bidirected and directed tree, respectively. In the schedules $GPRIO(I)$ and $EPRIO(\bar{I})$ a packet created by a task τ_i can be delayed at most $c - 1$ times between s_i and v_i .*

Proof. We prove the claim only for $GPRIO(I)$ since for $EPRIO(\bar{I})$ we can follow exactly the same argumentation. Consider a packet $M_{i,j}$ and let $e \in P_i$ be the arc on P_i between s_i and v_i which is closest to v_r . Since we are only interested in the behaviour of $M_{i,j}$ between s_i and v_i it is sufficient to consider only the tasks \tilde{T} which use e and whose priority on e is at least as high as the priority of τ_i . W.l.o.g. let $\tau_1 \prec \tau_2 \prec \dots \prec \tau_i$ be these tasks.

We consider the following instance I' instead of I . (While doing this we assume that the timesteps when the respective tasks create new packets remain the same.) For each task τ_k let L_k denote the number of arcs on P_k below e . We replace each path P_k by a path P'_k which consists (in this order) of

- a new path of length L_k which is used only by τ_k ,
- the arc e , and

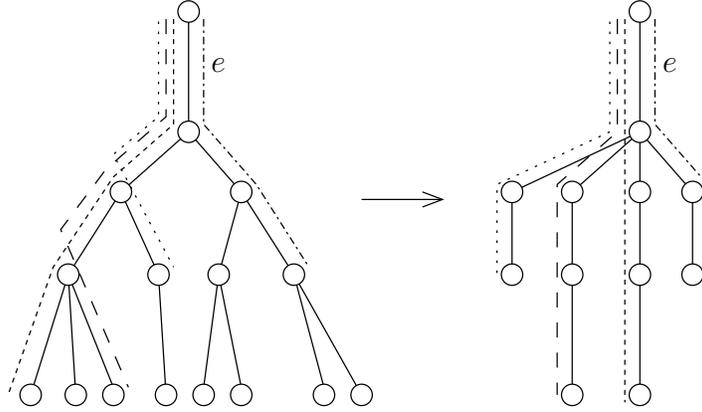


FIGURE 4.2. The transformation from I to I' in the proof of Theorem 4.3.

- all arcs of P_k above e .

See Figure 4.2 for a sketch. In $GPRIO(I)$ and $GPRIO(I')$ we use a global prioritization and all tasks in \tilde{T} use e . Thus, all packets created by tasks in \tilde{T} traverse e in $GPRIO(I)$ at the same timesteps as in $GPRIO(I')$. (This can be shown by an induction which transforms I to I' stepwise.)

So now consider e in I' . All tasks have a period length of p . Before reaching e no packet is delayed. This implies that the packet $M_{i,j}$ is delayed at most $i - 1$ times. Since $i \leq c$ this proves the claim. \square

Now we focus on the delay of the packets on the way away from the root.

Lemma 4.4. *Let I be an instance of the sporadic PPRP on a bidirected tree with $c \leq p/3$. In the schedule $GPRIO(I)$ on the way away from v_r each packet is delayed at most $c - 1$ times.*

Proof. First observe that on the way down a packet is delayed only on the first down-arc. We prove the claim by induction over the height of the respective tasks. First, consider a task τ_i with $h(\tau_i) = 0$. Let $e = (u, v)$ be the first down-arc on P_i . On the way up each packet is delayed at most $c - 1$ times. Thus, two packets created by τ_i reach u with a time difference of at least $\lceil 2p/3 \rceil$. Since $c \leq \lfloor p/3 \rfloor$ in each time interval of length $\lceil 2p/3 \rceil$ there are less than c packets which use e and which have a higher priority than τ_i . Thus, each packet created by τ_i is delayed at most $c - 1$ times on its way down.

Now assume the claim is true for all tasks τ_j such $h(\tau_j) \leq m$. Consider a task τ_i with $h(\tau_i) = m + 1$. Let again $e = (u, v)$ be the first down-arc on P_i . From the induction hypothesis we conclude that for each task τ_j which has a higher priority than τ_i on e the following holds: Two packets created by τ_j reach u with a time difference of at least $\lceil p/3 \rceil$. Thus, in each interval of length $\lceil p/3 \rceil$ there are less than c packets which use e and have a higher priority than τ_i . Like above, two packets created by τ_i reach u with a time difference of at least $\lceil 2p/3 \rceil$. Thus, each packet created by τ_i is delayed at most $c - 1$ times on its way down. \square

Now we can proof the limits guaranteed by the schedule $BTREE(I)$.

Proof of Theorem 4.1: From Lemma 4.3 it follows that packets are delayed at most $c - 1$ times on their way up. Lemma 4.4 shows that on their way down they are delayed at most $c - 1$ times as well. Hence, the schedule $BTREE(I)$ guarantees a limit of $2c + D_i - 2$ for each task τ_i . \square

Combining Lemma 4.3 with a similar reasoning as in Lemma 4.4 we can now prove that $EPRIO(I)$ guarantees a limit of $\frac{3}{2}c + D_i - 1$ for each task.

Proof of Theorem 4.2: Let τ_i be a task with peak vertex v . Assume that the path of τ_i was colored with color $f(\tau_i)$ when v was considered by the algorithm. First, we discuss the case that for no arc $\tilde{e} \in E_v$ there are more than $c/2$ paths which use both \tilde{e} and \bar{e} . Then, there are at most $c/2 + (f(\tau_i) - 1) + (c - f(\tau_i)) = 3c/2 - 1$ tasks which share an arc with P_i and which have a higher priority than τ_i on these arcs. We say that such tasks *interfere* with τ_i .

Now we discuss the case that there is an arc $\tilde{e} \in E_v$ as described above. If P_i does not use \tilde{e} we can show as above that at most $3c/2 - 1$ tasks interfere with τ_i . Now consider the case that P_i uses \tilde{e} . Due to our assumption on the coloring, for such tasks there are at most $(f(\tau_i) - 1) + (c - f(\tau_i)) = c - 1$ tasks which interfere with τ_i .

Since also $c \leq 2p/5$ one can show like in Lemma 4.3 that on its way up the packets of τ_i are delayed at most $c/2 + (f(\tau_i) - 1)$ times. Like in Lemma 4.4 one can show inductively that on the way down each packet created by τ_i is delayed at most $c - f(\tau_i)$ times. In the inductive step one uses that two packets created by the same task have a minimum time difference of $2p/5$ and $c \leq 2p/5$. We conclude that each packet is delayed at most $\frac{3}{2}c - 1$ times in total. \square

Here we would like to comment that without a bound on the congestion c in the schedules $GPRIO(I)$ and $EPRIO(I)$ it might happen that a packet is delayed more than $c - 1$ times on its way down. Consider the example instance depicted in Figure 4.3 with $c = p$. Each depicted path is used by $p/2$ tasks.

The task in the set T_1 have highest priority. Thus, depending on the creation times of the packets it might happen that there are p packets \mathcal{M}_2 created by tasks in T_2 which leave the vertex v in a row. Similarly, there could be a set of p packets \mathcal{M}_4 in a row created by tasks in T_4 which arrive at v' and each of the next sets of $p/2$ packets created by T_2 arrives with a time distance of $p/2$ to the previous set. If the packets \mathcal{M}_2 and \mathcal{M}_4 collide directly, then the packets \mathcal{M}_4 are delayed p times. This results in $2p$ packets created by T_2 moving down in a row. These $2p$ packets will have priority over any packet further down in the tree which shares a path with them. In particular, a packet further down in the tree can be delayed $2p + p/2 - 1$ times ($2p$ times by the $2p$ packets mentioned before and $p/2 - 1$ times by other packets using the respective arc). One can continue this construction and thus obtain instances in which there is a packet which is delayed arbitrarily often on its way down. This justifies that we require a bound on the congestion in order to guarantee good limits for $GPRIO(I)$ and $EPRIO(I)$.

4.1. Lower Bounds. Now we prove that the schedules $GPRIO(I)$ and $EPRIO(I)$ are best possible. We show that there are instances on directed trees for which no global-priority schedule can guarantee a better limit than $GPRIO(I)$. Also, we present instances on directed trees for which no edge-priority schedule can guarantee a better limit than $EPRIO(I)$.

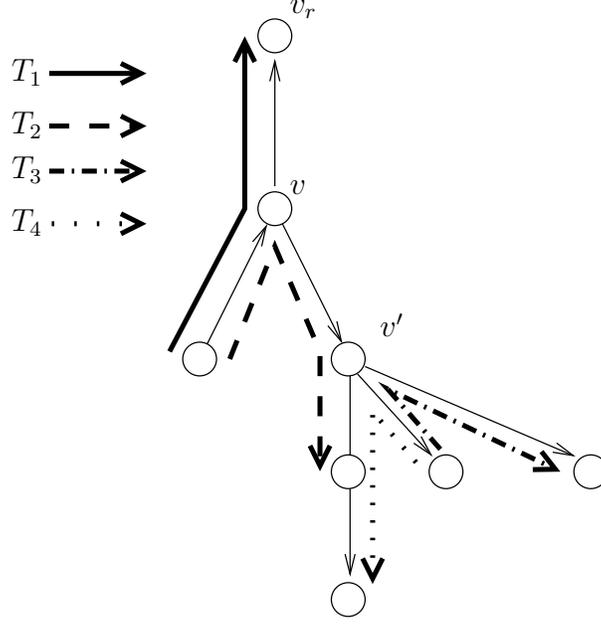


FIGURE 4.3. Example on a rooted directed tree where in $GPRIO(I)$ and $EPRIO(I)$ a packet can be delayed more than $c - 1$ times on its way down.

Theorem 4.5. *For each period length p and any value for c there exist instances of the sporadic PPRP on directed trees for which no global-priority schedule can guarantee a better limit than $2c + D_i - 2$.*

Proof. Consider a star graph with one vertex v_r in the center with c ingoing arcs $E = \{e_1, \dots, e_c\}$ and c outgoing arcs $E' = \{e'_1, \dots, e'_c\}$. We introduce c^2 tasks. The path of each task uses one of the arcs in E and one of the arcs in E' such that no two tasks use the same two arcs. Denote by I the resulting instance. We claim that for I no global-priority schedule can guarantee a better limit than $2c = 2c + D_i - 2$.

Assume we have a global-priority schedule for I . Then, there must be a task τ which has the lowest priority. Assume w.l.o.g. that τ uses the arcs e_1 and e'_1 . Denote by E_1 and E'_1 the other tasks which use e_1 and e'_1 . We consider the (sporadic) case that only τ and the tasks in $E_1 \cup E'_1$ create packets. Let M_τ be a packet created by τ . Since we are in the sporadic setting it can happen that M_τ is delayed once by a packet of every task in E_1 and once by a packet of every task in E'_1 . Thus, in this case M_τ needs $2c$ steps. Hence, the schedule cannot guarantee a better limit than $2c + D_i - 2$. \square

Note that Theorem 4.5 implies the stated lower bound also for global-priority schedules on bidirected trees (since directed trees are special cases of bidirected trees). Also note that in the instance described in Theorem 4.5 there can be a task which is delayed $2c - 2$ times by the schedule $GPRIO(I)$, even in the strict periodic setting (due to bad tie-breaking decisions).

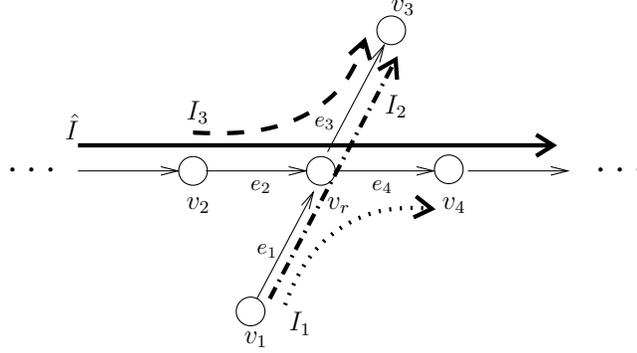


FIGURE 4.4. The gadget used in the proof of Theorem 4.6.

Theorem 4.6. *For each period length p and any even value for c there exist instances of the sporadic PPRP on directed trees for which no edge-priority schedule can guarantee a better limit than $\frac{3}{2}c + D_i - 1$.*

Proof. Consider a very long path along which $c/2$ tasks send their packets. Denote by \hat{I} these tasks. Assume on the contrary that we have a schedule S which guarantees a better limit than $\frac{3}{2}c + D_i - 1$ for each task τ_i .

