A Constant Approximation Algorithm for Scheduling Packets on Line Networks*

Guy Even[1], Moti Medina[2], and Adi Rosén[3]

1 **Tel Aviv University, Tel Aviv, Israel**
guy@eng.tau.ac.il
2 **MPI for Informatics, Saarbrücken, Germany**
mmedina@mpi-inf.mpg.de
3 **CNRS and Université Paris Diderot, Paris, France**
adiro@liafa.univ-paris-diderot.fr

Abstract

In this paper we improve the approximation ratio for the problem of scheduling packets on line networks with bounded buffers with the aim of maximizing the throughput. Each node in the network has a local buffer of bounded size B, and each edge (or link) can transmit a limited number c of packets in every time unit. The input to the problem consists of a set of packet requests, each defined by a source node, a destination node, and a release time. We denote by n the size of the network. A solution for this problem is a schedule that delivers (some of the) packets to their destinations without violating the capacity constraints of the network (buffers or edges). Our goal is to design an efficient algorithm that computes a schedule that maximizes the number of packets that arrive to their respective destinations.

We give a randomized approximation algorithm with constant approximation ratio for the case where the buffer-size to link-capacity ratio, B/c, does not depend on the input size. This improves over the previously best result of $O(\log^* n)$ [11]. Our improvement is based on a new combinatorial lemma that we prove, stating, roughly speaking, that if packets are allowed to stay put in buffers only a limited number of time steps, $2d$, where d is the longest source-destination distance, then the optimal solution is decreased by only a constant factor. This claim was not previously known in the integral (unsplitable, zero-one) case, and may find additional applications for routing and scheduling algorithms.

While we are not able to give the same improvement for the related problem when packets have hard deadlines, our algorithm does support "soft deadlines". That is, if packets have deadlines, we achieve a constant approximation ratio when the produced solution is allowed to miss deadlines by at most $\log n$ time units.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases approximation algorithms, linear programming, randomized rounding, packet scheduling, admission control

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.40

1 Introduction

In this paper we give an approximation algorithm with an improved approximation ratio for a network-scheduling problem which has been studied in numerous previous works in a number of variants (cf. [2, 3, 5, 8, 14, 11]). The problem consists of a directed line network

* The full version of this paper can be found in http://arxiv.org/abs/1602.06174.

24th Annual European Symposium on Algorithms (ESA 2016).
Editors: Piotr Sankowski and Christos Zaroliagis; Article No. 40; pp. 40:1–40:16

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

over nodes $\{0, \ldots, n-1\}$, where each node i can send packets to node $i+1$, and can also store packets in a local buffer. The maximum number of packets that can be sent in a single time unit over a given link is denoted by c, and the number of packets each node can store at any given time is denoted by B. An instance of the problem is further defined by a set of packets $r_i = (a_i, b_i, t_i)$, $1 \leq i \leq M$, where a_i is the source node of the packet, b_i is its destination node, and $t_i \geq 1$ is the release time of the packet at vertex a_i. The goal is that of maximizing the number of packets that reach their respective destinations without violating the links or the buffers capacities. See Section 2 for a formal definition of the problem.

We present a randomized approximation algorithm for that problem, which has a *constant* approximation ratio for the case that the ratio B/c does not depend on the input size, improving upon the previous $O(\log^* n)$ approximation ratio given in [11, Theorem 3]. While this constant approximation result does not hold for the variant of the problem where packets have deadlines, our algorithm does provide a constant-approximation solution that abides to "soft deadlines". That is, in that solution each packet is delivered at most $\log n$ time units past its deadline.

Our algorithm is based on a novel combinatorial lemma 3 which states the following. Consider a set of packets such that all source-destination distances are bounded from above by some d. The throughput of an optimal solution in which every packet r_i must reach its destination no later than time $t_i + 2d$ is an $\Omega(B/c)$-fraction of the unrestricted optimal throughput. This lemma plays a crucial role in our algorithm, and we believe that it may find additional applications for scheduling and routing algorithms in networks. We emphasize that the fractional version of a similar property, i.e., when packets are splitable and one accrues a benefit also from the delivery of partial packets, presented first in [5], does not imply the integral version that we prove here.

We emphasize that the problem studied here, namely, maximizing the throughput on a network with bounded buffers, has resisted substantial efforts in its (more applicable) distributed, online setting, even for the simple network of a directed line. Indeed, even the question whether or not there exists a constant competitive online distributed algorithm for that problem on the line network remains unanswered at this point. We therefore study here the offline setting with the hope that, in addition to its own interest, results and ideas from this setting will contribute to progress on the distributed problem.

Related Work. The problem of scheduling packets so as to maximize the throughput (i.e., maximize the number of packets that reach their destinations) in a network with bounded buffers was first considered in [2], where this problem is studied for various types of networks in the distributed setting. The results in that paper, even for the simple network of a directed line, were far from tight but no substantial progress has been made since on the realistic, distributed and online, setting. This has motivated the study of this problem in easier settings, as a first step towards solving the realistic, possibly applicable, scenario.

