Approximation Algorithms for Time-Constrained Scheduling on Line Networks

Harald Räcke* Adi Rosén†

Abstract

We consider the problem of time-constrained scheduling of packets in a communication network. Each packet has, in addition to its source and its destination, a release time and a deadline. The goal of an algorithm is to maximize the number of packets that arrive to their destinations by their respective deadlines, given the network constraints.

We consider the line network, and a setting where each node has a buffer of size B packets (where B can be finite or infinite), and each edge has capacity $C \geq 1$. To the best of our knowledge this is the first work to study time-constrained scheduling in a setting when buffers can be of limited size. We give approximation algorithms that achieve expected approximation ratio of $O(\max\{\log^* n - \log^* B, 1\} + \max\{\log^* \Sigma - \log^* C, 1\})$, where n is the length of the line, and Σ is the maximum slack a message can have (the slack is the number of time steps a message can be idle and still arrive within its deadline).

A special case of our setting is the setting of buffers of unlimited capacity and edge capacities 1, which has been previously studied by Adler et al. [2]. For this case our results considerably improve upon previous results: We obtain an approximation ratio of $O(\min\{\log^* n, \log^* \Sigma, \log^* M\})$ (where M is the number of messages in the instance), which is a significant improvement upon the results of Adler et al. who obtained a guarantee of $O(\min\{\log n, \log \Sigma, \log M\})$.

1 Introduction

Today, communication networks constitute the infrastructure for numerous applications. These include various applications where the delivery of the data over the network must be done under tight quality-of-service constraints. For example, real-time video or audio applications require that the packets arrive to their destination within some given delay, or else they are of no use. A similar phenomenon is seen also in interconnection networks where emerging real-time applications, such as industrial process control and avionics [14], rely on communication with limited delay. This has motivated research into the efficient delivery of time-constrained messages (or packets) over communication networks.

*Department of Computer Science and Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, Coventry, UK, email: H.Raecke@warwick.ac.uk

†CNRS and University of Paris 11 Laboratoire de Recherche en Informatique (LRI) Bât. 490, Université Paris Sud 91405 Orsay, France emai: adiro@lri.fr

Informally (see Section 2 for a formal definition) the problem we consider constitutes of a communication network, and a set of messages, each message having a source node, s, a target node, t, a release time, r, and a deadline, d. The goal is to maximize the number of packets that arrive to their respective destinations by their respective deadlines, under the constraints that the network imposes (such as limited link capacity, and/or limited buffer capacity, etc.). The first analytical results for this problem with arbitrary release times and deadlines were given by Adler et al. [2] who studied the case where the buffers have unlimited capacity and the link capacities are 1 (thus, the capacity of the links is the only bottleneck). They considered the linear network (and the cycle network), showed that the problem is NP-hard on the linear network, and gave a logarithmic approximation algorithm.

In fact, Adler et al. [2] did not give a (direct) approximation algorithm for the problem, but rather gave a constant-approximation algorithm for a more restricted model (the bufferless case - where a packet may only wait at a source node but not once it made its first step towards its destination). They then showed that buffers can only improve the optimal solution by a logarithmic factor, even if they are of unlimited size. They further showed that indeed such a gap can occur.

More precisely, Adler et al. [2] identified three important parameters of the problem. The length of the line, n, the number of messages, M, and the maximum slack a message can have, Σ (the slack of a message is the number of time steps it can be idle in the network and still arrive in time to its destination). They showed that there exists an approximation algorithm with approximation ratio $O(\min\{\log n, \log \Sigma, \log M\})$. Similar results were later obtained for the tree topology and the grid topology [1], by similar means of giving a constant approximation for the bufferless case, and showing that buffers can improve the optimal solution by only a logarithmic factor.

The concentration on linear networks (or on networks with simple structure like trees or grids) is motivated not only by application reasons (routes are often defined by means of long linear stretches), but also by the fact that on general networks the problem is hard to approximate to within a factor of $M^{1-\epsilon}$ unless $NP = ZPP$, where M is the number of packets [1].

In this paper we generalize the setting studied in previous works [1, 2, 11], and consider line networks where the buffers at the nodes have finite, or infinite, capacity B, and the links have some arbitrary capacity $C \geq 1$. To the best of our knowledge this is the first analytical work that considers time-constrained scheduling of packets in a network with finite buffers. We give an approximation algorithm that achieves an expected approximation ratio of $O(\max\{\log^* n - \log^* B, 1\} + \max\{\log^* \Sigma - \log^* C, 1\})$. Note, that we assume that the bounds C and B are *uniform*, i.e., we assume that every node has the same buffer-size B and every link has the same capacity C. While our results obviously still hold if buffer-sizes and link-capacities vary by constant factors (as this only can improve the optimum throughput by a constant factor), our result does not hold for the case that one is merely given lower bounds of B and C, on buffer-sizes and link-capacities, respectively.

A slight modification of our algorithm also gives an algorithm with an approximation ratio of $O(\min\{\log^* n, \log^* \Sigma\}) = O(\min\{\log^* n, \log^* \Sigma, \log^* M\})$ (we note that one can

assume without loss of generality that $\Sigma \leq M$. See Lemma 17). This is of interest, among other things, in the context of the setting of buffers of unlimited capacity and link capacities 1, studied in [1, 2], for which our results give a significant improvement over the results of Adler et al. [2].

Our results are obtained by a direct approximation algorithm for the buffered setting, as opposed to previous works that obtained their results using an algorithm for the bufferless case. The results of Adler et al. [2] showing that a logarithmic gap can occur between the bufferless case and the buffered case (when buffers are unlimited in size), indicate that this approach is necessary. Our algorithm is based, among other things, on a formulation of the problem as an edge-disjoint path problem on *directed* two-dimensional grids of a certain structure. The proof of correctness of this algorithm relies on the special structure of the grid. While previous results for the edge-disjoint paths problem on grids do exist [8, 9], they apply to *undirected* grids only. Indeed the algorithms of Kleinberg and Tardos [9] use a technique for routing across (sub)grids, which is inherently undirected, and thus cannot be applied in our case.

Our results also give approximation algorithms for the problem of packet scheduling with bounded buffers on line networks [3, 4, 5] (i.e., when there are limited buffers, but packets do not have deadlines). This problem can be formulated as a special case of our setting where $\Sigma = M$.[1] Previous work has mainly considered online (and sometimes distributed) algorithms for this problem, achieving polylogarithmic (in n) competitive ratios (for the centralized setting). Our results give the first approximation for this problem achieving approximation ratio of $O(\log^* n)$.

1.1 Our Results

Consider instances of a line network of n nodes, edges with capacities $C \geq 1$, buffers of size $B \geq 1$, and a set $\mathcal{M}$ of messages, $|\mathcal{M}| = M$. The main results of our paper are cast in the following theorems which are proved in Section 3.

Theorem 1. *There is an algorithm with running time polynomial in n and M with an expected approximation ratio of $O(\max\{\log^* n - \log^* B, 1\} + \max\{\log^* \Sigma - \log^* C, 1\})$.*

A slight modification of this algorithm gives the following.

Theorem 2. *There is an algorithm with running time polynomial in n and M, that achieves expected approximation ratio of $O(\min\{\log^* n, \log^* \Sigma\})$.*

The case where packets do not have deadlines (and edge capacities are 1) was considered (in the online setting) in [3, 5, 4]. The goal is to maximize the number of packets that reach their destination rather than being dropped due to limited buffer space. This scenario can be modeled in our framework by setting the slack of all messages to be M. For this case we have the following theorem.

Theorem 3. *If packets do not have deadlines, then there is an algorithm with running time polynomial in n and M, that achieves expected approximation ratio of $O(\log^* n)$.*

[1] See Lemma 17.

1.2 Related Work

Adler et al. [2] were the first to model and analytically study the problem of time-constrained scheduling in communication networks, when release times and deadlines are arbitrary (e.g., not all packets are released simultaneously). They considered two models, the *bufferless* case, where packets cannot be buffered en-route, and the *buffered* case, where packets can be buffered en-route, but buffers are of infinite capacity. They considered the linear (and cycle) network and gave a constant-approximation algorithm for the bufferless case. They further showed that *unlimited* buffers can improve the optimal solution over the bufferless setting by only a factor of $\Lambda = O(\min\{\log n, \log \Sigma, \log M\})$, where n is the length of the line, M is the number of messages, and Σ is the maximum slack of a message in the instance. They thus also showed the existence of a Λ-approximation algorithm for the buffered setting. They also showed that the approximation ratio is constant in some special cases (e.g., when all packets are released simultaneously), and that the problem is NP-hard (both for the buffered and bufferless cases) on the linear network.