At every other vertex with degree 2 we introduce a gadget. Figure 4.4 shows the gadget: We have a star with the vertices v_1, v_2, v_3, v_4, v_r and the arcs $(v_1, v_r), (v_2, v_r), (v_r, v_3), (v_r, v_4)$. We introduce $c/2$ tasks with the path (v_2, v_r, v_3) , $c/2$ tasks with the path (v_1, v_r, v_3) , and $c/2$ tasks with the path (v_1, v_r, v_4) . We denote these tasks by I_1, I_2 , and I_3 , respectively. We introduce $\frac{3}{2}c^2$ of these gadgets. For every task $\hat{\tau} \in \hat{I}$ there can be at most $3c$ gadgets in which a task $\tau \notin \hat{I}$ has a higher priority than $\hat{\tau}$. Since $|\hat{I}| = c/2$ by the pigeon hole principle there must be a gadget in which every task in \hat{I} has a higher priority than the task not in \hat{I} which use this gadget. We prove in the sequel that depending on the release time of the packets in this gadget a packet created by one of the tasks in $I_1 \cup I_2 \cup I_3$ needs at least $\frac{3}{2}c + 1$ steps.

First, consider the arc $e_1 = (v_1, v_r)$. Denote by the $\tau^{(1)} \in I_1$ the task in I_1 which has the lowest priority among all tasks in I_1 . Denote by x the number of tasks in I_2 which have a higher priority than $\tau^{(1)}$ and by y the number of tasks in I_2 which have a lower priority than $\tau^{(1)}$. Now consider the arc $e_3 = (v_r, v_3)$. Let $\tau^{(3)} \in I_3$ the task in I_3 with lowest priority. We define x' to be the number of tasks in I_2 with higher priority than $\tau^{(3)}$ and by y' the number of tasks in I_2 with lower priority than $\tau^{(3)}$ on e_3 . Observe that $x + y = c/2 = |I_2| = x' + y'$.

Now we distinguish several cases: First assume that $x \geq y$. In particular, this implies that $x \geq c/2$. Then there is a realisation in which a packet $M^{(1)}$ of $\tau^{(1)}$ is delayed once by a packet from each task in $\hat{I} \cup I_2 \cup I_1 \setminus \{\tau^{(1)}\}$. Thus, $M^{(1)}$ is delayed $\frac{3}{2}c - 1$ times. With a similar reasoning we can handle the case that $x' \geq y'$. So now assume that $x < y$ and $x' < y'$.

If $x' < y$ then there are tasks $I'_2 \subseteq I_2$ which have a lower priority than $\tau^{(1)}$ on e_1 and a lower priority than $\tau^{(3)}$ on e_3 . Note that $|I'_2| \geq y - x'$. Let task $\tau^{(2)} \in I'_2$ be the task which has the lowest priority on e_1 among all tasks in I'_2 . There is a realisation in which a packet of $\tau^{(2)}$ is delayed $x + c/2 + x' + c/2 + y - x' - 1 = \frac{3}{2}c - 1$

times. We can handle the case that $x < y'$ similarly. So now assume that $x \geq y'$ and $x' \geq y$. We observe that there are no cases left since $x' \geq y > x \geq y' > x'$ yields a contradiction. \square

For our schedules $GPRIO(I)$ and $EPRIO(I)$ we required a bound on the congestion of $c \leq p/3$ and $c \leq 2p/5$, respectively, which is stricter than the general condition $c \leq p$. Now we show that *any* global-priority or edge-priority schedule on a directed tree needs to require such a bound since otherwise it cannot guarantee any finite limit for each task.

Proposition 4.7. *Let $\alpha > 0$. There is an instance $I_\alpha = (G, T, p)$ of the sporadic PPRP on a directed tree G with $c = p$ such that for any edge-priority schedule ES there is a task $\tau \in T$ for which ES cannot guarantee a limit k with $k \leq \alpha \cdot c + D_i$.*

Proof. We describe how to construct the instance I_α . We introduce sets of tasks T_i . Each set T_i contains $p/2$ tasks which all have the same path. Let T_1 be the first of these sets. All tasks in T_1 are scheduled on a very long path P_1 . For each arc e on P_1 , there is a set of $p/2$ tasks T_e which use the arc e but no other arc on P_1 . We assume that P_1 is so long that there must be an arc e such that all tasks in T_1 have higher priority than the tasks in T_e (otherwise we could define creation times for the packets such that a packet from a task in T_1 is delayed αc times). We define $T_2 := T_e$. We prolong the path P_2 of the tasks in T_2 (actually, we do the following procedure for each arc $\bar{e} \in P_1$ but for our analysis later only the arc e will be important). For each arc e' on P_2 we introduce $c/2$ tasks $T_{e'}$ which use e' but no other arc on P_2 . Like above, we define P_2 to be so long that there must be an arc $e' \in P_2$ such that the task T_2 all have a higher priority than the tasks $T_{e'}$. By continuing inductively, for each edge-priority schedule ES for I_α we obtain sets T_1, \dots, T_ℓ with $c/2$ tasks each such that for each i we have that the tasks $T_i \cup T_{i+1}$ share an arc on which each task in T_i has a higher priority than any task in T_{i+1} . In the sequel, we show that there are creation times for the packets such in a set T_j there is a task whose packets are delayed at least αc times.

We denote by a *pack* of packets a set of $p/2$ packets which were all created at the same time by the tasks in a set T_i . We let T_1 create a pack $\mathcal{M}_1^{(1)}$ and we let T_2 create two packs $\mathcal{M}_1^{(2)}$ and $\mathcal{M}_2^{(2)}$. We define their creation times such that the pack $\mathcal{M}_1^{(1)}$ delays the pack $\mathcal{M}_1^{(2)}$ and thus the packets of packs $\mathcal{M}_1^{(1)}$ and $\mathcal{M}_2^{(2)}$ move in a row (after the delay). For induction, assume that T_i has created i packs $\mathcal{M}_1^{(i)}, \dots, \mathcal{M}_i^{(i)}$ whose packets move in a row. We let T_{i+1} create $i+1$ packs $\mathcal{M}_1^{(i+1)}, \dots, \mathcal{M}_{i+1}^{(i+1)}$ which collide with the packs $\mathcal{M}_1^{(i)}, \dots, \mathcal{M}_i^{(i)}$. Since the tasks in T_i all have a higher priority than the tasks in T_{i+1} this results in $\mathcal{M}_1^{(i+1)}, \dots, \mathcal{M}_{i+1}^{(i+1)}$ all moving in a row after the delay. By continuing inductively, we obtain that the tasks $T_{2\alpha+1}$ create $2\alpha+1$ packs (all moving in a row) with $(2\alpha+1) \cdot \frac{c}{2}$ packets in total. Thus, by choosing the creation times appropriately, we can enforce that a packet M created by a task in $T_{2\alpha+2}$ is delayed by all these packets. Hence, M is delayed $(2\alpha+1) \cdot \frac{c}{2} > \alpha c$ times. This implies that ES cannot guarantee a limit of $\alpha \cdot c + D_{2\alpha+2}$ for $T_{2\alpha+2}$. \square

Note that Proposition 4.7 implies the same statement for global-priority schedules (since they are special cases of edge-priority schedules).

Theorem 4.8. *For any $D > 0$ and any $c, p \in \mathbb{N}$ there is an instance $I = (G, T, p)$ of the sporadic PPRP with given paths on a chain graph G such that $D_i = D$ for each task $\tau_i \in T$, all tasks have the same start and the same destination vertex, and for every edge-priority schedule there is at least one task $\tau \in T$ which has a limit in $\Omega(c \cdot D)$.*

Proof. The instance I is constructed as follows. We introduce c^D tasks. Intuitively, they are arranged in a D -dimensional hypercube with edge-length c . We identify each task $\tau(k_0, k_1, \dots, k_{D-1})$ by integers $k_i \in \{0, 1, \dots, c-1\}$ for $i \in \{0, \dots, D-1\}$. Our graph is a chain graph with vertices v_0, \dots, v_D and c^{D-1} parallel edges connected each pair of adjacent vertices v_j, v_{j+1} . The paths are chosen such that on the connection between the vertices v_j, v_{j+1} two tasks $\tau(k_0, k_1, \dots, k_{D-1}), \tau(k'_0, k'_1, \dots, k'_{D-1})$ share an edge if and only if $k_i = k'_i$ for all $i \in \{0, 1, \dots, j-1, j+1, \dots, D-1\}$. We note that each edge is used by exactly c tasks.

Assume that there is an edge-priority schedule for I . For each task $\tau \in T$ let $d_{\tau, \ell}$ denote the number of tasks which have a higher priority than τ on the edge used by τ connecting v_ℓ and $v_{\ell+1}$. We calculate that $\sum_{\ell=0}^{D-1} \sum_{\tau \in T} d_{\tau, \ell} = Dc^{D-1} \sum_{k=0}^{c-1} k = Dc^{D-1} \frac{c(c-1)}{2} = Dc^D \frac{c-1}{2}$. Hence, there must be one task $\tau \in T$ such that $\sum_{\ell=0}^{D-1} d_{\tau, \ell} \geq D \frac{c-1}{2}$. We consider the following realization: The task τ creates packets and any other task τ' creates packets if and only if it shares an edge with τ and on this edge τ' has a higher priority than τ . We call the latter tasks the *blocking tasks*. By definition of I we note that if two blocking tasks share an edge then this edge is used by τ as well. In the realization we can find suitable timesteps for the creation of the packets such that a packet created by τ is delayed $D \frac{c-1}{2}$ times. Hence, the limit guaranteed for τ has to be at least $D \frac{c-1}{2} \in \Omega(c \cdot D)$. \square

Finally, we prove that if the tasks have arbitrary period lengths p_i , in general we cannot obtain schedules guaranteeing good limits for all tasks like in $EPRIO(I)$.

Theorem 4.9. *For every $\alpha, \epsilon > 0$ there is an instance $I = (G, T)$ of the sporadic PPRP with arbitrary period lengths on a path such that for any edge-periodic schedule there is a task $\tau_i \in T$ whose limit is at least $\alpha \cdot p_i + D_i$. Furthermore, for each edge e it holds that $\sum_{\tau_i \in T_e} \frac{1}{p_i} \leq \frac{1}{\alpha} + \epsilon$.*

Proof. Define $k := \frac{\alpha^5}{\epsilon} + 1$. We define G to be a path with k vertices v_0, v_1, \dots, v_k . We define k *big tasks* τ_i ($i = 0, \dots, k-1$) with start vertex $s_i = v_i$, destination vertex $t_i = v_{i+1}$, and period length $p_i = \alpha$. Also, we define a set of α^2 identical *small tasks*, all with start vertex $s_j = v_0$, destination vertex $t_j = v_k$, and period length $p_j = \frac{\alpha^2}{\epsilon}$ (with $j = k, \dots, k + \alpha^2 - 1$). We claim that for each edge-periodic schedule there is a task $\tau_i \in T$ whose limit is greater than $\alpha \cdot p_i + D_i$. Assume on the contrary that there is an edge-periodic schedule where this is not true. In this schedule, for each small task $\tau_j \in T$ there can be at most $\alpha \cdot \frac{\alpha^2}{\epsilon}$ edges on which the respective big task has a higher priority than τ_j . Since $k = \frac{\alpha^5}{\epsilon} + 1 = \alpha \cdot \frac{\alpha^2}{\epsilon} \cdot \alpha^2 + 1$ and there are α^2 small tasks, there must be one big task τ_i which has a lower priority than all the small tasks on its edge. Hence, the limit of τ_i is at least $\alpha^2 + 1 \geq \alpha \cdot p_i + D_i$.

Finally, for each edge e we have that $\sum_{\tau_i \in T_e} \frac{1}{p_i} = \frac{1}{\alpha} + \alpha^2 \cdot \frac{\epsilon}{\alpha^2} = \frac{1}{\alpha} + \epsilon$. \square

5. GLOBAL- AND EDGE-PRIORITY SCHEDULES, STRICT PERIODIC SETTING

In this section we study the strict periodic packet routing problem. Since here each packet $M_{i,j+1}$ is created *exactly* p timesteps after $M_{i,j}$, we have more control over the instance. We use this control to weaken the bounds on the congestion which we require for $GPRIO(I)$ and $EPRIO(I)$ in order to guarantee the respective limits. In particular, we show that in the strict periodic setting the two schedules guarantee the bounds stated in Theorems 4.1 and 4.2 already if $c \leq p/2$ (and not only if $c \leq p/3$ or $c \leq 2p/5$, respectively). Then, we show that if we weaken the definition of a global-priority schedule we can obtain *quasi-global-priority schedules* for instances on directed trees which guarantee a limit of $p/2 + D_i - 1$ for each task τ_i if $c \leq p/2$. Note that this is better than our global-priority schedule $GPRIO(I)$ which does not guarantee a better bound than $2c + D_i - 2$, not even in the strict periodic setting.