Angelov et al. [3] give centralized online randomized algorithms for the line network, achieving an $O(\log^3 n)$-competitive ratio. Azar and Zachut [5] improved the randomized competitive ratio to $O(\log^2 n)$ which was later improved by Even and Medina [6, 8] to $O(\log n)$. A deterministic $O(\log^5 n)$-competitive algorithm was given in [7, 8], which was later improved in [9] to $O(\log n)$ if buffer and link capacities are not very small (not smaller than 5).

The related problem of maximizing the throughput when packets have deadlines (i.e., a packet is counted towards the quality of the solution only if it arrives to its destination before a known deadline) on line network with unbounded input queues is known to be NP-hard [1].

The same problem in a variant of the setting, where the input queues are bounded, is shown in [11] to have a $O(\log^* n)$-approximation randomized algorithm. The setting in the present paper is the same setting as the one of the latter paper, and the results of [11] immediately give an $O(\log^* n)$-approximation randomized algorithm for the problem and setting we study in the present paper.

2 Preliminaries

2.1 Model and problem statement

We consider the standard model of synchronous store-and-forward packet routing networks [2, 3, 5]. The network is modeled by a directed path over n vertices. Namely, the network is a directed graph $G = (V, E)$, where $V = \{0, \ldots, (n-1)\}$ and there is a directed edge from vertex u to vertex v if $v = u + 1$. The network resources are specified by two positive integer parameters B and c that describe, respectively, the local buffer capacity of every vertex and the capacity of every edge. In every time step, at most B packets can be stored in the local buffer of each vertex, and at most c packets can be transmitted along each edge.

The input consists of a set of packet requests $R = \{r_i\}_{i=1}^{M}$. A packet request is specified by a 3-tuple $r_i = (a_i, b_i, t_i)$, where $a_i \in V$ is the *source node* of the packet, $b_i \in V$ is its *destination node*, and $t_i \in \mathbb{N}$ is the release time of the packet at vertex a_i. Note that $b_i > a_i$, and r_i is ready to leave a_i in time step t_i.

A solution is a schedule S. For each request r_i, the schedule S specifies a sequence s_i of transitions that packet r_i undergoes. A *rejected* request r_i is simply discarded at time t_i, and no further treatment is required (i.e., $s_i = \{reject\}$). An *accepted* request r_i is delivered from a_i to b_i by a sequence s_i of actions, where each action is either "store" or "forward". Consider the packet of request r_i. Suppose that in time t the packet is in vertex v. A store action means that the packet is stored in the buffer of v, and will still be in vertex v in time step $t + 1$. A forward action means that the packet is transmitted to vertex $v + 1$, and will be in vertex $v + 1$ in time step $t + 1$. The packet of request r_i reaches its destination b_i after exactly $b_i - a_i$ forward steps. Once a packet reaches its destination, it is removed from the network and it no longer consumes any of the network's resources.

A schedule must satisfy the following constraints:

1. The *buffer capacity constraint* asserts that at any time step t, and in every vertex v, at most B packets are stored in v's buffer.
2. The *link capacity constraint* asserts that at any step t, at most c packets can be transmitted along each edge.

The *throughput* of a schedule S is the number of accepted requests. We denote the throughput of a schedule S by $|S|$. As opposed to online algorithms, there is no point in injecting a packet to the network unless it reaches its destination. Namely, a packet that is not rejected and does not reach its destination only consumes network resources without any benefit. Hence, without loss of generality, we assume that every packet that is dropped before reaching its designation is rejected at its source node at its release time.

We consider the offline optimization problem of finding a schedule that maximizes the throughput. We propose a centralized constant-ratio approximation algorithm. By *offline* we mean that the algorithm receives all requests in advance[1]. By *centralized* we mean that all

[1] The number of requests M is finite and known in the offline setting. This is not the case in the online setting in which the number of requests is not known in advance and may be unbounded.

the requests are known in one location where the algorithm is executed. Let $\mathsf{opt}(R)$ denote a schedule of maximum throughput for the set of requests R. Let $\mathsf{alg}(R)$ denote the schedule computed by alg on input R. We say that the approximation ratio of a scheduling algorithm alg is c if $\forall R : |\mathsf{alg}(R)| \geq c \cdot |\mathsf{opt}(R)|$. For a randomized algorithm we say that the expected approximation ratio is c if $\forall R : \mathbf{E}\left[|\mathsf{alg}(R)|\right] \geq c \cdot |\mathsf{opt}(R)|$.

The Max-Pkt-Line Problem. The problem of maximum throughput scheduling of packet requests on directed line (Max-Pkt-Line) is defined as follows. The input consists of: n - the size of the network, B - node buffer capacities, c - link capacities, and M packet requests $\{r_i\}_{i=1}^{M}$. The output is a schedule S. The goal is to maximize the throughput of S.

2.2 Path Packing in a uni-directed 2D-Grid

In this section we define a problem of maximum cardinality path packing in a two-dimensional uni-directed grid (Max-Path-Grid). This problem is equivalent to Max-Pkt-Line, and was used for that purpose in previous work, where the formal reduction is also presented [4, 1, 5, 11]. As the two problems are equivalent, we use in the sequel terminology from both problems interchangeably.