These results have been later extended by Adler et al. [1] in two directions. First, the results have been extended to the case of weighted packets, where the goal is to maximize the total weight of the packets that arrive to their destinations in time (rather than maximizing their number). Second, the results have been extended to the topology of trees and the topology of grids (when the routing paths of packets are fixed according to order-dimension routing[2]). It is further shown in [1] that on general networks the problem cannot be approximated to within a factor of $M^{1-\epsilon}$ unless $NP = ZPP$. The online setting of the problem has been considered in [1, 2, 11].

The problem of packet scheduling on networks with finite buffers has been studied in several works [3, 4, 5, 6]. In this setting there are no deadlines for the packets, and the goal is to maximize the number of packets that arrive to their destination rather than being dropped due to buffer space limitations. This problem has been mainly addressed in the online (and sometimes distributed) settings. Packets are injected by an adversary in an online manner, and in each time-step an online algorithm has to decide which packets to forward, which packets to store, and which packets to drop. Online algorithms for the line, with polylogarithmic (in n) competitive ratios, were given in [4, 5]. Online distributed protocols for the line with competitive ratio of $O(n^{2/3})$ were given in [3].

2 Model and Preliminaries

We consider directed linear networks. The linear network consists of n nodes $\{1, \ldots, n\}$, and $n-1$ directed edges, $(i, i+1)$, for $1 \leq i \leq n-1$. The system is synchronous, and in each time step each edge can transmit C messages, for some finite $C \geq 1$.

We are given a set $\mathcal{M}$, $|\mathcal{M}| = M$ of messages. Each message $m = (s_m, t_m, r_m, d_m) \in \mathcal{M}$ consists of a *source node* $s_m \in \{1, \ldots, n\}$ from the linear array, a *target node* $t_m \in$

[2]In order-dimension routing a packet from node $a = (a_1, a_2, \ldots, a_d)$ to node $b = (b_1, b_2, \ldots, b_d)$ is first sent to node $(b_1, a_2, \ldots, a_d)$ then to node $(b_1, b_2, a_3 \ldots, a_d)$ etc.

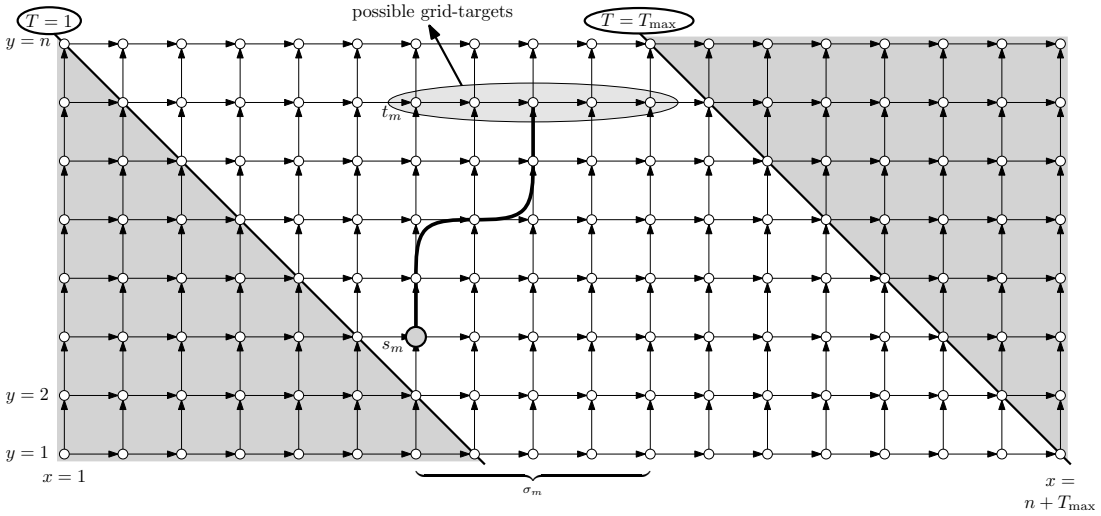

Figure 1: The routing problem can be illustrated by a grid of dimension $(n + T_{\max}) \times n$. The horizontal distance between the end-points of a message corresponds to the slack σ_m. Note that the shaded regions do not contain any source or target nodes.

$\{s_m, \ldots, n\}$ (i.e., a node reachable from s_m), a release time r_m, $0 \leq r_m \leq T_{\max}$ and a deadline d_m, $r_m \leq d_m \leq T_{\max}$. Here, $T_{\max}$ denotes the maximum deadline of any message from $\mathcal{M}$. For a message m, we define the *slack* $\sigma_m := (d_m - r_m) - (t_m - s_m)$ to be the number of time steps a message can reside in a buffer, and still arrive at its destination by its deadline. Without loss of generality we can assume that the slack of any message from $\mathcal{M}$ is nonnegative and at most M (See Lemma 17 in the appendix for a justification).

Every node has a buffer which is used to store packets waiting to be sent from this node. This buffer can store at most B packets, for some finite $B \geq 1$. A message with release time r_m appears in node s_m at time r_m. For a node v and time τ we denote by $\mathcal{I}_v(\tau)$ the set of messages with source node v and release time τ. For a node v and time τ we denote by $\mathcal{M}_v^{\text{in}}(\tau - 1)$ the set of messages that are sent to v at time $\tau - 1$, and do not have v as their destination node. Note that this set depends on the specific routing algorithm used and that it may be empty. When a message with deadline d_m and target node t_m is sent along edge $(t_m - 1, t_m)$ in a time step $\tau \leq d_m$, the message is being delivered and is eliminated from the system (in particular the message does not occupy any more buffer space). We denote by $\mathcal{T}_v(\tau)$ the set of messages in the buffer of node v at time τ. Initially (i.e., at time 0) this set is empty. In any time step τ each node v has to select at most $B + C$ messages from $\mathcal{I}_v(\tau) \cup \mathcal{T}_v(\tau) \cup \mathcal{M}_v^{\text{in}}(\tau - 1)$. At most B of these messages are stored in v's buffer, and constitute the set $\mathcal{T}_v(\tau + 1)$, and at most C of these messages are sent along v's outgoing edge.

In order to visualize and specify valid schedules for the packets we build a two dimensional directed grid G as seen in Figure 1. A node (x, y), $1 \leq x \leq T_{\max} + n$; $1 \leq y \leq n$ corresponds to a node y of the linear array at time step $\tau = x+y-n$. Horizontal grid edges

correspond to the buffers and have capacity B. A vertical edge corresponds to an edge of the linear array at a certain time step τ, and has capacity C: edge $((x,y),(x,y+1))$ corresponds to the usage of edge $(y,y+1)$ at time step $\tau = x+y-n$. We note that up to notations (and the fact that edges that correspond to buffers have finite capacity) this formulation is equivalent to the one used in [2].

For a message $m = (s_m, t_m, r_m, d_m)$ we refer to the node $(r_m - s_m + n, s_m)$ as the *grid-source* of the message. Further, we call the nodes $(r_m - s_m + n, t_m), \ldots, (d_m - t_m + n, t_m)$ the *possible grid-targets* of m. A schedule for a message is given by a path from its grid-source to one of its possible grid-targets. This path specifies when the message traverses an edge in the network and when it resides in a buffer. Note that the possibility for a message to be sent to one of several possible grid-targets reflects the fact that we allow a message to be delivered ahead of its deadline. We refer to the problem thus defined as the *packet scheduling problem on the grid*.

A valid schedule for the messages can be converted into a set of paths in the grid between source-target pairs without violating edge-capacities. This means that a vertical edge is used by at most C paths, and horizontal edges are used by at most B paths. Conversely, any valid set of paths for the messages (i.e., a set of paths that does not violate edge capacities, and for each message connects its grid-source to at most one of its grid-targets), can be converted into a valid schedule.

By viewing our problem as a path selection problem on the grid we directly obtain connections to the unsplittable flow problem. In fact, if we connect the possible grid-targets of a message m to a super-target T_m and require that a message chooses a path from its source s_m to T_m we obtain a one-to-one correspondence between our packet scheduling problem, and the resulting unsplittable flow problem.

So one result of the paper is that the unsplittable flow problem can be well approximated for these specific instances.