First, we give our theorems for $GPRIO(I)$ and $EPRIO(I)$ in the strict periodic setting.

Theorem 5.1. *Let I be an instance of the strict PPRP on a bidirected tree with $c \leq p/2$. Then $GPRIO(I)$ guarantees a limit of $2c + D_i - 2$ for each task τ_i .*

Theorem 5.2. *Let I be an instance of the strict PPRP on a directed tree with $c \leq p/2$. Then $EPRIO(I)$ guarantees a limit of $\frac{3}{2}c + D_i - 1$ for each task τ_i .*

Before we can prove the two theorems, we need to study the behaviour of packets on their way up in $GPRIO(I)$ and $EPRIO(I)$. First, we show that if an up-arc e is used by a packet M at a time t , then at time $t + p$ the arc e is used by a packet M' whose priority is not lower than the priority of M .

Lemma 5.3. *Let I and \bar{I} be instances of the strict PPRP on a bidirected tree and a directed tree, respectively. If in $GPRIO(I)$ or $EPRIO(\bar{I})$ an up-arc e is used by a packet created by a task τ_i at time t then at time $t + p$ it is used by a packet by a task τ_j with $\tau_j \preceq \tau_i$.*

Proof. In order to prove the claim of the lemma, we show the following even stronger statement. Fix an arc e and a point in time t . Denote by $M_1 \preceq M_2 \preceq \dots \preceq M_k$ the packets which wait to use e at time t (we use the short notation $M \preceq M'$ for two packets M, M' if for their corresponding tasks τ, τ' it holds that $\tau \preceq \tau'$). Denote by $M'_1 \preceq M'_2 \preceq \dots \preceq M'_{k'}$ the packets which wait to use e at time $t + p$. We show that then $k' \geq k$ and $M'_i \preceq M_i$ for all i with $1 \leq i \leq k$. Observe that this statement implies the statement of the lemma since M_1 and M'_1 traverse e at times t and $t + p$, respectively.

W.l.o.g. we assume that the first p arcs of the path of each task τ are only used by τ . Note that on these arcs the claim is clear. In the sequel, we show the claim for the other arcs.

For an $e = (u, v)$ we define $h(e) := h(u)$. We prove the claim by induction over $h(e)$. If u is a leaf then the claim is clear. So now assume that the claim holds for all arcs e' with $h(e') \geq k$. Now consider an arc e with $h(e) = k - 1$.

We show the claim by induction over t . The arc e is not used before time $t = p$ (since the first p arcs of each path are only used by one task and we now show the claim for the remaining arcs). Thus, the claim trivially holds for all timesteps $t < p$. So now assume that there is a value t^* such that the claim holds for e for all timesteps $t \leq t^*$. We need to show that the claim then also holds for timestep

$t = t^* + 1$. We compare the waiting packets at timestep $t^* + 1$ with the corresponding packets at the timestep $t^* + 1 + p$. As above, denote by $M_1 \preceq M_2 \preceq \dots \preceq M_k$ and $M'_1 \preceq M'_2 \preceq \dots \preceq M'_{k'}$ the packets waiting for using e at times t^* and $t^* + p$, respectively. Then, the packets M_1 and M'_1 , respectively, traverse e .

Let e_1, \dots, e_ℓ denote the ingoing arcs of u . Denote by $\bar{M}_1 \preceq \bar{M}_2 \preceq \dots \preceq \bar{M}_m$ and $\bar{M}'_1 \preceq \bar{M}'_2 \preceq \dots \preceq \bar{M}'_{m'}$ the packets which traverse e_1, \dots, e_ℓ at time t^* and which need to use e . Since we assumed the claim to be true for all arcs e' with $h(e') \geq k$ we have that $m' \geq m$ and $\bar{M}_i \preceq \bar{M}'_i$ for all i with $1 \leq i \leq m$. Recall that we prioritized the tasks by their height and that G is an intree. In particular, this implies that if \bar{M}_i needs to use e then so does \bar{M}'_i . Thus, at times $t^* + 1$ and $t^* + 1 + p$ the packets $M_2, \dots, M_k, \bar{M}_1, \dots, \bar{M}_m$ and the packets $M'_2, \dots, M'_{k'}, \bar{M}_1, \dots, \bar{M}_{m'}$ wait to use e , respectively. Since $M_i \preceq \bar{M}_i$ and $\bar{M}_i \preceq \bar{M}'_i$ for the respective values i this proves the claim. \square

Next, we show a monotonicity property for the delay of any two packets $M_{i,j}$ and $M_{i,j'}$ with $j' > j$ on their way up.

Lemma 5.4. *Let I and \bar{I} be instances of the strict PPRP on a bidirected tree and a directed tree, respectively. If in $GPRIO(I)$ or $EPRIO(\bar{I})$ a packet $M_{i,j}$ is delayed k times before traversing an up-arc $e \in P_i$ then each packet $M_{i,j'}$ with $j' \geq j$ is delayed at least k times before traversing e .*

Proof. We show the claim by induction over the arcs of $P_i = (e_1, e_2, \dots, e_m)$. In order to simplify the proof we assume w.l.o.g. that e_1 is used only by τ_i . Thus, for e_1 the claim is obvious. So now assume that the claim holds for all arcs up to arc e_ℓ . We need to show that it also holds for $e_{\ell+1}$. Consider two packets $M_{i,j}$ and $M_{i,j'}$ with $j' \geq j$. Assume that $M_{i,j}$ was delayed k_ℓ times before traversing e_ℓ and $k_{\ell+1}$ times before traversing $e_{\ell+1}$. This implies that during the time interval $[j \cdot p + k_\ell + \ell, j \cdot p + k_{\ell+1} + \ell)$ the arc $e_{\ell+1}$ is used by packets created by tasks with higher priority than τ_i . Due to Lemma 5.3 this implies that the same holds for the interval $[j' \cdot p + k_\ell + \ell, j' \cdot p + k_{\ell+1} + \ell)$. From the induction hypothesis we conclude that $M_{i,j'}$ is delayed at least k_ℓ times before traversing e_ℓ . We conclude that also $M_{i,j'}$ is delayed at least $k_{\ell+1}$ times before traversing $e_{\ell+1}$. \square

Now we can prove Theorems 5.1 and 5.2.

Proof of Theorem 5.1: Consider a task τ_i with peak vertex v . From Lemma 5.4 we conclude that two packets created by τ_i have a minimum time distance of c when reaching v (without requiring a bound for c). With similar arguments as in Theorem 4.1 we can prove that then on the way away from v_r each packet is delayed no more than $c - 1$ times. The induction hypothesis is that on the way away from v_r two packets created by the same task have a minimum distance of $p - c \geq p/2$. Thus, in the strict periodic setting $GPRIO(I)$ guarantees a limit of $2c + D_i - 2$ for each task τ_i if $c \leq p/2$. \square

Similarly, we can prove the bound for $EPRIO(I)$ in the strict periodic setting assuming that $c \leq p/2$.

Proof of Theorem 5.2: Consider a task τ_i with peak vertex v . From Lemma 5.4 we conclude that two packets created by τ_i have a minimum time distance of c when reaching v (again without requiring a bound for c). We can argue that then on the way down each packet is delayed no more than $c - 1$ times and hence in the strict

periodic setting $EPRIO(I)$ guarantees a limit of $\frac{3}{2}c + D_i - 1$ for each task τ_i if $c \leq p/2$. \square

5.1. Quasi-Global-Priority Schedules. In this section we weaken the definition of global-priority schedules as follows: We no longer require that we have a priority relation \prec for the tasks which is a total order. Instead, we require only that the restriction of \prec to the tasks which created the packets which wait for using any arc e at any time t is a total order. This is justifiable since if we want to decide what packet we want to give priority we will consider only the tasks which created packets waiting to use e . If for these tasks \prec gives a total order it is well-defined what task has highest priority.

Definition 5.5 (Quasi-global-priority Schedules). Let $I = (G, T, p)$ be an instance of the strict periodic packet routing problem. We call a schedule S for I a *quasi-global-priority schedule* if there is a relation $\prec \subseteq T \times T$ with the following properties: Let e be an arc, let t be a timestep, and let $T_{e,t} \subseteq T_e$ be the tasks which correspond to the packets which wait to use e at time t .

- The relation $\prec_{e,t} := \prec \cap (T_{e,t} \times T_{e,t})$ is a total order.
- If there is at least one packet waiting to use e at timestep t , a packet uses e at time t whose corresponding task τ has highest priority according to $\prec_{e,t}$. Formally, $\tau \succeq_{e,t} \tau'$ for any other task τ' that created a packet waiting to use e .
- If the task with highest priority according to $\prec_{e,t}$ has created several packets which are waiting to use e then the packet moves first which was created first.

Assume we are given an instance of the strict PPRP on a directed tree with $c \leq p/a$ for an integer $a \geq 2$. We present an algorithm which computes a quasi-global-priority schedule $QPRIO(I)$ which guarantees a limit of $\lceil p/a \rceil + D_i - 1$ for each task τ_i . This is better than our schedule $GPRIO(I)$ which guarantees only a bound of $2c + D_i - 2$. Also recall that there are instances in the strict periodic setting for which $GPRIO(I)$ does not perform better than this and in the sporadic setting no global-priority schedule can guarantee a better bound than $GPRIO(I)$.

Algorithm $QPRIO(I)$

- (1) Compute direct template schedule with initial delay $w_i < p/a$ for each task τ_i .
- (2) Define priority relation \prec_g such that a packet created by a task τ_i does not wait if it has been delayed w_i times already.

Intuitively, the schedule $QPRIO(I)$ imitates the template schedule $DTREE(I)$. We compute a path coloring for the paths of the tasks using at most p/a colors. Then we compute a time-dependent edge-coloring $g : E \times \{0, \dots, p-1\} \rightarrow \{0, \dots, \lceil p/a \rceil - 1\}$: We start with an arbitrary arc e^* and define the map g such that for each value $k \in \{0, \dots, \lceil p/a \rceil - 1\}$ and each interval $J \subseteq \mathbb{N}$ with $|J| \geq \lceil p/a \rceil$ we have that $k \in g(e^*, J \bmod p)$ (where $(J \bmod p)$ denotes the set $\{j \bmod p \mid j \in J\}$). The values $g(e, i)$ for the other arcs are then obtained from the consecutive property (see Section 3.1). Denote by $DTREE_g(I)$ the template schedule obtained from g and the path coloring. Note that since we are in the strict periodic setting in this

schedule all packets created by a task τ_i have the same initial delay. Denote by w_i this delay. From the definition of our time-dependent coloring we conclude that for each task τ_i we have that $w_i < p/a \leq p/2$.

We derive a quasi-global-priority schedule in which no packet suffers more delay than in $DTREE_g(I)$. We define our prioritization as follows: Let τ_i and τ_j be two tasks such that P_i and P_j both use an arc $e = (u, v)$. Let ℓ_i and ℓ_j the distance between s_i and u , and s_j and u , respectively. Now consider the families of intervals $A_{e,i}^{(k)} = [\ell_i + k \cdot p, \ell_i + w_i + k \cdot p]$ and $A_{e,j}^{(k)} = [\ell_j + k \cdot p, \ell_j + w_j + k \cdot p]$. Note that the packet $M_{i,k}$ arrives at u at time $\ell_i + k \cdot p$ if it is never delayed. In $DTREE(I)$ it arrives at u at time $\ell_i + w_i + k \cdot p$. The respective statements hold for $M_{j,k}$.

Recall that $w_i < p/2$ and $w_j < p/2$. This implies that $w_i + w_j < p$. Hence, each interval from one of the families can intersect with at most one interval of the other family. We define the relation \prec_g such that in $QPRIO(I)$ each packet $M_{i,k}$ is located on u only within the time interval $A_{e,i}^{(k)}$: If no two intervals of the two families $A_{e,i}^{(k)}$ and $A_{e,j}^{(k')}$ intersect then we define $\tau_i \prec_g \tau_j$ or $\tau_j \prec_g \tau_i$ arbitrarily. Now assume that there are values k and k' such that $A_{e,i}^{(k)}$ and $A_{e,j}^{(k')}$ intersect. Note that $\ell_i + w_i + k \cdot p \neq \ell_j + w_j + k' \cdot p$ since otherwise $DTREE(I)$ would be invalid. If $\ell_i + w_i + k \cdot p < \ell_j + w_j + k' \cdot p$ we define $\tau_i \prec_g \tau_j$, otherwise we define $\tau_j \prec_g \tau_i$.