The grid, denoted by $G^{st} = (V^{st}, E^{st})$, is an infinite directed acyclic graph. The vertex set V^{st} equals $V \times \mathbb{N}$, where $V = \{0, 1, \ldots, (n-1)\}$. Note that we use the first coordinate (that corresponds to vertices in V) for the y-axis and the second coordinate (that corresponds to time steps) for the x-axis. The edge set consists of horizontal edges (also called store edges) directed to the right and vertical edge (also called forward edges) directed upwards. The capacity of vertical edges is c and the capacity of horizontal edges is B. We often refer to G^{st} as the space-time grid (in short, grid) because the x-axis is related to time and the y-axis corresponds to the vertices in V.

A *path request* in the grid is a tuple $r^{st} = (a_i, t_i, b_i)$, where $a_i, b_i \in V$ and $t_i \in \mathbb{N}$. The request is for a path that starts in node (a_i, t_i) and ends in any node in the row of b_i (i.e., the end of the path can be any node (b_i, t), where $t \geq t_i$).

A *packing* is a set of paths S^{st} that abides the capacity constraints. For every grid edge e, the number of paths in S^{st} that contain e is not greater than the capacity of e.

Given a set of path requests $R^{st} = \{r_i^{st}\}_{i=1}^{M}$, the goal in the Max-Path-Grid problem is to find a packing S^{st} with the largest cardinality. (Each path in S^{st} serves a distinct path request.)

Multi-Commodity Flows (MCFs). Our use of path packing problems gives rise to *fractional* relaxations of that problem, namely to multi-commodity flows (MCFs) with unit demands on uni-directional grids. We deferred the definitions and terminology of MCFs to the full version.

2.3 Tiling, Classification, and Sketch Graphs

To define our algorithm we make use of partitions of the space-time grid described above into sub-grids. We define here the notions we use for this purpose. In this section we focus on the case of unit capacities, namely, $B = c = 1$. An extension to other values of B and c can be found [8].

Tiling. *Tiling* is a partitioning of the two-dimensional space-time grid (in short, grid) into squares, called *tiles*. Two parameters specify a tiling: the side length k, an even integer,

of the squares and the shifting (φ_x, φ_y) of the squares. The shifting refers to the x- and y-coordinates of the bottom left corner of the tiles modulo k. Thus, the tile $T_{i,j}$ is the subset of the grid vertices defined by

$$T_{i,j} \triangleq \{(v,t) \in V \times \mathbb{N} \mid ik \leq v - \varphi_y < (i+1)k \text{ and } jk \leq t - \varphi_x < (j+1)k\},$$

where φ_x and φ_y denote the horizontal and vertical shifting, respectively. We consider two possible shifts for each axis, namely, $\varphi_x, \varphi_y \in \{0, k/2\}$.

Quadrants and Classification. Consider a tile T. Let (x', y') denote the lower left corner (i.e., south-west corner) of T. The *south-west quadrant* of T is the set of vertices (x, y) such that $x' \leq x \leq x' + k/2$ and $y' \leq y \leq y' + k/2$.

For every vertex (x, y) in the grid, there exists exactly one shifting $(\varphi_x, \varphi_y) \in \{0, k/2\}^2$ such that (x, y) falls in the south-west (SW) quadrant of a tile. Fix the tile side length k. We define a *class* for every shifting (φ_x, φ_y). The class that corresponds to the shifting (φ_x, φ_y) consists of all the path requests r_i^{st} whose origin (a_i, t_i) belongs to a SW quadrant of a tile in the tiling that uses the shifting (φ_x, φ_y).

Sketch graph and paths. Consider a fixed tiling. The *sketch graph* is the graph obtained from the grid after coalescing each tile into a single node. There is a directed edge (s_1, s_2) between two tiles s_1, s_2 in the sketch graph if there is a directed edge $(\alpha, \beta) \in E^{st}$ such that $\alpha \in s_1$ and $\beta \in s_2$. Let p^s denote the projection of a path p in the grid to the sketch graph. We refer to p^s as the *sketch path* corresponding to p. Note that the length of p^s is at most $\lceil |p|/k \rceil + 1$.

3 Outline of our Algorithm

For the sake of simplicity we focus hereafter on the case of unit capacities, namely $B = c = 1$. Extension to non-unit capacities are discussed in Section 6.1.

Packet requests are categorized into three categories: short, medium, and long, according to the source-destination distance of each packet. A separate approximation algorithm is executed for each category. The algorithm returns a highest throughput solution among the solutions computed for the three categories.

Notation. Two thresholds are used for defining short, medium, and long requests: $\ell_M \triangleq 3 \ln n, \ell_S \triangleq 3 \cdot \ln(\ell_M) = 3 \cdot \ln(3 \ln n)$.

▶ **Definition 1.** A request r_i is a *short* request if $b_i - a_i \leq \ell_S$. A request r_i is a *medium* request if $\ell_S < b_i - a_i \leq \ell_M$. A request r_i is a *long* request if $b_i - a_i > \ell_M$.

We use a deterministic algorithm for the class of short packets, and in Theorem 7 we prove that this deterministic algorithm achieves a constant approximation ratio. We use a randomized algorithm for each of the classes of medium and long packets; in Theorem 15 we prove that this randomized algorithm achieves a constant approximation ratio in expectation for each of these classes. Thus, we obtain the following corollary.