3 General Instances

In this section we develop our main result. We are given an instance of the packet scheduling problem, and we consider its equivalent formulation on the grid graph as defined in the previous section. A message can be successfully scheduled by assigning it a path from its grid-source to one of its grid-targets. In Section 4 we will design a constant factor approximation for *restricted instances*. These restricted instances are instances for which for every message (1) the vertical distance in the grid between its grid-source and any of its grid-destinations (i.e., the distance in the array) falls within a certain range, and (2) the horizontal distance in the grid between its grid-source and any of its grid-destinations (i.e., the number of time steps it resides in a buffer) falls within a certain range.

In order to use this result as a sub-routine for general instances we show in this section how to decompose a given general instance into a collection of restricted instances with the following property. For a message m in the general instance we generate several messages $m'_1, \ldots, m'_s$, each of which belongs to a different restricted instance. The mes-

sages m'_i will have the same grid-source as m but will only have a subset of m's possible grid-targets as their own grid-targets.

Let $\mathcal{T}_m$ denote the set of possible grid-targets of a message m. We make sure that we do not *lose* any grid-targets by requiring that for every message m in the general instance $\bigcup_i \mathcal{T}_{m'_i} = \mathcal{T}_m$. Thus, the optimal number of messages that is routable in all restricted instances combined will be at least as large as the number of messages routable in the original general instance. By routing all messages of the "best" restricted instance (i.e. the instance for which our constant factor approximation algorithm of Section 4 routes the largest number of messages) we obtain an overall approximation guarantee of $O(s)$, where s denotes the number of instances that we generate.

Definition 4. *For given values D and Σ, (D, Σ)-*restricted instance *of the packet scheduling problem on the grid is an instance in which every message m with grid-source (x_s, y_s) fulfills the following. Let (x_t, y_t) denote any of m's possible grid-targets. Then*

$$\frac{49}{B} \ln\left(\frac{8e}{B} \cdot D \cdot \min\{DB, \Sigma C\}\right) \leq y_t - y_s \leq D \ . \tag{1}$$

Further,

$$\frac{49}{C} \ln\left(\frac{8e}{C} \cdot \Sigma \cdot \min\{DB, \Sigma C\}\right) \leq x_t - x_s \leq \Sigma \ . \tag{2}$$

The rough intuition for the above numbers (for the case $C = B = 1$) is as follows. Our algorithm for restricted instances[3] will be based on randomized rounding. Given an instance where the horizontal and vertical distances are bounded by Σ and D, respectively, we show in Lemma 10 that we can decompose this instance into instances on sub-grids of dimension $O(\Sigma) \times O(D)$. These instances are then approximated via randomized rounding. However, for randomized rounding to work edges usually need a large capacity of $\Omega(\log(\#\text{variables}))$, where in standard applications of randomized rounding there is one variable for every edge, which leads to a required edge capacity of $\Omega(\log(D \cdot \Sigma))$ for our case.

In Section 4 we show that we can perform a randomized rounding approach with edge capacities $O(\log \Sigma)$ and $O(\log D)$ for horizontal and vertical edges, respectively. Then we will show that we can "*simulate*" edges of large capacities for source-target pairs that are far apart. This will be the key step in making our randomized rounding approach work, and leads to the required lower bounds of $\Omega(\log \Sigma)$ and $\Omega(\log D)$, respectively.

In the following we first show how to obtain our results assuming a constant factor approximation for restricted instances. Recall that without loss of generality we can assume that the maximum slack is at most M. Let $\Sigma \leq M$ denote the maximum slack, and let n denote the length of the array. We partition the range of Σ (possible slack values, i.e., horizontal distance in the grid) and D (vertical distance) into intervals in a

[3]Note that a grid-instance derived from a packet scheduling problem on the line is usually not a restricted instance for any value of D and Σ as we require a lower bound on the horizontal distance between a grid-source and a possible grid-target.

recursive manner. Let $\Sigma_0 = \Sigma$ and define $\Sigma_i := \frac{c}{C}\ln(\Sigma_{i-1})$ for $c = 147$. Similarly, define $N_0 = n$ and define $N_i := \frac{c}{B}\ln(N_{i-1})$. Define $I_s^\Sigma := [\Sigma_s, \Sigma_{s-1}]$ and $I_r^n := [N_r, N_{r-1}]$.

We will see in Lemma 6 that if we restrict horizontal distances between a source and its possible targets to be in interval I_s^Σ and vertical distances to be in interval I_r^n, then we have an (N_{r-1}, Σ_{s-1})-restricted instance provided that $N_{r-1} \geq 8e$ and $\Sigma_{s-1} \geq 8e$. We now give the following proposition regarding the recursive partitioning. Its proof appears in Section C in the appendix.

Proposition 5. *There exist values δ_n and δ_Σ with $\delta_n \leq \max\{\log^* n - \log^* B, 0\} + O(1)$ and $\delta_\Sigma \leq \max\{\log^* \Sigma - \log^* C, 0\} + O(1)$, such that $N_{\delta_n} = O(1)$ and $\Sigma_{\delta_\Sigma} = O(1)$. In addition we have $\forall i < \delta_n : N_i \geq 8e$, and $\forall i < \delta_\Sigma : \Sigma_i \geq 8e$.*

In Section 4 we give a constant factor approximation algorithm for restricted instances. The following lemma shown that the instances that we get by restricting the horizontal and vertical distances in the array according to the recursion defined above, indeed give us restricted instances in the sense of Definition 4.

Lemma 6. *Suppose we are given an instance in which the horizontal distance between a grid-source and any of its grid-targets is in I_s^Σ, and the vertical distance between them is in I_r^n, for some values $s \leq \delta_\Sigma$ and $r \leq \delta_n$. Then the algorithm of Section 4 can be used to obtain a constant factor approximation for that instance.*

Proof. We only need to show that the intervals I_s^Σ and I_r^n fulfill the requirements on the range of horizontal and vertical distances from Definition 4. We only show this for an interval of the form I_r^n as the other case is analogous. Let $D := N_{r-1}$ denote the upper bound of the interval. We have to show that for the lower bound N_r we have

$$N_r \geq \frac{49}{B}\ln\left(\frac{8e}{B} \cdot N_{r-1} \cdot \min\{N_{r-1}B, \Sigma C\}\right) .$$

We can obtain the following bound for the RHS of the above equation.

$$\text{RHS} \leq \frac{49}{B}\ln\left(8e \cdot N_{r-1}^2\right) \leq \frac{147}{B}\ln(N_{r-1}) = N_r ,$$

where the second step follows since $N_{r-1} \geq 8e$ due to Proposition 5. □

Every combination of intervals in the range given in the above lemma generates a sub-instance for which we can apply the approximation algorithm from Section 4. However, some combinations of intervals can be solved easier.

In the following we derive lemmas for some of these easier instances. For this we require one ingredient from the result of Section 4, which is the following lemma:

Lemma 10. *Given a (D, Σ)-restricted instance there is a randomized procedure that partitions the grid into sub-grids such that the height and width of a sub-grid are bounded by $2D$ and 2Σ, respectively, and such that in expectation $\text{OPT}' \geq \frac{1}{O(1)}\text{OPT}$, where OPT' denotes the number of messages that can be routed after deleting targets separated from their source.*

Let $I^{\Sigma}_{>s}$ denote the union of all intervals $I^{\Sigma}_r, r > s$, and let $I^n_{>s}$ denote the union of intervals $I^n_r, r > s$. Note that e.g. $I^n_{>s}$ simply specifies the interval of vertical distances smaller than N_s.

Lemma 7. *Given an instance with allowed vertical distances from the interval $I^n_{>3}$ and allowed horizontal distances from the interval $I^{\Sigma}_{>3}$, we can compute a constant factor approximation in time polynomial in n and M.*

Proof. We can use the transformation from Lemma 10 to partition the grid into sub-grids of width at most $2\Sigma_3$ and height at most $2N_3$, such that any source-target pair (or more precisely a grid-source/possible grid-target pair) is contained within a sub-grid with constant probability. Now, we use a constant factor approximation for each sub-grid and thereby obtain (in expectation) a constant factor approximation for the instance.

The constant factor approximation for a sub-grid G uses the following brute-force approach. Note that asymptotically, $\Sigma_3 = O(\log\log\log\Sigma) = O(\log\log\log M)$ and $N_3 = O(\log\log\log n)$. There are at most $P := O(N_3 \cdot \Sigma_3 \cdot \binom{\Sigma_3+N_3}{N_3})$ path connecting end-points in G. Define a candidate solution to be a choice of number of messages along every path in P. We can check a candidate solution in polynomial time for feasibility. The idea is to generate all candidate solutions that may be optimal and choose the best among these. For this, note that any path can carry at most C messages in an optimum solution.