Note that in general this does not yield a transitive ordering. However, we will show in the sequel that at any point in time the restriction of \prec_g to the tasks which created the packets which wait for using an arc is indeed transitive. Hence, the schedule resulting from \prec_g is well-defined. Denote by $QPRIO(I)$ this schedule.

Theorem 5.6. *Let I be an instance of the strict PPRP on a directed tree with $c \leq p/a$ for an integer $a \geq 2$. The schedule $QPRIO(I)$ is a quasi-global-priority schedule which guarantees a limit of $\lceil p/a \rceil + D_i - 1$ for each task τ_i .*

Before we can prove the theorem we need two lemmata.

Lemma 5.7. *Let I be an instance of the strict PPRP on a directed tree and let τ_i be a task which uses an arc $e = (u, v)$. In the schedule $QPRIO(I)$ each packet $M_{i,k}$ is located on u only within the time interval $A_{e,i}^{(k)}$.*

Proof. We show the claim by induction. W.l.o.g. we assume that the first arc on the path of each task τ_i is used only by τ_i . For the base case let $e = (u, v)$ be an arc such that u is a leaf. Since e is used by only one task the claim is clear.

For the inductive step consider an arc $e = (u, v)$ and suppose the claim is true for all arcs in the subtree below e (i.e., the subtree which is obtained by removing e and then removing the tree rooted at v).

Let I_e denote the set of all tasks using e . For each task τ_i let ℓ_i denote the distance between s_i and u (like above). By induction we know that each packet $M_{i,k}$ arrives on u within the time interval $A_{e,i}^{(k)} = [\ell_i + k \cdot p, \ell_i + w_i + k \cdot p]$. We show by induction for each point in time t that if a packet $M_{i,k}$ is located on u at time t then $t \in A_{e,i}^{(k)}$. For $t = 0$ the claim is true since $\ell_i > 0$ for all tasks τ_i and thus there are no packets on u . So now assume that the claim is true for all $t \leq t^*$ and consider $t := t^* + 1$. Assume that a packet $M_{i,k}$ is located on u at time t . We want to show that then $t \in A_{e,i}^{(k)}$. We distinguish two cases. First assume that $M_{i,k}$ was located on u at time $t - 1$ as well. Then by induction we know that for all packets $M_{i',k'}$ located at u at time $t - 1$ it holds that $(t - 1) \in A_{e,i'}^{(k')}$. Thus, by

definition of the schedule, if $t \notin A_{e,i}^{(k)}$ then $M_{i,k}$ would have been transferred over e at time $t - 1$. So now assume that $M_{i,k}$ was not located on u at time $t - 1$, i.e., arrived on u at time t . However, above we already derived that $M_{i,k}$ arrives on u within $A_{e,i}^{(k)}$. This completes the proof. \square

Lemma 5.8. *Let I be an instance of the strict PPRP, let t be a timestep and let e be an arc. Let $\mathcal{M}_{e,t}$ be the set of packets which wait to use e at timestep t in the schedule $QPRIO(I)$. Let $T_{e,t}$ denote the tasks which created the packets $\mathcal{M}_{e,t}$. Then, the relation $\prec_g \cap T_{e,t} \cap T_{e,t}$ is transitive relation.*

Proof. Follows from Lemma 5.7 and the definition of \prec_g . \square

Proof of Theorem 5.6: Lemma 5.8 implies that $QPRIO(I)$ is a quasi-global-priority schedule resulting from the relation \prec_g . Due to Lemma 5.7 we know that in $QPRIO(I)$ no packet is delayed more often than in $DTREE(I)$. Since $w_i < p/a$ for each task τ_i we conclude that each packet is delayed at most $\lceil p/a \rceil - 1$ times which implies the stated bound. \square

6. IMITATION THEOREMS

In the previous sections we studied template schedules as well as global-priority and edge-priority schedules. It seems as if template schedules in general are more powerful than priority schedules. In this section we show that any global-priority schedule on any graph can be imitated by a template schedule. Also, we show that any edge-priority schedule on a bidirected tree can be imitated by a template schedule. With “imitate” we mean that we can find a template schedule which guarantees the same limits for each task as the respective priority schedule in the strict periodic case and almost the same limits in the sporadic case.

We prove the result by showing that in the strict-periodic setting global schedules on general graphs and edge-priority schedules on bidirected trees behave periodically after a certain time (and thus operate like template schedules). This allows us to state template schedules which match this periodic behaviour. Finally, we show that there are edge-priority schedules on cycle graphs for the strict-periodic setting which never behave periodically. Thus, they cannot directly be imitated by template schedules.

6.1. Template Schedules and Global-Priority Schedules. Given an instance I of the strict periodic packet routing problem on an arbitrary graph. Let S be a global-priority schedule for I . We show how to construct a template schedule S_t which imitates the schedule S . First, we prove that S behaves periodically after a certain time.

Lemma 6.1. *Let I be an instance of the strict PPRP on an arbitrary graph. Let S be a global-priority schedule for I . For each task τ_i and each arc $e \in P_i$ there is a timestep k_i and an offset $a_{e,i}$ such that for all timesteps $k \geq k_i$ we have that*

- S transports a packet of τ_i over e if $k \equiv a_{e,i} \pmod{p}$ and
- S does not transport a packet of τ_i over e if $k \not\equiv a_{e,i} \pmod{p}$

Proof. We prove the claim by induction over the tasks. Assume the tasks are ordered such that $\tau_1 \prec_S \tau_2 \prec_S \dots \prec_S \tau_{|T|}$. W.l.o.g. we assume that the first arc of the path of each task is only used by this one task. For the task τ_1 we can easily

find a value k_1 (e.g., $k_1 := |P_1|$) and suitable offsets such that statement in the lemma holds.

Now we assume for the induction hypothesis that for all tasks τ_1, \dots, τ_m we have values k_i and suitable time offsets with the properties of the lemma statement. Consider the task τ_{m+1} . We show the claim for τ_{m+1} by another induction over the arcs of $P_{m+1} = (e_1, \dots, e_s)$. For e_1 the claim is trivial (since by assumption only τ_{m+1} uses this arc). Assume inductively that we have found a value k_{m+1}^r such that for all e_j with $j \leq r$ we have found offsets as stated in the lemma which hold for all timesteps k with $k \geq k_{m+1}^r$.

Now consider the arc e_{r+1} . We define $\tilde{k} := \max\{k_1, \dots, k_m, k_{m+1}^r\}$. First assume that there are $p-1$ other tasks which use e_{r+1} and which have a higher priority than τ_{m+1} in S . Then we define $k_{m+1}^{r+1} := \tilde{k}$ and $a_{e_{r+1}, m+1}$ is defined such that $a_{e_{r+1}, m+1} \neq a_{e_{r+1}, i'}$ for all tasks $\tau_{i'}$ which also use e_{r+1} . Now assume that on e_{r+1} there are at most $p-2$ tasks which have a higher priority than τ_{m+1} . Then, from time \tilde{k} on we know from the induction hypothesis that exactly every p timesteps a new packet from τ_{m+1} arrives and needs to use e_{r+1} . However, there might be some other packets created by τ_{m+1} waiting to use e_{r+1} which could not use e_{r+1} yet. During the time interval $[0, \tilde{k}]$ exactly $\lfloor \frac{\tilde{k}}{p} \rfloor$ packets were created by τ_{m+1} . At least two packets created by τ_{m+1} can use e_{r+1} in each time interval $[\tilde{k} + v \cdot p, \tilde{k} + (v+1) \cdot p - 1]$, for $v \in \mathbb{N}$. Thus, if we define $k_{m+1}^{r+1} := 2\tilde{k}$ we can find a suitable offset $a_{e_{r+1}, m+1}$ which satisfies the statement in the lemma. Finally, we define $k_{m+1} := k_{m+1}^s$. \square

Now we can derive a template schedule S_t which emulates S and thus guarantees the same limits as S .

Theorem 6.2. *Let I be an instance of the strict PPRP on an arbitrary graph. For each global-priority schedule S for I there is a template schedule S_t which guarantees the same limit as S for each task.*

Proof. We construct the template schedule S_t from the offsets stated in Lemma 6.1. Consider a task τ_i . The lemma implies that all packets which were created by τ_i after timestep k_i cannot reach their respective destination faster in S than in S_t . \square

Corollary 6.3. *Let $I = (G, T, p)$ be an instance of the sporadic PPRP on an arbitrary graph G . Let S be a global-priority schedule for I which guarantees a limit of k_i for each task τ_i . There is a template schedule which guarantees a limit of $k_i + p$ for each task τ_i .*

6.2. Template Schedules and Edge-Priority Schedules on Bidirected Trees.

Given an instance I of the strict periodic packet routing problem on a bidirected tree G . Let S be an edge-priority schedule for I . We show how to construct a template schedule S_t which guarantees the same limits for the tasks as S . Like in Lemma 6.1 we show that S behaves periodically after a certain time.

Since our tree is bidirected here we will interpret each arc as two directed arcs. We assign each arc a *level*. For an arc $e = (u, v)$ which is oriented *away from* v_r we define $level(e)$ to be the distance between v_r and v . For an arc $e = (u, v)$ which is oriented *towards* v_r we define $level(e)$ to be the distance between u and v_r multiplied by -1 . Figure 6.1 shows a sketch which yields some intuition for the definition.

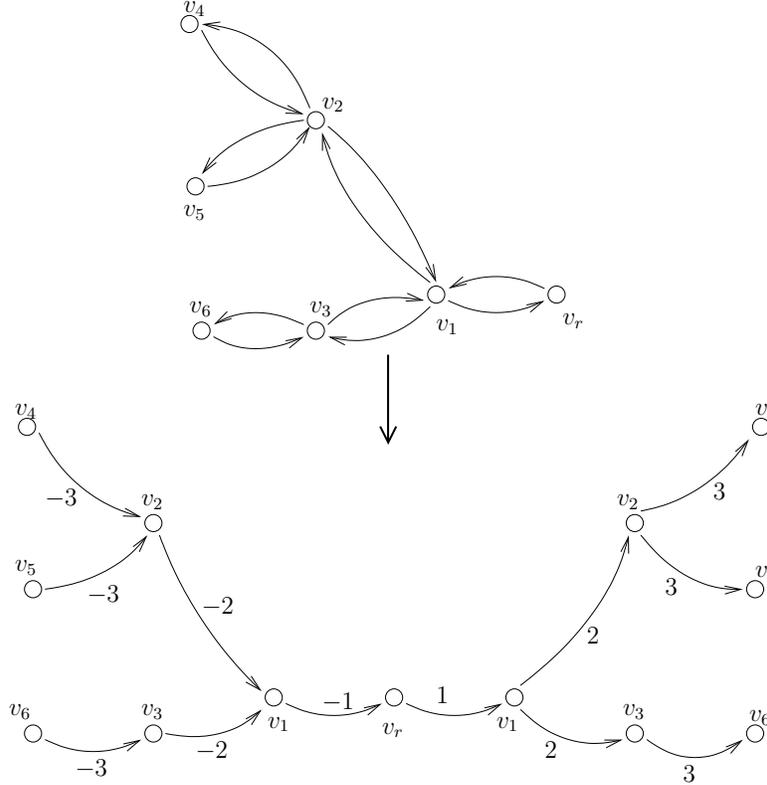


FIGURE 6.1. The lower figure depicts the levels of the arcs of the tree above. Note that in the lower figure all vertices (but v_r) appear twice.

Note that if an arc e is the antiparallel arc to an arc e' then $level(e) = -level(e')$. Moreover, there is no arc e with $level(e) = 0$.

Lemma 6.4. *Let I be an instance of the strict PPRP on a bidirected tree. Let S be an edge-priority schedule for I . For each arc $e \in P_i$ and each task τ_i there is a timestep k_e and an offset $a_{e,i}$ such that for all timesteps $k \geq k_e$ we have that*

- S transports a packet of τ_i over e if $k \equiv a_{e,i} \pmod p$ and
- S does not transports a packet of τ_i over e if $k \not\equiv a_{e,i} \pmod p$

Proof. We assume w.l.o.g. that the first arc on the path of each task is used only by this one task. We prove the lemma by induction over the levels of the arcs, starting with the arcs in the level with lowest index. Let e be such an arc. Due to our assumption above there is exactly one task τ_i which uses e and thus we can choose $k_e = 0$ and $a_{e,i} = 0$.