▶ **Corollary 2** (Main Result). *If $B = c = 1$, then there exists a randomized approximation algorithm for the Max-Pkt-Line problem that achieves a constant approximation ratio in expectation.*

In Section 6.1 we discuss non-unit capacities, give the approximation ratio for this case and show that we achieve a constant approximation ratio as long as the ratio B/c does not depend on the input size.

4 Approximation Algorithm for Short Packets

In this section we present a constant ratio deterministic approximation algorithm for short packets. This algorithm, which is key to achieving the results of the present paper, makes use of a new combinatorial lemma that we prove in the next subsection, stating, roughly speaking, that if packets from a given set of packets are allowed to stay put in buffers (i.e., use horizontal edges in the grid) only a limited number of time steps, $2d$ (where d is the longest source-destination distance in the set of packets), then the optimal solution is decreased by only a constant factor. We believe that this lemma may find additional applications in future work on routing and scheduling problems.

4.1 Bounding Path Lengths in the Grid

In this section we prove that bounding, from above, the number of horizontal edges along a path incurs only a small reduction in the throughput. Previously known bounds along these lines hold only for fractional solutions [5], while we present here the first such claim for integral schedules.

Let R_d denote a set of packet requests r_i, $i \geq 1$, such that $b_i - a_i \leq d$ for any i. Consider the paths in the space-time grid that are allocated to the accepted requests. We prove that restricting the path lengths to $2d$ decreases both the optimal fractional and the optimal *integral* throughput only by a multiplicative factor of $O(c/B)$. We note that if the ratio B/c is a constant, then we are guaranteed an optimal solution which is only a constant away from the unrestricted optimal solution.

Notation. For a single commodity acyclic flow f_i, let $p_{\max}(f_i)$ denote the diameter of the support of f_i (i.e., length of longest path[2]). For an MCF $F = \{f_i\}_{i \in I}$, let $p_{\max}(F) \triangleq \max_{i \in I} p_{\max}(f_i)$. Let $F^*_{frac}(R)$ (respectively, $F^*_{int}(R)$) denote a maximum throughput fractional (resp., integral) MCF with respect to the set of requests R. Similarly, let $F^*_{frac}(R \mid p_{\max} < d')$ (respectively, $F^*_{int}(R \mid p_{\max} < d')$) denote a maximum throughput fractional (resp., integral) MCF with respect to the set of requests R subject to the additional constraint that the maximum path length is at most d'.

▶ **Lemma 3.**

$$F^*_{frac}(R_d \mid p_{\max} \leq 2d) \geq \frac{c}{B+2c} \cdot F^*_{frac}(R_d),$$
$$F^*_{int}(R_d \mid p_{\max} \leq 2d) \geq \frac{c}{2(B+c)} \cdot F^*_{int}(R_d).$$

Proof. Partition the space-time grid into *slabs* S_j of "width" d. Slab S_j contains the vertices (v,k), where $k \in [(j-1) \cdot d, j \cdot d]$, $j \geq 1$. We refer to vertices of the form (v, jd) as the *boundary* of S_j. Note that if $v - u \leq d$, then the forward-only vertical path from (u, jd) to $(v, jd + (v-u))$ is contained in slab S_{j+1}.

[2] Without loss of generality, we may assume that each single commodity flow f_i is acyclic.

We begin with the fractional case. Let $f^* = F^*_{frac}(R_d)$ denote an optimal fractional solution for R_d. Consider request r_i and the corresponding single commodity flow f^*_i in f^*. Decompose f^*_i to flow-paths $\{p_\ell\}_\ell$. For each flow-path p_ℓ in f^*_i, let p'_ℓ denote the prefix of p_ℓ till it reaches the boundary of a slab. Note that $p'_\ell = p_\ell$ if p_ℓ is confined to a single slab. If $p'_\ell \subsetneq p_\ell$, then let (v, jd) denote the last vertex of p'_ℓ. Namely, the path p'_ℓ begins in $(a_i, t_i) \in S_j$ and ends in (v, jd). Let q''_ℓ denote the forward-only path from (v, jd) to $(b_i, jd + (b_i - v))$. (If $p'_\ell = p_\ell$, then q''_ℓ is an empty path.) Note that q''_ℓ is confined to the slab S_{j+1}. We refer to the vertex (v, jd) in the intersection of p'_ℓ and q''_ℓ as the *boundary* vertex. Let g_i denote the fractional single commodity flow for request r_i obtained by adding the concatenated flow-paths $q_\ell \triangleq p'_\ell \circ q''_\ell$ each with the flow amount of f^*_i along p_ℓ. Define the MCF g by $g(e) \triangleq \sum_{i \in I} g_i(e)$. For every edge e, part of the flow $g(e)$ is due to prefixes p'_ℓ, and the remaining flow is due to suffixes q''_ℓ. We denote the part due to prefixes by $g_{pre}(e)$ and refer to it as the prefix-flow. We denote the part due to suffixes by $g_{suf}(e)$ and refer to it as the suffix-flow. By definition, $g(e) = g_{pre}(e) + g_{suf}(e)$.

The support of g_i is contained in the union of two consecutive slabs. Hence, the diameter of the support of g_i is bounded by $2d$. Hence $p_{\max}(g) \leq 2d$.