However, this may still result in a large number of candidate solutions as C may be as large as M. Therefore, we round, for every path, the choice of how many messages we want to send along that path, down to powers of two. This only decreases the value of the optimal candidate solution by a factor of two, but dramatically decreases the number of candidate solutions we have to look at.

We have $\lceil \log C \rceil$ choices for every path, which means that the number of candidate solutions is at most $(\log C)^P = O((\log M)^P)$, which is polynomial in M and n. (To see this for M, we consider the ratio $(\log M)^P / M = 2^{P \log\log M} / 2^{\log M}$, which tends to 0 for large enough M since $P = O(2^{O(\log\log\log M + \log\log\log n)}) = O((\log\log M \cdot \log\log n)^c)$ for a constanc c). □

We can also find a constant factor approximation for instances in which one parameter (the maximum slack or the maximum distance) is constant.

Lemma 8. *Given an instance with allowed vertical distances from the interval $I^n_{>\delta_n}$, we can efficiently compute a constant factor approximation.*

Proof. We can again use the transformation presented in Lemma 10 and obtain sub-grids with constant height such that only considering source-target pairs within sub-grids still gives a close to optimum solution.

In such a sub-grid there are only a constant number of different pairs of array locations. We fix a single pair and compute the maximum number of messages that we can send for this pair. Scheduling different messages for the same pair can be done greedily. In each time step the source node in the array simply forwards the $\leq C$ messages that have the earliest deadline among all available messages that can still make it to the target. From the remaining messages it buffers the $\leq B$ messages that have the furthest deadline.

Note, that all other nodes in the array only have to forward the messages they got from the previous time step as there are no messages injected at these nodes.

We run this algorithm for every pair and return the best pair. This gives a constant factor approximation. □

Lemma 9. *Let S denote the original instance of the packet scheduling problem on a grid, and let S' denote an instance obtained from S by only including possible targets with horizontal distance at most Σ_{δ_Σ} from their source (hence, horizontal distances are from the interval $I^{\Sigma}_{>\delta_\Sigma}$). Then we can obtain a constant factor approximation for S'.*

Proof. Using the transformation from Lemma 10 we obtain sub-grids with constant width. We get a constant factor approximation for such a grid as follows.

How many messages can an optimum algorithm send between source-target pairs in a sub-grid? We can increase this number by making all horizontal edges undirected and setting their capacity B to infinity. This new network is equivalent to a single column where all edges have capacity $C' \leq 2\Sigma_{\delta_\Sigma} \cdot C = O(1) \cdot C$, where the bound holds since $2\Sigma_{\delta_\Sigma}$ is an upper bound on the width of the sub-grid.

The scheduling problem on a single column is known as Interval Scheduling and the following Greedy-approach is known to give the optimal solution (see e.g. [10]). Scan the column from bottom to top. For each edge look at the subset of messages that have their source below the edge and schedule those $\leq C'$ messages whose targets are lowest in the column (a variant of Earliest Deadline First).

This means that the optimum algorithm routes at most the number of messages obtainable via an Earliest Deadline First schedule on a single column with edge-capacities C'. Note that for every source our instance S' includes a possible target in the same column. Therefore, if we run an Earliest Deadline First schedule in each column independently, we will route at least the number of messages that can be obtained via an Earliest Deadline First schedule on a single column with edge-capacities C. The throughput of these two solutions can only differ by a factor of $C'/C = O(1)$. Hence, we obtain a constant factor approximation. □

We conclude this section with our main results. Theorem 2 stems from a slight modification of the above algorithm.

Theorem 1. *There is an approximation algorithm for the time constrained packet scheduling problem on line networks that runs in time polynomial in n and M and obtains an expected approximation guarantee $O(\max\{\log^* n - \log^* B, 1\} + \max\{\log^* \Sigma - \log^* C, 1\})$.*

Proof. We generate the following instances, where $X \oplus Y$, means an instance that contains all source-target pairs from our original grid-instance for which the horizontal distance is in interval X and the vertical distance is in interval Y.

- Instances $I^{\Sigma}_i \oplus I^n_j$, $1 \leq i \leq \delta_\Sigma$, $1 \leq j \leq 3$. These are at most $\mathrm{O}(\max\{\log^* \Sigma - \log^* C, 1\})$ instances. These instances are restricted and fulfill the requirement for Lemma 16 in Section 4.

- Instances $I_j^\Sigma \oplus I_i^n$, $1 \leq i \leq \delta_n$, $1 \leq j \leq 3$. These are at most $O(\max\{\log^* n - \log^* B, 1\})$ instances. These instances are restricted and fulfill the requirement for Lemma 16 in Section 4.
- Instance $I_{>3}^\Sigma \oplus I_{>3}^n$. Source-target pairs in this instance have a very small slack and a very small distance in the array. This can be dealt with using Lemma 7.
- Instance $I_{>0}^\Sigma \oplus I_{>\delta_n}^n$. This instance contains source-target pairs that only have to travel a constant distance in the array. This can be dealt with using Lemma 8.
- Instance $I_{>\delta_\Sigma}^\Sigma \oplus I_{>0}^n$. This instance contains source-target pairs that only have constant slack. This can be dealt with using Lemma 9.

Every source-target pair is contained in at least one instance, and we can approximate each instance by a constant factor. There are only $O(\max\{\log^* n - \log^* B, 1\} + \max\{\log^* \Sigma - \log^* C, 1\})$ instances. Hence, the theorem follows. □

Theorem 2. *There is an algorithm with running time polynomial in n and M, that achieves expected approximation ratio of $O(\min\{\log^* n, \log^* \Sigma\})$.*

Proof. Using the proof of Lemma 7 we can get a constant factor approximation by brute-force for the case that horizontal distances are from the interval $I_{>j}^\Sigma$ and vertical distance are from $I_{>i}^n$, for any i and j. The running time for this brute-force solution is $O(\text{poly}(M,n) \cdot (\log C)^P) = O(\text{poly}(M,n) \cdot (\log M)^P)$ where $P = O(N_i \cdot \Sigma_j \cdot \binom{\Sigma_j + N_i}{N_i})$.

Recall that w.l.o.g $M \geq \Sigma$. Consider the case that $\Sigma \geq \exp(\exp(\exp(n)))$. Then we can solve a class of the form $I_{>3}^\Sigma \oplus I_{>0}^n$ by brute-force in time polynomial in M due to the fact that $N_0 = O(\log\log\log M)$ and $\Sigma_3 = O(\log\log\log M)$. Similarly, if $n \geq \exp(\exp(\exp(\Sigma)))$ we can solve a class of the form $I_{>0}^\Sigma \oplus I_{>3}^n$ by brute-force in time polynomial in n.

This means that e.g. in the case $n \geq \exp(\exp(\exp(\Sigma)))$ we can cover all source-target pairs by instances $I_{>0}^\Sigma \oplus I_{>3}^n$, $I_{>\delta_\Sigma}^\Sigma \oplus I_{>0}^n$, and $I_{>i}^\Sigma \oplus I_{>j}^n$, $1 \leq i \leq \delta_\Sigma, 1 \leq j \leq 3$ which are at most $O(\log^* \Sigma)$ instances. An analogous statement holds for the case $\Sigma \geq \exp(\exp(\exp(n)))$. If neither of these cases occurs then $\log^*(\Sigma) = O(\log^* n)$ and we are done. Altogether this gives the theorem. □

4 Restricted Instances

In this section we give a constant factor approximation algorithm for restricted instances. That is, we are given an instance of the packet scheduling problem on the grid, which is also a (D, Σ)-*restricted instance* in the sense of Definition 4. The approximation algorithm works in several steps.

Step 1: Reducing width and height

In the first step we randomly partition our grid into sub-grids such that each sub-grid has height at most $2D$ and width at most 2Σ. We then delete all possible targets for

which the source is not contained in the same sub-grid. The following lemma states that we can do this without changing the throughput of an optimal solution by too much.

Lemma 10. *Given a (D, Σ)-restricted instance there is a randomized procedure that partitions the grid into sub-grids such that the height and width of a sub-grid are bounded by $2D$ and 2Σ, respectively, and such that in expectation $\mathrm{OPT}' \geq \frac{1}{O(1)} \mathrm{OPT}$, where OPT' denotes the number of messages that can be routed after deleting targets separated from their source.*

Proof. We first cut the grid into vertical stripes of width at most 2Σ. For this we choose an offset z uniformly at random from the range $[1, \ldots, 2\Sigma]$. We then cut all edges of the form $((z + k \cdot 2\Sigma, y), (z + k \cdot 2\Sigma + 1, y))$, $k \in \mathbb{Z}_0^+$, which are edges in vertical layers such that the horizontal distance between different layers is 2Σ.