Now assume that by induction the claim is true for all arcs e with $level(e) \leq m$. Now $e = (u, v)$ be an arc with $level(e) = m + 1$. If u is a leaf then there can be at most one task using e . The claim is then shown as in the base case. So now assume that u is not a leaf. Let e_1, \dots, e_s denote all ingoing arcs of u . Due to the induction hypothesis the claim is true for each of these arcs. Let $k := \max \{k_{e_1}, \dots, k_{e_s}\}$.

Assume w.l.o.g. that e is used by the tasks $T_e := \{\tau_1, \dots, \tau_r\}$ with the priority order $\tau_1 \prec \tau_2 \prec \dots \prec \tau_r$. We show the claim by induction over these tasks. Since τ_1 has highest priority among the tasks in T_e there is an offset $a_{e,1}$ with the properties stated above for all k with $k \geq \tilde{k} =: k_e^1$. Assume for induction that we have such offsets for all tasks $\tau_i \in T_e$ with $i \leq m$ such that the claim holds for all timesteps $k \geq k_m^e$. Now consider the task τ_{m+1} . If $m+1 = p$ then all other tasks in T_e have a higher priority and there is only one offset $a_{e,m+1} \in \{0, \dots, p-1\}$ left for τ_{m+1} . Thus, the lemma statement holds with $a_{e,m+1}$ for all timesteps $k \geq k_{m+1}^e := k_m^e$. Now assume that $m+1 < p$. Then, in each interval $[k_m^e + i \cdot p, k_m^e + (i+1) \cdot p - 1]$ (for all $i \in \mathbb{N}$) at least two packets created by τ_{m+1} can use e . In the first k_m^e timesteps $\lfloor \frac{k_m^e}{p} \rfloor$ packets were created by τ_{m+1} . With the induction hypothesis we conclude that there is an offset $a_{e,m+1}$ such that the lemma holds for all timestep $k \geq k_{m+1}^e := 2k_m^e$. Finally, we define $k_e := \max_i k_e^i$. \square

Theorem 6.5. *Let I be an instance of the strict PPRP on a bidirected tree. For each edge-priority schedule S for I there is a template schedule S_t which guarantees the same limit for each task as S .*

Proof. Follows from Lemma 6.4 (similar proof as for Theorem 6.2). \square

Corollary 6.6. *Let $I = (G, T, p)$ be an instance of the sporadic PPRP on a bidirected tree G . Let S be an edge-priority schedule for I which guarantees a limit of k_i for each task τ_i . There is a template schedule which guarantees a limit of $k_i + p$ for each task τ_i .*

6.3. Template Schedules and Edge-Priority Schedules on General Graphs.

In the following theorem we show that there are edge-priority schedules for the strict PPRP on a cycle graph which do not behave periodically after any point in time. Hence, we cannot prove a lemma similar to Lemma 6.1 or Lemma 6.4 for this setting. This implies that the schedule cannot be directly emulated by a template schedule.

Theorem 6.7. *There is an instance I of the strict periodic PPRP on a cycle graph and an edge-priority schedule S for I with the following property: there is no timestep t such that there is a template schedule for I which is identical to S after time t .*

Proof. The instance I is depicted in Figure 6.2: we have a cycle graph with four vertices v_1, \dots, v_4 and two tasks τ_1, τ_2 with $P_1 = (v_1, v_2, v_3, v_4)$ and $P_2 = (v_3, v_4, v_1, v_2)$. We define $p := 2$. In the schedule S the task τ_1 has a higher priority than τ_2 on the arc (v_3, v_4) . The opposite prioritization holds on the arc (v_1, v_2) .

Assume on the contrary that there is a time t such that there is a template schedule S_t which behaves identically to S after time t . Since $p = 2$ it suffices to describe S_t by specifying whether a task uses an arc at even or odd timesteps. We distinguish two cases: First assume that the arc (v_2, v_3) is used by τ_1 at odd timesteps. Since the construction is symmetric, this implies that the arc (v_4, v_1) is used by the task τ_2 at odd timesteps. Due to the definition of S this implies that τ_1 uses the arc (v_3, v_4) at even timesteps and thus, every packet of τ_2 is delayed once in v_3 . However, this implies that τ_2 uses the arc (v_4, v_1) at even timesteps which is a contradiction. Now we assume that the arc (v_2, v_3) is used by τ_1 at even timesteps. By symmetry, this implies that (v_4, v_1) is used by τ_2 at even timesteps.

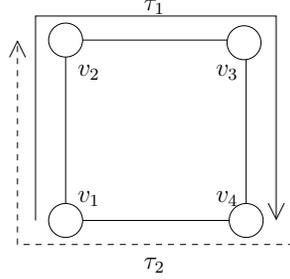


FIGURE 6.2. The instance I described in the proof of Theorem 6.7.

Hence, τ_2 uses (v_1, v_2) at odd timesteps. We conclude that τ_1 uses (v_2, v_3) at odd timesteps which is a contradiction. Thus, there can be no template schedule for I which is identical to S after any time t . \square

7. EDGE-PRIORITY ROUTING WITH SWING IN

In Section 5 we studied the schedules $GPRIO(I)$ and $EPRIO(I)$ in the strict periodic setting. In order to guarantee the stated limits we required that $c \leq p/2$. On the other hand, the template schedules $DTREE(I)$ and $BTREE(I)$ do not need any bound on c (apart from $c \leq p$). In this section we investigate whether one can also derive edge-priority schedules for the strict periodic setting with good limits for all tasks without any further restriction on the congestion. We achieve this by assigning priorities to the first packets created by each task independently from the prioritization of all remaining packets. This can also be understood as first running an arbitrary schedule for a fixed number of timesteps and then switching over to an edge-priority schedule. This bypasses the early time period during which edge-priority schedules behave differently than after they have swung in (compare Lemma 6.4). We will see that in this framework we can derive edge-priority schedules which guarantee the same bounds as our template schedules without requiring any bound on the congestion apart from $c \leq p$.

Definition 7.1 (Edge-Priority Schedule with Swing-In). Let $I = (G, T, p)$ be an instance of the strict PPRP. A periodic schedule S for I is an *edge-priority schedule with a swing-in of t_0* if it fulfills the properties of edge-priority schedules (see Definition 1.2) from timestep t_0 on.

Given an instance I of the strict PPRP on a directed tree G . W.l.o.g. we assume that $c = p$. In the sequel we describe how we can transform the schedule $DTREE(I)$ to an edge-priority schedule with swing-in. We show that there is a relation \prec_e for each arc e such that the transformed schedule prioritizes according to the relations \prec_e after $c + D - 1$ timesteps with $D := \max_i D_i$.

Let $task$ be the map which yields $DTREE(I)$. Now we want to imitate this schedule by an edge-priority schedule with swing in. For this, we modify the map $task$ inductively. We assign levels to the arcs like in Section 6.2. We iterate over the arcs from the lowest to the highest level. While doing that, we modify the map $task$. Let $e = (u, v)$ be an arc. For each task $\tau \in T_e$ we define values $a_e(\tau)$ and $\ell_e(\tau)$. These values denote the timesteps (modulo p) when the packets of τ arrive at u and leave u , respectively according to the current map $task$. Now we decrease

the value $\ell_e(\tau)$ for each task $\tau \in T_e$ simultaneously modulo p until there is a task τ such that $a_e(\tau) = \ell_e(\tau)$.

We fix all tasks τ for which we have that $a_e(\tau) = \ell_e(\tau)$ and also fix the corresponding values $a_e(\tau)$. We continue to decrease the value $\ell_e(\tau)$ of each unfixed task τ . However, we skip values which we already fixed, i.e., if for a task τ the values $\ell_e(\tau) - 1, \dots, \ell_e(\tau) - j \bmod p$ are fixed then in the decreasing step we set $\ell_e(\tau)$ to $\ell_e(\tau) - j - 1 \bmod p$. Also, if after one decreasing step for a task τ the values $a_e(\tau), \dots, a_e(\tau) + j \bmod p = \ell_e(\tau) - 1 \bmod p$ are fixed then we also fix τ and $\ell_e(\tau)$.

We define a map $task'$ such that $task'(e, t) = \tau$ if and only if $\ell_e(\tau) = t$. Denote by $DTREE(I)'$ the schedule resulting from the map $task'$. We will show in the sequel that $DTREE(I)'$ is indeed an edge-priority schedule with swing-in of $p + D - 1$. We will show this by giving a suitable relation \prec_e for each arc e .

Theorem 7.2. *Let I be a strict periodic instance of the PPRP on a directed tree with $c \leq p$. There is an edge-priority schedule with a swing-in of $p + D - 1$ which guarantees a limit of $p + D_i - 1$ for each task τ_i .*

With a similar procedure one can transform the schedule $BTREE(I)$ into an edge-priority schedule with a swing-in of $2p + D - 1$.

Theorem 7.3. *Let I be a strict periodic instance of the PPRP on a bidirected tree with $c \leq p$. There is an edge-priority schedule with a swing-in of $D + 2p - 1$ which guarantees a limit of $2p + D_i - 1$ for each task τ_i .*

We will now prove Theorem 7.2. Before we can prove the theorem we need to show three lemmata. Theorem 7.3 can be shown similarly.

Lemma 7.4. *Let I be a strict periodic instance of the PPRP on a directed tree with $c \leq p$. In the schedule $DTREE(I)'$ no packet reaches its destination later than in $DTREE(I)$.*

Proof. In our modifications we have the following invariant: Let e and e' be two consecutive arcs on the path of τ . Then, $a_{e'}(\tau) = \ell_e(\tau) + 1$. The total waiting time of τ in e and e' is given by $(\ell_e(\tau) - a_e(\tau)) \bmod p + (\ell_{e'}(\tau) - a_{e'}(\tau)) \bmod p$. Our modifications are defined such that this value does not change when we modify $\ell_e(\tau)$. When modifying the value $\ell_e(\tau)$ for the last arc $e \in P_i$ (for which there is no corresponding arc e') the total delay of τ might decrease but not increase. \square

Now, we define a relation \prec_e for each arc e in order to show that $DTREE(I)'$ is indeed an edge-priority schedule with swing-in of $c + D - 1$. Let $e = (u, v)$ be an arc. For each task τ we define the values $a_e(\tau)$ and $\ell_e(\tau)$ from the map $task'$. The idea for the schedule later is that the packets created by τ arrive at u at timesteps t with $t \bmod p = a_e(\tau)$ and leave u at timesteps t with $t \bmod p = \ell_e(\tau)$. We will define \prec_e such that this is ensured. For each task τ let $A_e(\tau)$ be

- the set $\{a_e(\tau), \dots, \ell_e(\tau)\}$ if $a_e(\tau) \leq \ell_e(\tau)$ and
- the set $\{a_e(\tau), \dots, p - 1\} \cup \{0, \dots, \ell_e(\tau)\}$ if $a_e(\tau) > \ell_e(\tau)$

Intuitively, the set $A_e(\tau)$ contains the set of timesteps (modulo p) in which packets created by τ are located on u . If $a_e(\tau) \neq \ell_e(\tau)$ then we define $\tau' \prec_e \tau$ for all tasks τ' such that $\ell_e(\tau') \in A_e(\tau)$. We do this for all arcs e .

Lemma 7.5. *For each arc e the relation \prec_e is transitive.*

Proof. Assume on the contrary that there is an arc e such that \prec_e is not transitive. I.e., there are tasks $\tau_1, \tau_2, \dots, \tau_k$ such that $\tau_1 \prec_e \tau_2 \prec_e \dots \prec_e \tau_k \prec_e \tau_1$. Recall that we simultaneously decreased the values $\ell_e(\tau)$ and fixed them at some point. Assume w.l.o.g. that τ_1 was the first task among the above which was fixed. However, this implies that all values $A_e(\tau) \setminus \{\ell_e(\tau)\}$ were fixed at that point. However, since $\tau_k \prec_e \tau_1$ this implies that $\ell_e(\tau_k) \in A_e(\tau_1)$ and thus $\ell_e(\tau_k)$ was not fixed when τ_1 was fixed. Since $\ell_e(\tau_1) \neq \ell_e(\tau_k)$ this implies that $\ell_e(\tau_k) \in A_e(\tau) \setminus \{\ell_e(\tau)\}$ which is a contradiction. \square

Lemma 7.6. *Let I be a strict periodic instance of the PPRP on a directed tree with $c \leq p$. Assume the schedule $DTREE(I)'$ runs for $p + D - 1$ timesteps. After that, $DTREE(I)'$ obeys the prioritization \prec_e at each arc e .*

Proof. Reconsider the maps $a_e(\tau)$ and $\ell_e(\tau)$ defined from the map $task'$. Let $e = (u, v)$ be an arc and let $\tau \in T_e$. For each timestep $t \equiv a_e(\tau) \pmod p$ with $t \geq D + c - 1$ we have that in $DTREE(I)'$ a packet created by τ arrives at u . Similarly, for each timestep $t \equiv \ell_e(\tau) \pmod p$ with $t \geq D + c - 1$ we have that in $DTREE(I)'$ a packet created by τ leaves u .