Clearly, $|g_i| = |f^*_i|$ and hence $|g| = |f^*|$. Set $\rho = c/(B + 2c)$. To complete the proof, it suffices to prove that $\rho \cdot g$ satisfies the capacity constraints. Indeed, for a "store" edge $e = (v, t) \to (v, t + 1)$, we have $g_{suf}(e) = 0$ and $g_{pre}(e) \leq f^*(e) \leq B$. For a "forward" edge $e = (v, t) \to (v + 1, t + 1)$ we have: $g_{pre}(e) \leq f^*(e) \leq c$. On the other hand, $g_{suf}(e) \leq B + c$. The reason is as follows. All the suffix-flow along e starts in the same boundary vertex (u, jd) below e. The amount of flow forwarded by (u, jd) is bounded by the amount of incoming flow, which is bounded by $B + c$. This completes the proof of the fractional case.

We now prove the integral case. The proof is a variation of the proof for the fractional case in which the supports of pre-flows and suffix-flows are disjoint. Namely, one alternates between slabs that support prefix-flow and slabs that support suffix-flow.

In the integral case, each accepted request r_i is allocated a single path p_i, and the allocated paths satisfy the capacity constraints. As in the fractional case, let $q_i \triangleq p'_i \circ q''_i$, where p'_i is the prefix of p_i till a boundary vertex (v, jd), and q''_i is a forward-only path. We need to prove that there exists a subset of at least $c/(2(B + c))$ of the paths $\{q_i\}_i$ that satisfy the capacity constraints. This subset is constructed in two steps.

First, partition the requests into "even" and "odd" requests according to the parity of the slab that contains their origin (a_i, t_i). (The parity of request r_i is simply the parity of $\lceil t_i/d \rceil$.) Pick a part that has at least half of the accepted requests in $F^*_{int}(R_d)$; assume w.l.o.g. that such a part is the part of the even slabs. Then, we only keep accepted requests whose origin belong to even slabs.

In the second step, we consider all boundary vertices $(v, j \cdot d)$. For each boundary vertex, we keep up to c paths that traverse it, and delete the remaining paths if such paths exist. In the second step, again, at least a $c/(B + c)$ fraction of the paths survive. It follows that altogether at least $c/(2(B + c))$ of the paths survive.

We claim that the remaining paths satisfy the capacity constraints. Note that prefixes are restricted to even slabs, and suffixes are restricted to odd slabs. Thus, intersections, if any, are between two prefixes or two suffixes. Prefixes satisfy the capacity constraints because they are prefixes of $F^*_{int}(R_d)$. Suffixes satisfy the capacity constraints because if two suffixes intersect, then they start in the same boundary vertex. However, at most c paths emanating from every boundary vertex survive. Hence, the surviving paths satisfy the capacity constraints, as required. This completes the proof of the lemma. ◀

We note that if the ratio B/c is bounded by a constant, then Lemma 3 guarantees an optimal solution which is only a (different) constant away from the unrestricted optimal solution.

4.2 The Algorithm for Short Packets

Short requests are further partitioned into four classes, defined as follows. Consider four tilings each with side length $k \triangleq 4\ell_S$ and horizontal and vertical shifts in $\varphi_x, \varphi_y \in \{0, k/2\}$.[3] The four possible shifts define four classes: The packets of a certain class (shift) are the packets whose source nodes reside in the SW quadrants of the tiles according to a given shift. Observe that each packet request belongs to exactly one class. We say that a path p_i from (a_i, t_i) to the row of b_i is *confined to a tile* if p_i is contained in one tile. We bound from above the path lengths by $2\ell_S$ so

We claim that by exhaustive search, it is possible to efficiently compute a maximum throughput solution for each class, under the restriction that each path is of length at most $2\ell_S$. The algorithm computes an optimal (bounded path length) solution for each class, and returns a highest throughput solution among the four solutions.

The polynomial running time of the exhaustive search algorithm per class is based on the two following observations.

▶ **Observation 4.** *A path of length at most $k/2 = 2\ell_S$ that begins in the SW quadrant of tile T is confined to T.*

Proof. The tile side length equals $k = 4\ell_S$. If the origin of a request is in the SW quadrant of a tile and the path length is at most $2\ell_S = k/2$, then the end of the path belongs to the same tile. ◀

▶ **Observation 5** ([6, 9, 8])**.** *Partition the packets according to their source node. For each node v, order the packets with source node v in increasing order of their target point. There exists an optimal solution that does not include any packet with rank more than 2 in that ordering.*[4]

▶ **Lemma 6.** *If $B = c = 1$, an optimal solution for each class in which paths are confined to their origin tile is computable in time polynomial in n and M.*

Proof. Fix a tile T. Let X denote the set of short requests in T, having rank at most 2 according to the ordering defined in Observation 5. By Observation 5, an optimal solution can be computed out of the set of packets X. Let Y be the set of paths in T. We perform an exhaustive search to find the optimal solution.

It suffices for the exhaustive search to consider all possibly partial functions $f : X \to Y$, and for each such function check that (1) each packet in the domain of f can be served by the path associated with it, and (2) no two path in the image of f intersect. Then pick, among all functions that pass the checks, the one with the largest domain.

The size of X is at most $2(k/2)^2$ since there are at most $(k/2)^2$ possible source nodes. The size of Y is at most $(k/2)^2\binom{2k}{k}$ (there are $(k/2)^2$ possible source nodes, and a path is defined by the steps in which it goes horizontally, and those where it goes vertically).