Assume there is a source in column x_s with a possible target in column x_t. Then $x_t - x_s \leq \Sigma$ by the definition of a restricted instance. The target is separated in this step if there is an $x \in \{x_s, \ldots, x_t - 1\}$ with $x \bmod 2\Sigma \equiv z$. Since the range of x is only $x_t - x_s \leq \Sigma$, it holds that with probability $1/2$ we choose a z for which the source and the possible target are not separated.

Repeating the same process for horizontal layers (with distance $2D$ instead of 2Σ) gives again that a source-target pair survives the step with constant probability. Altogether any source-target pair in the instances survives both steps with constant probability. Therefore the optimal throughput only decreases by a constant factor (in expectation). □

Step 2: Solving the fractional problem

After applying the first step we compute the optimum fractional solution for the remaining pairs. This can be done by formulating the problem as a multicommodity flow problem. Let the set $\mathcal{T}_m$, for a given message m with source s_m, denote the set of possible targets for s_m (targets in our restricted instance). We introduce a *meta-source* $\tilde{s}_m$ and a *meta-target* $\tilde{t}_m$ for the message. We connect the *meta-source* to s_m via a directed edge $(\tilde{s}_m, s_m)$ of capacity 1. Further, we connect all possible targets $t \in \mathcal{T}_m$ to the meta-target via directed edges $(t, \tilde{t}_m)$ of infinite capacity. Now, we introduce a commodity between every meta-source/meta-target pair. We solve the maximum multicommodity flow problem on the flow network resulting from the above construction. The resulting fractional solution allows a source to pick fractional paths to connect to its possible targets but the total fractional weight of paths starting at a source s is limited to 1. It is straightforward to see that the fractional multicommodity flow problem is a relaxation of the routing problem defined by our restricted instance. Note that the network on which we solve the fractional solution is of size $\mathrm{poly}(n, \Sigma)$ and the number of commodities is bounded by M. Therefore the fractional solution is computed in time $\mathrm{poly}(n, M)$. In the following our goal is to round this fractional solution to an integral solution. This means that a source will have at most one path (with a weight of 1) to connect to one of its possible targets.

Step 3: Reducing flow within a sub-grid

Note that all the flow paths from the fractional solution have their end-points within one of the sub-grids generated in the first step. Now we partition these sub-grids further in order to guarantee that each resulting piece only contains a flow of value at most $4\min\{DB, \Sigma C\}$. We will delete flow between the different pieces that we generate.

Lemma 11. *We can generate sub-grids from the sub-grids of the first stage such that the total flow within each sub-grid is at most* $4\min\{DB, \Sigma C\}$*, and we only lose a constant fraction of the flow.*

Proof. Let $G = [x_l, \ldots, x_r] \times [y_b, \ldots, y_t]$ denote a subgrid resulting from the first transformation with side-length bounded by $2D$ and 2Σ, respectively. Assume that $\Sigma C \leq DB$ (the other case is analogous; replace ΣC by DB and perform the following cutting scheme from right to left instead of from top to bottom).

In order to obtain the desired partitioning of the grid we first check whether the total flow inside is less than $4\Sigma C$. If this is the case we do not have to perform any cut. Otherwise let y denote the largest row such that the total flow delivered to nodes in $G = [x_l, \ldots, x_r] \times [y, \ldots, y_t] > 4\Sigma C$. We cut the grid horizontally by removing all edges of the form $\{((y-1, x), (y, x)) \mid x \in [x_l \ldots, x_r]\}$. We also remove all flow-paths using these edges. The sub-grid $[x_l, \ldots, x_r] \times [y, \ldots, y_t]$ forms one piece.

This piece contains at most flow $4\Sigma C$, because deleting edges between layer $y-1$ and layer y deletes all flow paths that have a node in layer y as a target. Therefore, the total flow within the piece is at most the flow delivered to vertices $[x_l, \ldots, x_r] \times [y+1, \ldots, y_t]$, which is less than $4\Sigma C$ due to the choice of y. On the other hand, we can argue that the flow inside the piece is at least $2\Sigma C$. Since the width of the stripe is at most 2Σ we are deleting a total flow of $2\Sigma C$ from all the flow that is sent to a final destination in $[x_l, \ldots, x_r] \times [y, \ldots, y_t]$. Since the latter is more than $4\Sigma C$ due to the choice of y, we are left with flow at least $2\Sigma C$ within the piece.

We can amortize the flow that we deleted (at most $2\Sigma C$) against the flow that remains (at least $2\Sigma C$). This ensures that we only cut a constant fraction of the flow paths. By iterating the above procedure on the remaining stripe $[x_l, \ldots, x_r] \times [y_b, \ldots, y-1]$ we get the desired partitioning. □

4.1 Rounding the Transformed Solution

In this section we give a rounding algorithm that rounds the fractional solution that we obtain from Step 3 into an integral solution while only loosing a constant fraction of the flow in this process. We round the solution for each sub-grid resulting from Step 3 independently. Let G denote such a sub-grid. Recall that G has height at most $2D$, width at most 2Σ, and that the flow within G is of value at most $4 \cdot \min\{BD, \Sigma C\}$.

Simply applying randomized rounding (see [13]) to the fractional flow paths will not work as some edge usually gets assigned too many flow paths which renders the solution infeasible. Randomized rounding requires that the edges have a certain minimum capacity. In order to obtain this we do the following. We will first remove source-target pairs

in a randomized way that guarantees that any fixed pair is only deleted with constant probability. This is done in a way that guarantees that the remaining sources and targets are separated by vertical and horizontal layers that are completely empty. These layers will then be used to "simulate" a grid in which edges have a high capacity so that randomized rounding can be performed. Define

$$h := \left\lceil \frac{7}{B} \ln(8e/B \cdot 2D \cdot \min\{BD, \Sigma C\}) \right\rceil \quad , \tag{3}$$

and

$$w := \left\lceil \frac{7}{C} \ln(8e/C \cdot 2\Sigma \cdot \min\{BD, \Sigma C\}) \right\rceil \quad . \tag{4}$$

For simplicity of the following exposition we add padding to the grid G. We add $3w$ empty columns to the left, $3w$ empty columns to the right, $3h$ empty rows at the bottom and $3h$ rows at the top of G. Then we re-number the rows and columns such that the bottom left corner node of G becomes node $(1, 1)$ and the top right node becomes node $(\text{height(G)}, \text{width}(G))$.

Introducing Gadgets

In the following we call a $w \times h$ sub-grid of G a *block*. We design a gadget structure that consists of 7×7 blocks, where we number the blocks according to their rows and columns with bottom-left being block $(1, 1)$ and top-right being block $(7, 7)$. The center block (block $(4, 4)$) is called the *core*. We delay the further description of the gadget-structure to the following section. Here we first describe where to place these gadgets in the grid G. The goal is to place them randomly such that any source or target in G is contained in the core of a gadget with constant probability. A second requirement is that for two nodes u and v that are horizontally and vertically far apart the probability that both of them are in a core is also constant. Finally, we require that the gadgets are not too far apart.

We choose horizontal and vertical grid lines in G such that the y-value y of a horizontal line fulfills $y \bmod h \equiv 1$, and the x-value x of a vertical line fulfills $x \bmod w \equiv 1$. Each linear array of G fulfilling this is chosen with probability $1/7$. A selected vertical (horizontal) line with some x-value x (y-value y) is called *active*, if the successive lines with x-values $x + iw$, $i = 1, \ldots, 6$ (y-values $y + ih$, $i = 1, \ldots, 6$) are *not* selected.

In addition to these active lines we add extra lines in order to fill gaps. As long as there exists a pair of consecutive active horizontal (vertical) lines with vertical (horizontal) distance larger than $20w$ (20h) we add a horizontal line in between in such a way that still each line has $7w$ ($7h$) non-active columns (rows) right of (above) it. Now, let I denote the set of grid nodes that lie in the intersection of two lines (active lines or extra lines). For each node $v \in I$ we place a gadget such that its bottom-left node coincides with v.

In the following we call a node $(x, y) \in G$ with $x \in [3w + 1, \text{width}(G) - 3w - 1]$ and $y \in [3h + 1, \text{height}(G) - 3h - 1]$ a *non-padding* node. Note that all sources and targets are at non-padding nodes.