By definition of \prec_e all tasks whose corresponding packets leave u at timesteps $t \equiv k \pmod p$ with $k \in A_e(\tau) \setminus \{\ell_e(\tau)\}$ have a higher priority than τ . Thus, after $D + c - 1$ timesteps the schedule $DTREE(I)'$ obeys prioritization \prec_e . \square

Proof of Theorem 7.2: Follows from Lemma 7.4, Lemma 7.5, Lemma 7.6, and the limit of $DTREE(I)$ proven in Theorem 3.1. \square

8. COMPLEXITY

In this section we prove NP -hardness results for the problems studied in this paper. First of all, recall our polynomial time algorithm for computing the schedule $DTREE(I)$. The schedule guarantees a limit of $c + D_i - 1$ for each task τ_i . We show that the algorithm is best possible (unless $P = NP$) in the sense that already on instances on directed trees it is NP -hard to determine whether there is a schedule which guarantees a delay of at most $c - 2$ for each task. We will show that this holds for each period length $p \geq 6$. If we allow the graph to have a more general structure we show that the problem becomes even harder: We show that the minimum delay for each task cannot be approximated with a better factor than $|T|^{1-\epsilon}$ for all $\epsilon > 0$ due to a reduction from COLORING. Finally, we show that it is NP -hard to determine whether there is a direct template schedule for an instance on a bidirected or an undirected tree.

Let I denote an instance of the strict PPRP. For a schedule S for I , we denote by $md(S)$ the maximum delay that a task encounters in S . Let $\mathcal{S}(I)$ denote the set of all schedules for I . We define the *minimum delay* $md_{min}(I)$ by $md_{min}(I) := \min_{S \in \mathcal{S}(I)} md(S)$. We show that it is NP -hard to compute $md_{min}(I)$. This holds even on directed trees.

Theorem 8.1. *Let I be an instance of the strict PPRP on a directed tree. It is NP -hard to compute whether $md_{min}(I) \leq c - 2$.*

Proof. We use a similar construction as used in [13] for showing that the (non-periodic) packet routing problem is NP -hard on directed trees. We reduce from 3-BOUNDED-3-SAT.

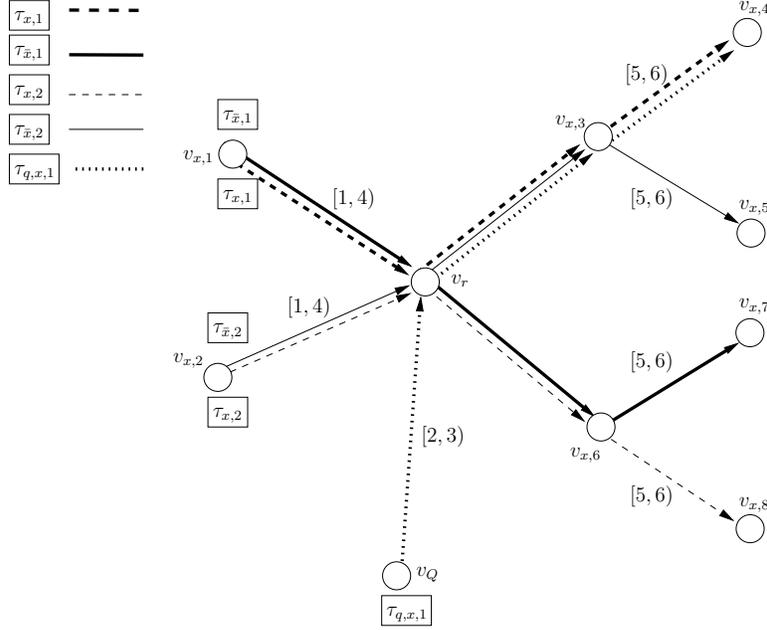


FIGURE 8.1. A part of the graph for the clause $Q = (x \vee y \vee z)$ (construction for Theorem 8.1). The parts corresponding to the variables y and z and their packets are omitted for brevity. The time intervals at the arcs denote the time when the respective arc is blocked. Note that the first timestep is $t = 0$.

Assume we are given a 3-BOUNDED-3-SAT formula ϕ . We construct an instance I of the periodic packet routing problem with fixed paths such that the underlying graph is a directed tree. In I we have that $p = c = 6$. We show that ϕ is satisfiable if and only if $md_{min}(I) \leq 4 = c - 2 = p - 2$.

We assume that in ϕ each clause contains at least two variables and that each variable occurs at most two times positive and at most two times negated. For each variable x in the formula, there are four tasks $\tau_{x,1}$, $\tau_{x,2}$, $\tau_{\bar{x},1}$, and $\tau_{\bar{x},2}$. The instance is defined such that in a schedule S with $md(S) \leq p - 2$ they can be scheduled at times $t \equiv 0 \pmod p$ or $t \equiv 4 \pmod p$. The intuition is that if x_1 and x_2 are scheduled at time $t \equiv 0 \pmod p$ then we set the variable x true, if x_1 and x_2 are scheduled at time $t \equiv 4 \pmod p$ then we set the variable x false.

Note that now it could happen that none of the above cases apply, i.e., the tasks $\tau_{x,1}$ and $\tau_{\bar{x},2}$ are scheduled at time $t \equiv 0 \pmod p$ or the tasks $\tau_{x,2}$ and $\tau_{\bar{x},1}$ are scheduled at time $t \equiv 0 \pmod p$. However, our construction ensures that this leads to a schedule with $md(I) \geq p - 1$. Moreover, for each literal in a clause Q there is one task. Tasks corresponding to literals in the same clause share the first arc on their path. Given a satisfying variable assignment for ϕ there is at least one literal in Q which satisfies the clause. The task which corresponds to this literal is scheduled last (at time $t \equiv 3 \pmod p$).

Now we describe the construction in detail. For each variable x we introduce vertices $v_{x,1}, v_{x,2}, \dots, v_{x,8}$. We also introduce the four *variable tasks* $\tau_{x,1}$, $\tau_{x,2}$, $\tau_{\bar{x},1}$,

Task	Path	
Variable tasks for variable x		
$\tau_{x,1}$	$v_{x,1}, v_r, v_{x,3}, v_{x,4}$	For each variable x
$\tau_{x,2}$	$v_{x,2}, v_r, v_{x,6}, v_{x,8}$	For each variable x
$\tau_{\bar{x},1}$	$v_{x,1}, v_r, v_{x,6}, v_{x,7}$	For each variable x
$\tau_{\bar{x},2}$	$v_{x,2}, v_r, v_{x,3}, v_{x,5}$	For each variable x
Clause tasks for clause C		
$\tau_{q,x,1}$	$v_Q, v_r, v_{x,3}, v_{x,4}$	where C contains the first positive occurrence of x
$\tau_{q,x,2}$	$v_Q, v_r, v_{x,6}, v_{x,8}$	where C contains the second positive occurrence of x
$\tau_{q,\bar{x},1}$	$v_Q, v_r, v_{x,6}, v_{x,7}$	where C contains the first negative occurrence of x
$\tau_{q,\bar{x},2}$	$v_Q, v_r, v_{x,3}, v_{x,5}$	where C contains the second negative occurrence of x

TABLE 3. The predefined paths of the tasks.

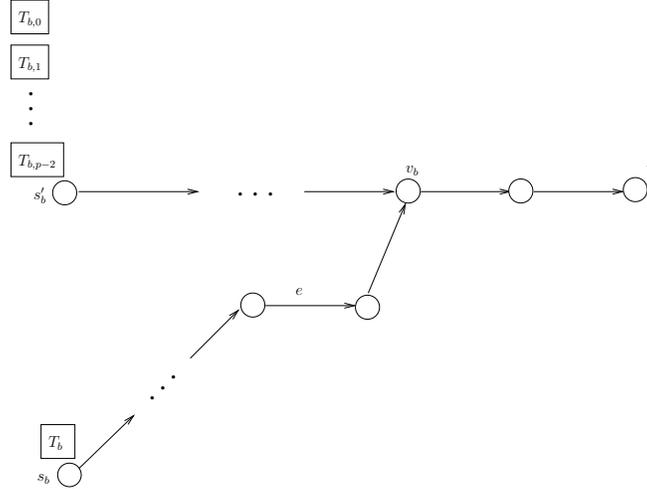


FIGURE 8.2. The paths of the tasks $\tau_b, \tau_{b,0}, \tau_{b,1}, \dots, \tau_{b,p-2}$ are defined such that if τ_b is never delayed before having reached v_b then it will not collide with any delay task $\tau_{b,i}$. However, if τ_b is delayed once, then it will arrive at v_b at the same time as the fastest delay task. The latter implies that in the overall instance there must be task which is delayed $p - 1$ times in total.

and $\tau_{\bar{x},2}$. In the formula, we fix an order of the clauses. For each clause Q there is a vertex v_Q . For each literal in Q there is one *clause task*. We write $\tau_{q,x,i}$ for the task corresponding to the i -th positive occurrence of x and $\tau_{q,\bar{x},i}$ for the task corresponding to the i -th negative occurrence of x . The predefined paths for the task are shown in Table 3 and sketched in Figure 8.1. Note that the underlying graph is a directed tree.

In order to make the construction work as desired we want to block some arcs at certain times. This can be done by introducing *blocking tasks* (see a sketch in Figure 8.2). A blocking task τ_b has to use an arc e at a certain time $t \equiv \bar{t} \pmod p$, otherwise there would be a task in I with a total delay of at least 5. The path of

τ_b is defined such that if τ_b is not delayed it uses the arc e at time $t \equiv \bar{t} \pmod{p}$. After having passed e the path of τ_b moves on to an arc where it crosses the path of $p - 1$ other tasks $T_b := \{\tau_{b,0}, \dots, \tau_{b,p-2}\}$. We call the latter the *delay tasks* of τ_b . Let s_b the start vertex of τ_b , let s'_b be the start vertex of the delay tasks for τ_b . We define a vertex t_b to be the destination vertex of τ_b and all its delay tasks.

There are $p - 2$ delay tasks for τ_b . Hence, in order to ensure that $md(S) \leq p - 2$ it is necessary that for each $k \in \{0, 1, \dots, p - 2\}$ there must be a task $\tau_k \in T_b$ which has an initial delay of k . The paths of τ_b and its delay tasks are designed such that τ_b does not interfere with its delay tasks if τ_b is not delayed at all. However, if τ_b is delayed once then it arrives at v_b at the same time as the task of T_b with zero initial delay. This implies that one of the tasks $\{\tau_b\} \cup T_b$ has a total delay of $p - 1$. In order to clarify matters we do not explicitly define the paths of the blocking packets. Instead, we state which arcs are blocked at what time. To simplify notation we write $[i, i + j)$ if we want to state that an arc is blocked during the time intervals $[i + k \cdot p, i + j + k \cdot p)$ for all $k \in \mathbb{N}$.

Let x be a variable. The arcs $(v_{x,1}, v_r)$ and $(v_{x,2}, v_r)$ are blocked during the time interval $[1, 4)$ (note that $t = 0$ is the first timestep). The arcs $(v_{x,3}, v_{x,4})$, $(v_{x,3}, v_{x,5})$, $(v_{x,6}, v_{x,7})$, and $(v_{x,6}, v_{x,8})$ are blocked during the time interval $[5, 6)$. For each clause Q with three literals the arc (v_Q, v_r) is blocked during the time interval $[2, 3)$. For each clause Q' with two literals the arc $(v_{Q'}, v_r)$ is blocked during the time interval $[1, 3)$. Let M be a packet which was created by a variable or a clause task. Due to the above reasoning we assume that if M wants to use an arc e at a time when e is blocked then M is delayed.