Therefore the number of possibly partial functions per tile is at most $2^{|X|} \cdot |Y|^{|X|} < 2^{2(k/2)^2} \cdot \left((k/2)^2 2^{2k}\right)^{2(k/2)^2} \leq 2^{poly(k)}$. Furthermore, given a function f the two needed checks

[3] Recall that $\ell_M \triangleq 3\ln n$ and $\ell_S \triangleq 3 \cdot \ln(\ell_M) = 3 \cdot \ln(3\ln n)$.
[4] Rank $B + c$ for non unit capacities.

can be done in time at most $O(|X| + |X|^2 \cdot k^2) = O(|X|^2 \cdot k^2)$. Since $k = O(\log \log n)$ for the case of the short packets, and $|X| \leq 2(k/2)^2$, the total time of the exhaustive search per tile is $poly(n)$.

The number of tiles that contain a request is bounded by the number of requests M. Hence, the running time of the algorithm for short requests is polynomial in n and M. ◀

▶ **Theorem 7.** *The approximation ratio of the algorithm for short requests is* $\frac{1}{16}$.

Proof. The short requests are partitioned to 4 classes. Then, for each class and tile, the exhaustive algorithm computes a solution which with cardinality at least a 1/4 of the optimal one, by the integral version of Lemma 3. ◀

5 Approximation Algorithm for Medium & Long Requests

We use the same algorithm for the two classes of medium and long requests, the only difference being some parameters of the algorithm. As indicated above, we consider at this point the case of unit capacities ($B = c = 1$). We further note that the approximation ratio of the algorithm for these classes is with respect to the optimal *fractional* solution.

Notation. Let $R_{d_{\min},d_{\max}}$ denote the set of packet requests whose source-to-destination distance is greater than $d_{\min}$ and at most $d_{\max}$. Formally, $R_{d_{\min},d_{\max}} \triangleq \{r_i \mid d_{\min} < b_i - a_i \leq d_{\max}\}$.

Parametrization. When applied to medium requests we use the parameter $d_{\max} = \ell_M$ and $d_{\min} = \ell_S$. When applied to long requests the parameters are $d_{\max} = n$ and $d_{\min} = \ell_M$. Note that these parameters satisfy $d_{\min} = 3 \cdot \ln d_{\max}$.

5.1 The Algorithm for $R_{d_{\min},d_{\max}}$

The algorithm for $R_{d_{\min},d_{\max}}$ proceeds as follows. To simplify notation, we abbreviate $R_{d_{\min},d_{\max}}$ by R. The parameters $d_{\min}$ and $d_{\max}$ must satisfy that $d_{\min} = 3 \cdot \ln d_{\max}$. We use the randomized rounding procedure by Raghavan [12, 13]. The description of this randomized rounding procedure is deferred to the full version.

1. Reduce the packet requests in R to path requests R^{st} over the space-time graph G^{st}.
2. Compute a maximum throughput fractional MCF $F \triangleq \{f_i\}_{r_i \in R^{st}}$ with edge capacities $\tilde{c}(e) = \lambda$ (for $\lambda = 1/(\beta(3) \cdot 6)$) [5] and bounded diameter $p_{\max}(F) \leq 2d_{\max}$. We remark that this MCF can be computed in time polynomial in n - the number of nodes and M - the number of requests.[6]

[5] The function $\beta : (-1, \infty) \to \mathbb{R}$ is defined by $\beta(\varepsilon) \triangleq (1 + \varepsilon) \ln(1 + \varepsilon) - \varepsilon$.

[6] Since always $d_{\max} \leq n$, we can consider a space-time grid of size at most $n \times (M \cdot 2n)$, which can be constructed by going over the release times of all M requests, eliminating "unnecessary" time steps. One can then compute a maximum throughput fractional solution with bounded diameter on this grid using linear programming. This is true because the constraint $p_{\max}(f_i) \leq d'$ is a linear constraint and can be imposed by a polynomial number of inequalities (i.e, polynomial in n and d'). For example, one can construct a product network with $(d' + 1)$ layers, and solve the MCF problem over this product graph.

3. Partition R to 4 classes $\{R^j\}_{j=1}^4$ according to the shift that results in the source node being in the SW quadrant of a $k \times k$ tiling, where $k \triangleq 2d_{\min} = 6 \ln d_{\max}$ (see Section 2.3). Pick a class R^j such that the throughput of F restricted to R^j is at least a quarter of the throughput of F, i.e., $|F(R^j)| \geq |F|/4$.
4. For each request $r_i \in R^j$, apply randomized rounding independently to f_i. The outcome of randomized rounding per request $r_i \in R^j$ is either "reject" or a path p_i in G^{st}. Let $R_{rnd} \subseteq R^j$ denote the subset of requests r_i that are assigned a path p_i by the randomized rounding procedure.
5. Let $R_{fltr} \subseteq R_{rnd}$ denote the requests that remain after applying filtering (described in Section 5.2).
6. Let $R_{quad} \subseteq R_{fltr}$ denote the requests for which routing in first quadrant is successful (as described in Section 5.3).
7. Complete the path of each request in R_{quad} by applying crossbar routing (as described in Section 5.4).