Proposition 12. *The randomized placement of the gadgets fulfills the following properties.*

1. *The area of the different gadgets are node disjoint.*
2. *A non-padding node (x, y) of G is inside the core area of a gadget with constant probability.*
3. *For non-padding nodes u, v at horizontal distance larger than $7h$ and vertical distance larger than $7w$ the probability that both of them are in a core is constant. Note that this, in particular, holds for a source-target pair because of the requirements of a restricted instance.*
4. *The horizontal (vertical) distance between gadgets is at most $20w$ ($20h$).*

The Structure of a Gadget

A gadget consists of 49 blocks ($w \times h$ sub-grids) arranged in a 7×7 grid. Let the bottom-left block be numbered with $(1, 1)$. The block $(4, 4)$ is the *core* of the gadget. We call the blocks $(1, 2) \cdots (7, 2)$ and $(1, 6) \cdots (7, 6)$ *horizontal highway areas*, while the blocks $(2, 1) \cdots (2, 7)$ and $(6, 1) \cdots (6, 7)$ are called *vertical highway areas*. See Figure 2 for an illustration of a gadget and the placement of gadgets in the grid. The gadgets are aligned into rows and columns of a grid. For each row of gadgets we flip a coin that determines randomly whether we "*build a horizontal highway*" *below* the cores (through highway area $(1, 2) \cdots (7, 2)$) or *above* the cores (through area $(1, 6) \cdots (7, 6)$). We make the analogous randomized choice for each column of gadgets. The idea behind these highways is that only these are used for communicating between different gadgets. A horizontal (vertical) highway consists of h horizontal (w vertical) linear arrays of the grid which we call "lanes" in the following. We will show that we can view this highway system as a collection of high capacity links that connects the gadgets with each other. Because of the high capacity we will be able to use a randomized rounding approach for routing along the highways.

The routing path of a message will consist of three parts. First, we route from the source of a message to the highway. Then we route from the source-gadget to the target-gadget along the highway system and finally, at the target gadget, we route from the highway to the target. Note, however, that e.g. for a source node that does not have a highway above or to the right of its core it is not possible to route to a highway (if only highways are allowed to be used for leaving the gadget area). Therefore, we will delete certain source-target pairs that cannot be routed anymore due to the choice of highways.

4.1.1 The rounding algorithm

In the first step of our rounding algorithm we delete all sources and targets that are not located in a core area. From Proposition 12 it follows that a fixed source-target pair survives with constant probability. Therefore in expectation this step only reduces the (expected) flow by a constant factor. Note that for this step to be possible it is crucial

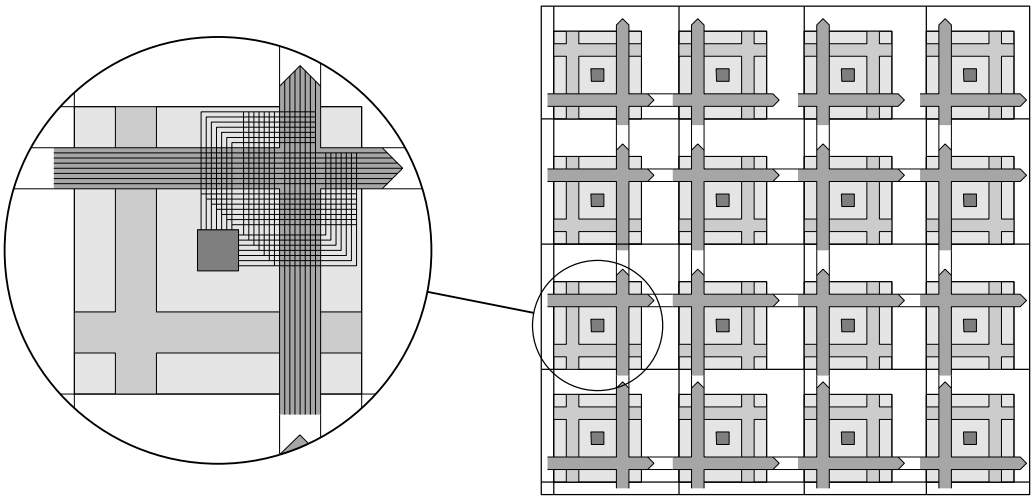

Figure 2: A source gadget, and a possible placement of various gadgets in the grid.

that the routing paths in our grid G connect source-target pairs that are at least $7h$ apart vertically and at least $7w$ apart horizontally. This follows from the definition of restricted instances. It ensures that for any pair, both the source and the target are in a core with constant probability.

We call a gadget a *source-gadget* if it has a highway above and to the right of the core. If a gadget has a highway to the left and to the bottom of the core we call it a target-gadget. After selecting the highways we delete a target if its gadget is not a target-gadget, and a source if its gadget is not a source-gadget. We delete a message if either its source or its target is deleted. A message will survive this step with constant probability. Therefore again we only reduce the flow by a constant factor.

Routing between gadgets

Now, we define a super-grid G' that contains a node for every gadget. Horizontal edges have capacity hB, while vertical edges have capacity wC. We map a source or target of the grid G to its corresponding gadget-node. The following claim states that the grid G' is a better communication network than G (up to a constant factor).

Lemma 13. *Suppose we are given a fractional solution for our problem on the grid G that only consists of flows between terminal pairs located in core areas. We can transform this solution into a solution on G' consisting of flow path connecting the corresponding gadget-nodes. The load on any edge of G' in the new solution is $O(1)$.*

Proof. The gadget-lines that we placed randomly into the grid and whose intersections define the origins of gadgets, partition the grid into disjoint rectangles, which we call regions in the following (to make them node-disjoint we define that a region contains its left and bottom boundary array but not its top and right boundary array). Each region

contains (at most) one gadget, and every gadget is contained in a region. Note that the width of a region is at most $20w$ and the height at most $20h$.

Let p denote a path in G for some message m between source s and target t. Let s' and t' denote the gadget-nodes in G' corresponding to the cores containing s and t, respectively. When we contract the regions into single vertices we obtain a graph isomorphic to G'. The path p translates into a path p' in the contracted graph in the natural way, and we use this path for routing between s' and t'. An edge $e = (A, B)$ in G' is then only used if the path p enters the region corresponding to a gadget B coming from the region corresponding to gadget A. The capacity of all grid-edges connecting these regions is at most a factor 20 larger than the capacity of the edge e in G'. Hence, the load on e will be at most $20 = O(1)$. □

Scaling down the solution given by the above lemma gives us a feasible fractional solution on G' that has nearly optimum throughput (up to a constant factor). The following lemma shows that the edge-capacities in the grid graph G' and their structure allows to transform a fractional solution on G' into an integral solution via randomized rounding.

Lemma 14. *Suppose that we are given a fractional flow solution in G' with throughput f. We can obtain an integral solution that routes a sub-set of the pairs routed in G' and obtains a throughput of $\frac{1}{2}\mu f$ with probability at least $1/4$, where μ denotes a scaling factor less than $1/2$.*

Proof. To prove this lemma we use the standard randomized rounding framework due to Raghavan and Thompson [13].

- Decompose the flow for a message m into different flow-path $p_1^m, \ldots, p_k^m$ and let $\lambda_1^m, \ldots, \lambda_k^m$ denote the weight of the flow paths.
- Route a message m along path p_i^m with probability $\mu\lambda_i^m$ and don't route it at all with probability $1 - \mu \sum_i \lambda_i^m$.

Clearly, the above routine gives us an integral solution with expected throughput μf, and because of Markov inequality the probability that we obtain at least $\frac{1}{2}\mu f$ is at least $1/2$.

What is the probability that there is an edge that exceeds its capacity? The standard way to analyze this is to first calculate the probability that a single edge violates its capacity and then to apply a union bound to get a result for all edges.

However, the grid G' may have $\Theta(D \cdot \Sigma)$ edges. For applying a union bound over that many edges the capacities of the individual edges (i.e., wC and hB, respectively) as given in Equation 3 and Equation 4 are not sufficient. Therefore, we first introduce a *small* number of random variables X_i with the guarantee that if none of the X_i exceeds its expectation by more than a constant factor, then none of the edge capacities are violated. Then we can apply a union bound over these variables.

In the following we only show that no horizontal edge will be violated. The proof for vertical edges is analogous. First, decompose the flow into a set $\mathcal{P}$ of flow-path, and let for a path $p \in \mathcal{P}$, $\lambda(p)$ denote the weight of p. For a subset $\mathcal{Q} \subset \mathcal{P}$ we use

$\lambda(\mathcal{Q}) := \sum_{q\in\mathcal{Q}} \lambda(q)$ to denote the weight of all paths in $\mathcal{Q}$. Let L denote a horizontal line of the grid G', and let $e_1, \ldots, e_k$, $k = O(\Sigma)$ denote the edges on this line ordered from left to right. Let for an edge e_i, $\mathcal{P}_i \subset \mathcal{P}$ denote the set of all paths that use the edge e_i. Clearly, $\forall i : \sum_{p\in\mathcal{P}_i} \lambda(p) \leq B$ as otherwise the edge capacity of edge e_i is violated.