Now we claim that ϕ is satisfiable if and only if there is a periodic schedule S in which each task is delayed at most four times. First we assume that ϕ is satisfiable. In [13] it was shown for a similar construction in the non-periodic case that ϕ is satisfiable if and only if the optimal makespan is 7. Denote by \mathcal{M}_i all packets which were created at time $t = i \cdot p$. We can turn a priority schedule S_{static} of length 7 into a periodic schedule in which all packets in \mathcal{M}_i arrive at their respective destination vertices at time $t = i \cdot p + 7$ the latest. We do this by infinitely repeating the packet movements defined in S_{static} . It remains to argue why packets in \mathcal{M}_i do not interfere with packets in \mathcal{M}_{i-1} . At time $t = i \cdot p$ the packets of \mathcal{M}_{i-1} which have not reached their destination vertices yet are located on vertices $v_{x,3}$ and $v_{x,6}$ for respective variables x . Thus, they do not interfere with the newly created packets in \mathcal{M}_i . Denote by S_{OPT} a schedule which minimizes $md(S)$. Since the paths of all tasks have length 3 we conclude that $md(S_{OPT}) = 7 - 3 = p - 2$. (We ignore the blocking tasks here since we discussed their delay already above.)

For the case that ϕ is not satisfiable it was shown in [13] that the length of each priority schedule S_{static} is at least 8. Since the paths of all packets have length 3 we conclude that in S_{static} there has to be at least one packet which is delayed at least 5 times. This implies the same statement for any periodic schedule. \square

Note that we cannot tighten this construction any further since our template schedule $DTREE(I)$ which can be computed in polynomial time guarantees a limit of $c - 1$ for each task. However, we can show that our result above holds for each period length $p \geq 6$.

Corollary 8.2. *Let $p \geq 6$. Let $I = (G, T, p)$ be an instance of the strict PPRP on a directed tree. It is NP-hard to compute whether $md_{min}(I) \leq c - 2$.*

Proof. For this reduction we again set $c := p$. We can adjust the construction presented in Theorem 8.1 for period lengths p with $p > 6$ as follows: For each variable task and each clause task we block the first arc on its path during the time interval $[0, p - 6)$. Then we shift the times when the arcs are blocked by $p - 6$ timesteps: For each variable x the arcs $(v_{x,1}, v_r)$ and $(v_{x,2}, v_r)$ are blocked during the time interval $[1 + (p - 6), 4 + (p - 6))$, the arcs $(v_{x,3}, v_{x,4})$, $(v_{x,3}, v_{x,5})$, $(v_{x,6}, v_{x,7})$, and $(v_{x,6}, v_{x,8})$ during the time interval $[5 + (p - 6), 6 + (p - 6))$. For each clause Q with three literals the arc (v_Q, v_r) is blocked during the time interval $[2 + (p - 6), 3 + (p - 6))$, for each clause Q' with only two literals the arc $(v_{Q'}, v_r)$ is blocked during the time interval $[1 + (p - 6), 3 + (p - 6))$. The blocking of an arc e works as described in Theorem 8.1: there is a task τ_b which needs to use the e at the given timestep or otherwise it would collide with the packets of the $p - 2$ tasks T_b . Now, each variable packet and each clause packet is delayed for $p - 6$ timesteps in the beginning (due to the blocked arcs). After that, each packet encounters another delay of at most 4 if and only if ϕ is satisfiable. \square

The above results are stated only for the strict PPRP. Note that in the sporadic case no schedule can guarantee less than $c - 1$ delays for each task: The release times of the packets can be chosen such that there is an arc on which c packets arrive at the same time (if they are not delayed already before).

If we allow a more general graph structure than trees, approximating $md_{min}(I)$ becomes much harder. We show that already on chain graphs, the minimum delay $md_{min}(I)$ cannot be approximated with a better ratio than $|T|^{1-\epsilon}$ for all $\epsilon > 0$.

Theorem 8.3. *Let $I = (G, T, p)$ be an instance of the strict PPRP on a chain graph. Assume that the paths of the tasks are given in the input. It is NP-hard to approximate $md_{min}(I)$ with a ratio of $|T|^{1-\epsilon}$ for all $\epsilon > 0$.*

Proof. We reduce from COLORING. Given a graph G , let $\chi(G)$ denote the chromatic number of G , i.e., the minimum number of colors needed to color G such that each pair of adjacent vertices is colored with different colors. It is NP-hard to approximate $\chi(G)$ with a ratio of $|V|^{1-\epsilon}$ for all $\epsilon > 0$ [15]. Let $G = (V, E)$ be a graph. We define $p := n = |V|$ (knowing that at most n colors are necessary to color G).

For each vertex $v_i \in V$ we define a task τ_i . We define the paths of the tasks such that two paths P_i and P_j corresponding to two tasks τ_i and τ_j share an arc if and only if $\{v_i, v_j\} \in E$.

We use $n^2 + 1$ vertices for this construction. Since $|E| \leq |V|^2$ this is sufficient. We call the construction the *basic setup*, see Figure 8.3 shows a sketch. All tasks move from the vertex s on the very left to the vertex u_1 on the very right. For our instance I we repeat the basic setup n times. Denote by u_i the last vertex of the i th copy of the basic setup (i.e., $t_i = u_n$).

We claim that $md_{min}(I) = \chi(G) - 1$. First we show that $md_{min}(I) \leq \chi(G) - 1$. Assume we know a coloring $c : V \rightarrow \{0, 2, \dots, \chi(G) - 1\}$ for G . We obtain a valid periodic schedule as follows: Each task τ_i is assigned an initial waiting time $w_i := c(v_i)$. Then it proceeds to its destination without being delayed any further. Since c is a valid coloring and the paths of two tasks T_i and T_j share an arc if and only if $\{v_i, v_j\} \in E$ this gives a valid schedule S with $md(S) = \chi(G) - 1$.

Now we want to show that $md_{min}(I) \geq \chi(G) - 1$. Assume we know a schedule S with $md(S) = md_{min}(I)$. W.l.o.g. we assume that S never delays a packet when

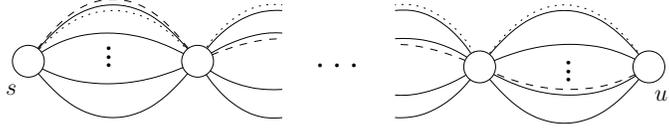


FIGURE 8.3. The basic chain. The dotted and dashed lines denote the path of two tasks τ_i and τ_j such that v_i and v_j are adjacent in G .

it is not necessary, i.e., packets are delayed only if more than one packet needs to use the same arc at a time. We define a coloring as follows: For a vertex τ_i let $d(\tau_i)$ denote the number of delays which τ_i encounters in S . We define a coloring $c : V \rightarrow \{0, 1, \dots, md(S) - 1\}$ by $c(v_i) := d(T_i)$. We want to show that this is indeed a valid vertex coloring for G .

For $1 \leq i \leq n$ let V_i denote all vertices corresponding to tasks which reach u_i after exactly $i \cdot n^2 + (i - 1)$ timesteps. We claim that each set V_i is an independent set. Since no task is delayed if not necessary we have that no task $\tau_i \in V_i$ is delayed after having reached the vertex u_i . This shows that c is a valid coloring for G with $md(S)$ colors.

We prove the claim by induction. Let $i = 1$. If a task τ_j arrives at u_1 after n^2 timesteps, it has never been delayed. Thus, all vertices corresponding to these tasks form an independent set. Now assume that the claim is true for all i' with $i' \leq i$. Consider the set V_{i+1} . By the induction hypothesis, each task τ_j with $v_j \in V_{i+1}$ arrives at u_i after at least $i \cdot n^2 + i$ timesteps: otherwise it would be included in a set $V_{i'}$ with $i' < i + 1$ and, thus, it would arrive at u_{i+1} after at most $(i + 1) \cdot n^2 + i' - 1 < (i + 1) \cdot n^2 + i$ steps. Since τ_j arrives at u_{i+1} after $(i + 1) \cdot n^2 + i$ steps we conclude that τ_j reached u_i after exactly $i \cdot n^2 + i$ steps. This implies that the vertices in V_{i+1} form an independent set.

It remains to show that the non-approximability of COLORING carries over to the computation of $md_{min}(I)$. Assume that there is an approximation algorithm which computes a value $\tilde{md}(I)$ such that $\tilde{md}(I) \leq \alpha(|T|) \cdot md_{min}(I)$. This would yield a coloring with $\tilde{md}(I) + 1$ colors. We calculate that

$$\begin{aligned} \tilde{md}(I) + 1 &\leq \alpha(|T|) \cdot md_{min}(I) + 1 \\ &\leq \alpha(|T|) (md_{min}(I) + 1) \\ &= \alpha(n) \chi(G) \end{aligned}$$

This implies that we have an approximation algorithm for COLORING with an approximation factor of $\alpha(n)$. Thus, it is NP -hard to approximate $md_{min}(I)$ on chain graphs with a ratio of $|T|^{1-\epsilon}$ for all $\epsilon > 0$. \square

Now we show that it is NP -hard to determine whether for an instance of the periodic packet routing problem on a bidirected tree there is a direct template schedule.

Theorem 8.4. *Let I be an instance of the strict PPRP or the sporadic PPRP on a bidirected tree. It is NP -hard to determine whether there is a direct template schedule for I .*

Proof. We give a reduction from DIRECTED-PATH-COLORING on bidirected binary trees: Given a binary tree G , a set of directed paths \mathcal{P} on G . The question

is whether it is possible to color the paths in \mathcal{P} with three colors such that each two paths which use an arc in the same direction have different colors. This problem is *NP*-hard [5].

The following construction is similar to the construction used for Proposition 3.7. Given an instance (G, \mathcal{P}) of the DIRECTED-PATH-COLORING such that $G = (V, E)$ is a bidirected tree we construct an instance (G', T') of the periodic packet routing problem as follows: We obtain $G' = (V', E')$ by taking G and replacing each pair of arcs between two vertices u and v by a path with anti-parallel arcs of length six between u and v . We call the vertices in V' which already existed in V the *old* vertices, all other vertices are called the *new* vertices. We note that G' is also a bidirected tree. For each vertex $v \in V$ there is a corresponding (old) vertex $v' \in V'$. For each path $P_i \in \mathcal{P}$ from $s_i \in V$ to $t_i \in V$ we introduce a task $\tau_i = (s'_i, t'_i) \in T'$. Since G' is a tree, the path for τ_i is implicitly given. We define $p := 3$.

Now we prove that the paths \mathcal{P} can be colored with three colors if and only if there is a direct template schedule for (G', T', p) . First assume that there is a valid coloring $f : \mathcal{P} \rightarrow \{0, 1, 2\}$ for the paths \mathcal{P} . We define a template schedule by setting $task(e_i, f(P_i)) := \tau_i$ for each task τ_i with e_i being the first arc on P_i . We define the remaining values for $task$ such that the resulting schedule is direct. Also, we define $\bar{p} := p = 3$. For an old vertex $u \in P_i$ let $leave(\tau_i, u, j)$ be the time when the packet $M_{i,j}$ leaves u . We observe that $leave(\tau_i, u, j) \equiv f(P_i) \pmod{3}$ for all old vertices $u \in P_i$ and all positive integer j . Two packets $M_{i,j}$ and $M_{i',j'}$ can collide only if there is an old vertex u and an arc $e = (u, v)$ which P_i and $P_{i'}$ have in common and if $leave(\tau_i, u, j) = leave(\tau_{i'}, u, j')$. But the latter implies that $f(P_i) = leave(\tau_i, u, j) \pmod{3} = leave(\tau_{i'}, u, j') \pmod{3} = f(P_{i'})$. But this cannot happen since f is a valid path coloring.

Now assume that we are given a valid direct template schedule S for (G', T', p) . Note that $\bar{p} \leq p = 3$. For each task τ_i we define e_i to be the first arc on P_i . We define our path coloring $f : \mathcal{P} \rightarrow \{0, 1, 2\}$ such that $task(e_i, f(P_i)) = \tau_i$. We observe that $task(\bar{e}, f(P_i)) = \tau_i$ for each arc $\bar{e} \in P_i$ which points out of an old vertex in P_i . This holds since each (bidirected) edge in G was replaced by a path of anti-parallel arcs with length six in G' and $\bar{p} \in \{1, 2, 3\}$. Thus, if f was not a valid path coloring then there would be an arc \bar{e} pointing out of an old vertex and two paths $P_i \neq P_j$ with $f(P_i) = f(P_j)$ which both use \bar{e} . However, this would imply that $\tau_i = task(\bar{e}, f(P_i)) = task(\bar{e}, f(P_j)) = \tau_j$ which is a contradiction since $P_i \neq P_j \Rightarrow \tau_i \neq \tau_j$. \square

The reduction above can also be adjusted to undirected trees.