5.2 Filtering

Notation. Let e denote an edge in the space-time grid G^{st}. Let e^s denote an edge in the sketch graph (see Section 2.3). We view e^s also as the set of edges in G^{st} that cross the tile boundary that corresponds to the sketch graph edge e^s. The path p_i is a random variable that denotes the path, if any, that is chosen for request r_i by the randomized rounding procedure. For a path p and an edge e let $\mathbb{1}_p(e)$ denote the 0-1 indicator function that equals 1 iff $e \in p$.

The set of filtered requests R_{fltr} is defined as follows (recall that $\lambda = 1/(\beta(3) \cdot 6)$).

▶ **Definition 8.** A request $r_i \in R_{fltr}$ if and only if r_i is accepted by the randomized rounding procedure, and for every sketch-edge e^s in the sketch-path p_i^s it holds that $\sum_i \mathbb{1}_{p_i^s}(e^s) \leq 4\lambda \cdot k$.

▶ **Claim 9.** $\mathbf{E}\left[|R_{fltr}|\right] \geq \left(1 - O(\frac{1}{k})\right) \cdot \mathbf{E}\left[|R_{rnd}|\right]$.

Proof. We begin by bounding the probability that at least $4\lambda k$ sketch paths cross a single sketch edge.

▶ **Lemma 10** (Chernoff Bound). *For every edge e^s in the sketch graph,*[7]

$$\mathbf{Pr}\left[\sum_i \mathbb{1}_{p_i^s}(e^s) > 4\lambda k\right] \leq e^{-k/6}\,. \tag{1}$$

Proof of lemma. Recall that the edge capacities in the MCF F are λ. The capacity constraint $\sum_i f_i(e) \leq \lambda$ implies that $f_i(e) \leq \lambda$. Each sketch edge e^s corresponds to the grid edges between adjacent tiles. Since the demand of each request is 1, it follows that $f_i(e^s) \leq 1$.

For every edge e and request r_i, we have $\mathbf{E}\left[\mathbb{1}_{p_i^s}(e^s)\right] = \mathbf{Pr}\left[\mathbb{1}_{p_i^s}(e^s) = 1\right] = f_i(e^s) \leq 1$. Fix a sketch edge e^s. The random variables $\{\mathbb{1}_{p_i^s}(e^s)\}_i$ are independent 0-1 variables. Moreover, $\sum_i \mathbf{E}\left[\mathbb{1}_{p_i^s}(e^s)\right] = \sum_i f_i(e^s) = \sum_{e \in e^s} \sum_i f_i(e) \leq \lambda \cdot k$. By Chernoff bound[8]

$$\mathbf{Pr}\left[\sum_i \mathbb{1}_{p_i^s}(e^s) > 4 \cdot \sum_i \mathbf{E}\left[\mathbb{1}_{p_i^s}(e^s)\right]\right] < e^{-\beta(3)\cdot\lambda k} = e^{-k/6}. \qquad \blacktriangleleft$$

[7] The e in the RHS is the base of the natural logarithm.

[8] We use the following version of Chernoff Bound [12, 15]. Let $\{X_i\}_i$ denote a sequence of independent random variables attaining values in $[0, 1]$. Assume that $\mathbf{E}[X_i] \leq \mu_i$. Let $X \triangleq \sum_i X_i$ and $\mu \triangleq \sum_i \mu_i$. Then, for $\varepsilon > 0$, $\mathbf{Pr}[X \geq (1 + \varepsilon) \cdot \mu] \leq e^{-\beta(\varepsilon)\cdot\mu}$.

A request $r_i \in R_{rnd}$ is not in R_{fltr} iff at least one of the edges $e^s \in p_i^s$ has more than $2\lambda k$ paths on it. Hence, by a union bound,

$$\mathbf{Pr}\left[r_i \notin R_{fltr} \mid r_i \in R_{rnd}\right] \leq |p_i^s| \cdot e^{-k/6} \leq \left(\left\lceil \frac{2d_{\max}}{k} \right\rceil + 2\right) \cdot e^{-\ln d_{\max}} = O\left(\frac{1}{k}\right),$$

since $k = 6 \ln d_{\max}$. ◀

5.3 Routing in the First Quadrant

In this section, we deal with the problem of evicting as many requests as possible from their origin quadrant to the boundary of the origin quadrant.

▶ Remark. Because $k/2 \leq d_{\min}$ every request that starts in a SW quadrant of a tile must reach the boundary (i.e., top or right side) of the quadrant before it can reach its destination.

The maximum flow algorithm. Consider a tile T. Let X denote set of requests r_i whose source (a_i, t_i) is in the south-west quadrant of T. We say that a subset $X' \subseteq X$ is *quadrant feasible* (in short, feasible) if it satisfies the following condition: There exists a set of edge disjoint paths $\{q_i \mid r_i \in X'\}$, where each path q_i starts in the source (a_i, t_i) of r_i and ends in the top or right side of the SW quadrant of T.

We employ a maximum-flow algorithm to solve the following problem.
Input: A set of requests X whose source is in the SW quadrant of T.
Goal: Compute a maximum cardinality quadrant-feasible subset $X' \subseteq X$.