Given a path p, x_p denotes a random variable that is 1 if p is chosen and 0 otherwise. For a consecutive chain of edges $e_i, \ldots, e_j$ let $\mathcal{P}_{ij} := \bigcup_{s=i}^{j} \mathcal{P}_s$. We cut the horizontal line into pieces such that each piece consists of edges $e_\ell, \ldots, e_r$ with $6B \leq \lambda(\mathcal{P}_{\ell r}) \leq 7B$ (we can do this in a greedy manner). This means that each piece "sees" flow paths of total weight at least $6B$ and at most $7B$. Note that for two pieces $\lambda(\mathcal{P}_{\ell_1 r_1} \cap \mathcal{P}_{\ell_2 r_2}) \leq B$ as everything in the intersection of the two sets has to be routed along the lane L from one piece to the next. Therefore, for each piece P there exist flow-paths of total weight $4B$ that are not contained in the piece preceding P in the horizontal array, and are also not contained in the piece following P. Hence, these flow paths are exclusive to the piece (among all pieces of the same horizontal array). Because of this we can bound the number of pieces by the total weight of flow paths. This gives that there are at most $\frac{1}{B}\min\{BD, \Sigma C\}$ pieces in any row and at most $\frac{1}{B} \cdot 2D \cdot \min\{BD, \Sigma C\}$ horizontal pieces in total (Here we ignore rows introduced by padding as these don't carry any flow).

We now introduce a random variable $X(\mathcal{P}_{\ell r}) := \sum_{p\in\mathcal{P}_{\ell r}} x_p$ for each piece. Clearly, if each of these variables is bounded by some value Z after randomized rounding then so is the load on a horizontal edge in G'. A variable $X(\mathcal{P}_{\ell r})$ is a sum of negatively correlated binary random variables with mean at most $7\mu B$ (recall that μ denotes the scaling parameter for our randomized rounding process). Chernoff bounds [7] (Lemma 18 with $\tilde{\mu} = 7\mu B$, and $W = 1$) tell us that $Pr[X(\mathcal{P}_{\ell r}) \geq (1+\delta)7\mu B] \leq e^{-7/3\delta\mu B} \leq e^{-\delta\mu B}$ for $\delta \geq 1$. Now, choosing $\delta = [\frac{1}{\mu B}\ln(\frac{8e}{B} \cdot 2D \cdot \min\{DB, \Sigma C\})]$-1 (then still $(1+\delta)7\mu \leq hB$) gives that a fixed piece has load larger than hB with probability at most $e/[\frac{8e}{B} \cdot 2D \cdot \min(DB, \Sigma C)]$. Since, the number of pieces is at most $1/B \cdot 2D \cdot \min\{DB, \Sigma C\}$ we get that the probability that any of the horizontal edges is violated is at most $1/8$. Via an analogous argument one can show that the probability of violating a vertical edge is only $1/8$. Hence, with probability at least $\frac{1}{2} - 2 \cdot \frac{1}{8} = \frac{1}{4}$ no edge capacity is violated and we have throughput at least $\frac{1}{2}\mu f$. □

Transferring the solution back to G

We can use this rounded solution to also derive an integral solution on G. The proof of the next lemma shows how to do that. Combining this with the results from the previous sections gives the following lemma.

Lemma 15. *Suppose we are given a fractional solution on G with throughput f. We can find an integral solution on G with throughput $\lambda \cdot f$ for a constant $\lambda > 0$.*

Proof. Given the flow with throughput f in G we can route a corresponding integral flow with throughput $\Omega(f)$ in G'. It remains to show that we can also route a large fraction of this flow in G without violating edge capacities. We first claim that we can use the solution in G' to route between gadgets in G. This means, for a message between gadget-nodes u and v in G' that correspond to gadgets U and V in G, that we can reserve an

exclusive path that starts on the highway leaving U and ends on the highway entering V.

We more or less want to map a path in G' to a path in G. For this we always route a message that uses an edge $e = (u, v)$ in G' on the highway connecting the gadgets U and V. However, we have to be very careful about the lane a message chooses because the grid G is directed, and this does not allow a message to switch lanes arbitrarily.

For a routing path in G' we define a horizontal and vertical segment to be a maximal set of consecutive horizontal and vertical edges, respectively. We schedule the messages by always routing one segment at a time. Suppose that we already have (partially) routed some messages. We identify a node $u \in G'$ that does not have any messages that still have to be routed to it. This means there may be some messages that are already routed *through* the gadget U corresponding to u; there may be some messages whose most recently routed segment ends at U, but there are no messages whose routing path in G' contain u, but the message has not "*reached*" U yet. Because G' is a DAG we can always find such a node.

We first route the next segment for all messages whose last routed segment ends at U and which have to go further (this means that these messages "*turn*" at U, i.e., they change their direction from horizontal to vertical or vice versa). Let $m_1^h, \ldots, m_a^h$ denote the messages that arrived at U horizontally (and hence have to leave vertically), and $m_1^v, \ldots, m_b^v$ denote messages that arrive at U vertically (and leave horizontally). The order among those messages is with respect to their lanes. This means for $p < q$, m_p^h uses on its most recent segment a lane below the lane used by m_q^h. Similarly, m_p^v uses a lane left to the lane used by m_q^v. We say a lane is *free* for outward traffic if it is not used by a message that is already routed through u (and hence has reserved its lane for maybe a lot longer). We consider messages in order of their lane from bottom to top (left to right) and assign it to the leftmost (bottom-most) lane that is still free. Routing from the incoming lane to the outgoing lane is done in the area of the grid where the two highways meet and form a crossbar-structure. This routing can be done in an edge-disjoint manner as long as the number of incoming or outgoing messages on a highway is less than the width of the highway. The latter property is guaranteed by the integral solution on G'. See Figure 3 for an example.

Finally, we have to route the messages that start in the core of U. We simply assign a unique free lane to them in an arbitrary manner. This works since the number of lanes in a highway is as large as the capacity of the corresponding edge in G'.

After choosing for each message the lane of the next segment we reserve this lane exclusively from U to the next gadget where the message has to turn. Observe that if say a horizontal lane is free for a gadget U (and therefore the above process assigns a message to this lane), then it is also free for a gadget X that a) comes after U in U's row and that b) still has to be used by some of the messages currently at U. This holds because the only way that the lane could be blocked is that we routed a message turning at X (or a predecessor of X) onto this lane and thereby blocked it. However, this is not possible since X (or the predecessor) did not yet receive all its messages (e.g. one message from U is still missing). So we would not have chosen it for handling the messages turning there. Because of this property it is guaranteed that each message

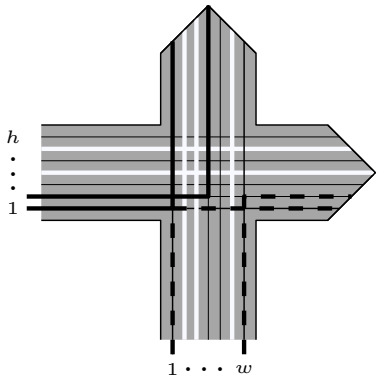

Figure 3: An example for switching lanes at a crossing. The white lines indicate lanes that are blocked by messages that go straight through the gadget. The messages can be greedily scheduled by assigning e.g. a message arriving on the lowest horizontal lane the leftmost vertical lane that is available.

from u can route its next segment in an edge-disjoint manner.

It remains to route from the sources inside the core of a source-gadget to the starting point on the highway, and from the endpoint of a highway in a target-gadget to the individual targets in the core. This part can be solved by applying techniques used for finding edge-disjoint paths in undirected grids (see [8]). The following is an adaption of the corresponding proof from [8] and is contained for completeness.

Consider e.g. a source-core as depicted in Figure 2. We only need to route all sources in an edge-disjoint manner to the border of the core. Routing from there to any lane on one of the highways is straightforward. Consider adding a super-source s to all sources inside the core and a super-target t to the boundary nodes at the top and the right. Let S denote the number of sources inside the core. If there exists a flow of value S between s and t then this flow is integral (or can be made integral) and can be decomposed into edge-disjoint paths since all edges have the same capacity.