Theorem 8.5. *Let I be an instance of the strict PPRP or the sporadic PPRP on an undirected tree. It is *NP*-hard to determine whether there is a direct template schedule for I .*

Proof. Can be shown with a similar reduction as Theorem 8.4. Here, if a path is used in only one direction, we introduce an artificial task which uses the path in the opposite direction. This ensures that each path has the same bandwidth in each direction. \square

9. GENERAL GRAPHS

In this section we study global-priority schedules on general graphs. We show that already on a very simple graph class, namely directed chain graphs, there are instances in which no global-priority schedule can guarantee a non-trivial limit for each task. We assume that for each task a path is given as part of the input.

Theorem 9.1. *For any $D > 0$ there is an instance $I = (G, T, p)$ of the strict PPRP with given paths on a chain graph G such that*

- $D_i \geq D$ for each task $\tau_i \in T$,
- all tasks have the same start and the same destination vertex, and
- for every global-priority schedule S there is at least one task $\tau_i \in T$ which has a limit in $\Omega(p \cdot D_i)$.

Proof. Fix a period length $p \in \mathbb{N}$. We construct an instance I of the periodic packet routing problem with fixed paths as follows: There is a single source vertex s , a single destination vertex t , and $p + 1$ task $\tau_i = (s, t)$. The chain graph G consists of L connected copies of a gadget N (see Figure 9.1 for a sketch). We can enlarge G arbitrarily by choosing L appropriately big. This will ensure the requirement $D_i \geq D$ for each task τ_i . Between two nodes we have exactly two arcs. For each such pair of arcs $e_1 = e_2 = (u, v)$ there are p tasks whose respective path uses e_1 and one task τ_i which uses e_2 . We say that the task τ_i is *alone* between u and v . \square

Proof. The gadget N is constructed as follows: In the first part there are $p + 2$ nodes such that each task τ_i is alone between exactly two adjacent vertices. Then, the first part is copied mirroredly. To create G we place L connected copies of N (see Figure 9.2). The vertex t is the very last vertex of G .

Let S be an arbitrary priority schedule. We claim that there is at least one task whose packets need at least $(L + 1) \cdot ((p - 1) \cdot p + 2p + 2) \in \Omega(p \cdot D_i)$ timesteps to reach t . W.l.o.g. assume that $\tau_1 \prec \tau_2 \prec \dots \prec \tau_{p+1}$ in S . In the first copy of N there is an arc $e = (\bar{u}, \bar{v})$ which is used by the tasks τ_1, \dots, τ_p (the task τ_{p+1} is alone between \bar{u} and \bar{v}). We call these tasks the *good* tasks and their packets the *good* packets.

New packets are created every p timesteps. Thus, there is a constant t such that at each timestep $t' \geq t$ exactly one packet created by a good task arrives at \bar{v} . (If this was not the case then the arc e would not transport enough packets in the long run.)

Now let k be an integer such that $k \cdot p \geq t$ and consider the packet $M_{p+1, k}$. Denote by v_1 the first vertex of the second copy of N . We claim that in each of copy of N after the first copy the packet $M_{p+1, k}$ needs at least $(p - 1) \cdot p + 2p + 2$ timesteps to pass the gadget. Consider the second copy of N (for the other copies the proof works similarly) and denote by v_1, \dots, v_{2p+2} its vertices. Denote by ℓ_i the time that $M_{p+1, k}$ needs to wait at the vertex v_i . Since all vertices v_i lie between \bar{v} and t we can guarantee that from timestep t on a continuous stream of good packets passing the second gadget.

We consider pairs of vertices $v_i, v_{i'}$ with $i' = 2p + 4 - i$. Let τ_j and $\tau_{j'}$ be the tasks which are alone between the vertices (v_{i-1}, v_i) and (v_i, v_{i+1}) respectively. We assume that i is chosen such that $j \neq p + 1 \neq j'$. We observe that τ_j is alone between the vertices (v_i, v_{i+1}) and $\tau_{j'}$ is alone between the vertices (v_{i-1}, v_i) .

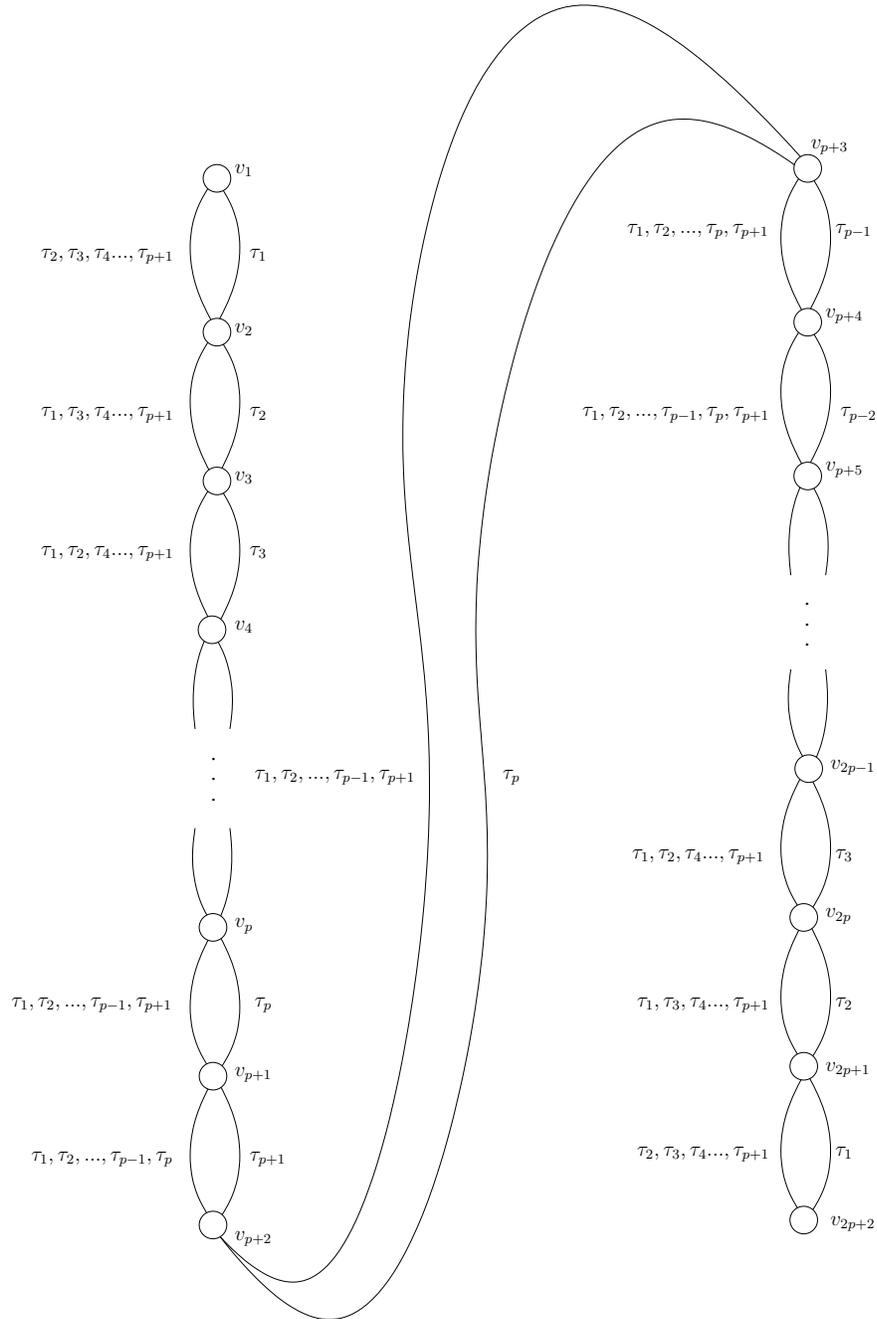
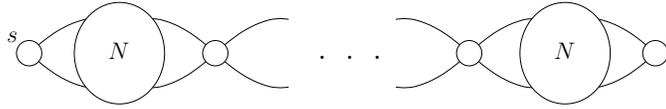


FIGURE 9.1. The gadget N . The tasks which are alone in the respective arc are stated next to the arc.

FIGURE 9.2. We place L connected copies of the gadget N .

Since all vertices v_i lie between \bar{v} and t we have the following: For all pairs of adjacent vertices for which τ_{p+1} is not alone, the packet $M_{p+1,k}$ uses the gap that is left by the packets whose respective tasks are alone there. E.g., for (v_{i-1}, v_i) the packet $M_{p+1,k}$ uses the gap left by the task τ_j . Since in each such arc there is only one gap this implies that if $j < j'$ then $\ell_i = j' - j$, whereas if $j > j'$ then $\ell_i = p - j + j'$. From this we conclude that $\ell_i + \ell_{i'} = p$.

From the construction it follows that there are only two vertices in N on which $M_{p+1,k}$ does not wait. Thus, there are $p - 1$ pairs of vertices $v_i, v_{i'}$ with $\ell_i + \ell_{i'} = p$. So in total, $M_{p+1,k}$ is delayed $p \cdot (p - 1)$ times. Hence, $M_{p+1,k}$ needs $(p - 1) \cdot p + 2p + 2$ timesteps to pass N . Since there are L copies of N in G the packet $M_{p+1,k}$ needs at least $(L - 1) \cdot ((p - 1) \cdot p + 2p + 2)$ timesteps to reach its destination. Recall that $D_{p+1} = L \cdot (2p + 2)$. We conclude that $(L - 1) \cdot ((p - 1) \cdot p + 2p + 2) \in \Omega(p \cdot D_{p+1})$. \square

ACKNOWLEDGEMENTS

We would like to thank Martin Niemeier for fruitful discussions on the topic.

REFERENCES

- [1] M. Andrews, A. Fernández, M. Harchol-Balter, F. Leighton, and L. Zhang. General dynamic routing with per-packet delay guarantees of $O(\text{distance} + 1/\text{session rate})$. *SIAM Journal of Computing*, 30:1594–1623, 2000.
- [2] C. Busch, M. Magdon-Ismail, M. Mavronicolas, and P. Spirakis. Direct routing: Algorithms and complexity. *Algorithmica*, 45:45–68, 2006.
- [3] G. C. Buttazzo. *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series)*. Springer-Verlag TELOS, Santa Clara, CA, USA, 2004.
- [4] M. di Ianni. Efficient delay routing. *Theoretical Computer Science*, 196:131–151, 1998.
- [5] T. Erlebach and K. Jansen. The complexity of path coloring and call scheduling. *Theoretical Computer Science*, 255:33–50, 2001.
- [6] K. Jansen. Approximation results for wavelength routing in directed trees. In *Proceedings of the 2nd Workshop on Optics and Computer Science*, 1997.
- [7] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-scheduling in $O(\text{congestion} + \text{dilation})$ steps. *Combinatorica*, 14:167–186, 1994.
- [8] F. T. Leighton, B. M. Maggs, and A. W. Richa. Fast algorithms for finding $O(\text{congestion} + \text{dilation})$ packet routing schedules. *Combinatorica*, 19:375–401, 1999.
- [9] F. T. Leighton, F. Makedon, and I. G. Tollis. A $2n - 2$ step algorithm for routing in an $n \times n$ array with constant size queues. In *Proceedings of the 1st Annual Symposium on Parallel Algorithms and Architectures*, pages 328–335, 1989.
- [10] J. Y.-T. Leung. *Handbook of Scheduling: Algorithms, Models and Performance Analysis*. 2004.
- [11] Y. Mansour and B. Patt-Shamir. Greedy packet scheduling on shortest paths. *Journal of Algorithms*, 14, 1993.
- [12] B. Peis, M. Skutella, and A. Wiese. Packet routing: Complexity and algorithms. In *Proceedings of the 7th Workshop on Approximation and Online Algorithms*, 2009.
- [13] B. Peis, M. Skutella, and A. Wiese. Packet routing: Complexity and algorithms. Technical Report 003-2009, Technische Universität Berlin, February 2009.

- [14] B. Peis, M. Skutella, and A. Wiese. Packet routing on the grid. In *Proceedings of the 9th Latin American Theoretical Informatics Symposium*, 2010.
- [15] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.