Hence, we can find edge-disjoint paths if and only if the demand across any cut inside the core does not violate its capacity. Furthermore, the bottleneck cut is attained at a sub-rectangle of the core.

In the following we delete a source that was originally chosen by the randomized rounding step on G' if it is contained in a rectangle that forms a violated cut. We have to analyze the probability that a source has to be deleted after it was chosen in the randomized rounding.

Fix a source s and a rectangle R with dimensions $x \times y$ such that $s \in R$. Let S denote the set of sources in R excluding s. The fractional flow from those sources after scaling down the fractional solution by the factor μ used in the randomized rounding is at most $\mu(x + y)$.[4] If, after rounding, the demand for these sources is larger than $(x + y) - 1 \geq \frac{1}{2}(x + y)$, then the cut formed by R is violated.

[4]This holds because the fractional solution on G' is derived from a solution on G. This would not be feasible if sources in R were sending more than $x + y$ flow.

The integral demand is a sum of independent binary random variables. Applying a Chernoff bound (Corollary 19 with $E[X] = \mu(x+y)$ and $\alpha = \frac{1}{2\mu}$) gives

$$\Pr[X \geq \frac{1}{2\mu} \cdot \mu(x+y)] \leq (\sqrt{2e\mu})^{x+y} = z^{x+y} \ ,$$

where we use $z := \sqrt{2e\mu}$.

The probability that there exists any rectangle R for s such that the cut formed by it is violated, is bounded by the infinite sum $\sum_x \sum_y xy \cdot z^{x+y}$, since there are xy rectangles of dimension $x \times y$. This sum is equal to $\frac{z^2}{(1-z)^4}$ which can be made an arbitrary small constant by choosing the scaling factor μ suitably small. Hence, we obtain the result that a source at this stage only has to be deleted with a small constant probability. An analogous argument holds for targets. Hence, at this stage a message is only deleted with a small constant probability. All remaining messages can be routed in an edge-disjoint manner. □

We can now state the main lemma of this section.

Lemma 16. *For a restricted instance of the packet scheduling problem there is a polynomial time algorithm that computes an integral solution which is only a constant factor away from the optimal fractional throughput.*

References

[1] Micah Adler, Sanjeev Khanna, Rajmohan Rajaraman, and Adi Rosén. Time-constrained scheduling of weighted packets on trees and meshes. *Algorithmica*, 36(2):123–152, 2003.

[2] Micah Adler, Arnold L. Rosenberg, Ramesh K. Sitaraman, and Walter Unger. Scheduling time-constrained communication in linear networks. *Theory of Computing Systems*, 35(6):599–623, 2002.

[3] William Aiello, Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Dynamic routing on networks with fixed-size buffers. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 771–780, 2003.

[4] Stanislav Angelov, Sanjeev Khanna, and Keshav Kunal. The network as a storage device: Dynamic routing with bounded buffers. In *Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 1–13, 2005.

[5] Yossi Azar and Rafi Zachut. Packet routing and information gathering in lines, rings and trees. In *Proceedings of the 13th European Symposium on Algorithms (ESA)*, pages 484–495, 2005.

[6] Eyal Gordon and Adi Rosén. Competitive weighted throughput analysis of greedy protocols on DAGs. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 227–236, 2005.

[7] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[8] Jon Kleinberg. *Approximation Algorithms for Disjoint Paths Problems*. PhD thesis, Massachusetts Institute of Technology, 1996.

[9] Jon Kleinberg and Éva Tardos. Approximations for the disjoint paths problem in high-diameter planar network. *Journal of Computer and System Sciences*, 57(1):61–73, 1998.

[10] Jon M. Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[11] Joseph (Seffi) Naor, Adi Rosén, and Gabriel Scalosub. Online time-constrained scheduling in linear networks. In *Proceedings of the 24th Conference of the IEEE Communications Society (INFOCOM)*, pages 855–865, 2005.

[12] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM Journal on Computing*, 26(2):350–368, 1997.

[13] Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[14] Jennifer Rexford, John Hall, and Kang G. Shin. A router architecture for real-time point-to-point networks. *Computer Architecture News*, 24(2):237–246, 1996.

A Maximum Slack Σ vs. Number of Messages M

Lemma 17. *Without loss of generality we can assume that Σ, the maximum slack, is at most M, the number of messages in the instance.*

Proof. We show that there exists an optimal solution such that the effective slack used by any message is at most M (by effective slack of a message in a solution we mean the number of time steps in the solution in which the message is idle). Consider the optimal solution (i.e., a solution that delivers the maximum number of messages by their deadlines), which also minimized the sum of effective slacks of all messages. We consider this solution as a set of paths on the grid (see Section 2). Assume towards a contradiction that there is a message $m = (s, t, r, d)$ with effective slack more than M. There must then exist a vertical line in the grid, with x coordinate in $[r - s + n, r - s + n + M + 1]$, such that no message arrives to its target on that line (as there are at most M messages altogether). Since the path of m itself crosses this line there is at least one path that

crosses this line. Now consider among those the path which is the "highest" in the grid. i.e., a path of a message such that no other path blocks the way to a target of that message. Then the solution can be converted such that this message is delivered earlier, thus contradicting the fact that we started with a solution with minimum sum of effective slacks. □

B Chernoff bounds

We use the following form of the Chernoff-Hoeffding bounds. For independent variables it has been shown by Hoeffding [7]. The extension to negatively correlated binary random variables is due to Panconesi and Srinivasan [12].

Lemma 18 ([12])**.** *Let $X_1, \ldots, X_n$ be negatively correlated binary random variables. Let $X = \sum_{i=1}^{n} X_i$, $\tilde{\mu} \geq E[X]$. Then for $\delta \geq 1$,*

$$\Pr[X \geq (1+\delta)\tilde{\mu}] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\tilde{\mu}} \leq e^{-\frac{\delta\tilde{\mu}}{3}} \quad .$$

Corollary 19. *Setting $\tilde{\mu} = E[X]$, we get*

$$\Pr[X \geq \alpha E[X]] \leq \left(\frac{e}{\alpha}\right)^{\alpha E[X]} \quad .$$

C Proof of Proposition 5

Proof. We only prove the proposition for δ_n. Recall that $N_0 = n$ and $N_i = \frac{c}{B} \ln N_{i-1}$. How often do we have to apply this recursive step until we reach a value N_s that is constant?

Claim 20. *Suppose that $\ln^{(r)}(n) \geq \frac{2c}{B}$, $\forall r < s$. Then*

$$N_s \leq \frac{2c}{B} \ln^{(s)} n \quad .$$

Proof. Proof by Induction. In the base case ($s = 1$) we have $N_1 = \frac{c}{B} \ln n \leq \frac{2c}{B} \ln^{(1)} n$. For the induction step ($s - 1 \longrightarrow s$) assume that the condition $\ln^{(r)} n \geq \frac{2c}{B}$ holds for all $r < s$. Then by the induction hypothesis we get $N_{s-1} \leq \frac{2c}{B} \ln^{(s-1)} n$. Now,

$$N_s = \frac{c}{B} \ln(N_{s-1}) \leq \frac{c}{B} \ln\left(\frac{2c}{B} \cdot \ln^{(s-1)} n\right)$$

$$\leq \frac{2c}{B} \ln \ln^{(s-1)} n = \frac{2c}{B} \ln^{(s)} n \quad ,$$

since $\frac{2c}{B} \leq \ln^{(s-1)} n$. □

In order to choose our parameter δ_n we apply the recursion and stop at a value s for which either $\ln^{(s)} n \leq \frac{2c}{B}$ or $N_s \leq 8e$. If we choose δ_n as this value of s we get that all $i < \delta_n$ fulfil $N_i \geq 8e$ and either $N_{\delta_n} \leq \frac{2c}{B} \ln^{(s)} n \leq (\frac{2c}{B})^2 = O(1)$ or $N_{\delta_n} \leq 8e = O(1)$.

How many steps do we need (or how large may δ_n be)? Let s denote the value when we stop the above process. Then

$$1 \leq 8e \leq N_{s-1} \leq \frac{2c}{B} \ln^{(s-1)} n \ ,$$

as otherwise we already would have stopped the process at $s-1$. It boils down to the question how large s can be such that still $\ln^{(s-1)} n \geq \frac{B}{2c}$. Asking the same question for the logarithm to base 2 (log) instead of the natural logarithm (ln) only increases the value.

The latter is at most $\max\{\log^* n - \log^*(B/2c), 0\} \leq \max\{\log^* n - \log^* B, 0\} + O(1)$. $\square$