The algorithm is simply a maximum-flow algorithm over the following network, denoted by $N(X)$. Augment the quadrant with a super source $\tilde{s}$ and a super sink $\tilde{t}$. The super source $\tilde{s}$ is connected to every source (a_i, t_i) (of a request $r_i \in X$) with a unit capacity directed edge. (If γ requests share the same source, then the capacity of the edge is γ.) There is a unit capacity edge from every vertex in the top side and right side of the SW quadrant of T to the super sink $\tilde{t}$. All the grid edges are assigned unit capacities. Compute an integral maximum flow in the network. Decompose the flow to unit flow paths. These flow paths are the paths that are allocated to the requests in X'.

Analysis. Fix a tile T and let $R_T \subseteq R_{fltr}$ denote the set of requests in R_{fltr} whose source vertex is in the SW quadrant of T. Let $R'_T \subseteq R_T$ denote the quadrant-feasible subset of maximum cardinality computed by the max-flow algorithm. Let $R_{quad} = \bigcup_T R'_T$.

We now prove the following theorem that relates the *expected value* of $|R'_T|$ to the expected value of $|R_T|$. Observe that it is not always true that the same relation holds for any specific R_T that results from a specific random tape used by the randomized rounding procedure.

▶ **Theorem 11.** *[10, 11]* $\mathbf{E}_\tau\left[|R_{quad}|\right] \geq 0.93 \cdot \mathbf{E}_\tau\left[|R_{fltr}|\right]$*, where* τ *is the probability space induced by the randomized rounding procedure.*

Proof. By linearity of expectation, it suffices to prove that $\mathbf{E}_\tau\left[|R'_T|\right] \geq 0.93 \cdot \mathbf{E}_\tau\left[|R_T|\right]$, for any given tile T.

The proof goes along the following lines. We define a certain capacity constraint over rectangles; this definition makes use of the capacity of the boundary of the rectangles, and the number of requests within them. We define the set $\hat{R}_T \subseteq R_T$ to be a set of requests based on the capacity constraints of the rectangles containing the source of the requests. We prove that: (1) The set $\hat{R}_T$ thus defined is feasible, and (2) $\mathbf{E}_\tau\left[\left|\hat{R}_T\right|\right] \geq 0.93 \cdot \mathbf{E}_\tau\left[|R_T|\right]$. By

the algorithm, R'_T is of maximum cardinality (maximum flow), therefore, $|R'_T| \geq |\hat{R}_T|$, and the theorem follows.

We now describe how the feasible subset $\hat{R}_T$ is defined. Consider a subset S of the vertices in the SW quadrant of T. Let $\mathsf{dem}(S)$ denote the number of requests in R_T whose origin is in S. Let $\mathsf{cap}(S)$ denote the capacity of the edges in the network $N(X)$ that emanate from S. By the min-cut max-flow theorem, a set of requests $X \subseteq R_T$ is feasible if and only if $\mathsf{dem}(S) \leq \mathsf{cap}(S)$ for every cut $S \cup \{\tilde{s}\}$ in the network $N(X)$.

In fact, it is not necessary to consider all the cuts. It suffices to consider only axis parallel rectangles contained in the quadrant T. The reason is that without loss of generality, the set S is connected in the underlying undirected graph of the grid (i.e., consider each connected components of S separately). Every "connected" set S can be replaced by the smallest rectangle $Z(S)$ that contains S. We claim that $\mathsf{cap}(S) \geq \mathsf{cap}(Z(S))$ and $\mathsf{dem}(S) \leq \mathsf{dem}(Z(S))$. Indeed, there is an injection from the edges in the cut of $Z(S)$ to the edges in the cut of S. For example, a vertical edge e in the cut of $Z(S)$ is mapped to the topmost edge e' in the cut of S that is in the column of e. Hence, $\mathsf{cap}(Z(S)) \leq \mathsf{cap}(S)$. On the other hand, as $S \subseteq Z(S)$, it follows that $\mathsf{dem}(S) \leq \mathsf{dem}(Z(S))$. Hence if $\mathsf{dem}(S) > \mathsf{cap}(S)$, then $\mathsf{dem}(Z(S)) > \mathsf{cap}(Z(S))$.

We say that a rectangle Z is *overloaded* if $\mathsf{dem}(Z) > \mathsf{cap}(Z)$. The set $\hat{R}_T$ is defined to be the set of requests $r_i \in R_T$ such that the source of r_i is not included in an overloaded rectangle. Namely, $\hat{R}_T \triangleq \{r_i \in R_T \mid \forall \text{ rectangles } Z : Z \text{ is overloaded } \Rightarrow (a_i, t_i) \notin Z\}$. Consider an $x \times y$ rectangle Z. We wish to bound from above the probability that $\mathsf{dem}(Z) > \mathsf{cap}(Z)$. Note that $\mathsf{cap}(Z) = x + y$. Since requests in R_T that start in Z must exit the quadrant, it follows that $\mathsf{dem}(Z)$ is bounded by the number of paths in R_T that cross the top or right side of Z (note that there might be additional paths that do not start in Z but cross Z.). The amount of flow that emanates from Z is bounded by $\lambda \cdot (x+y)$ (the initial capacities are λ and there are $x+y$ edges in the cut). By the randomized rounding procedure, $\mathbf{Pr}[e \in p_i] = f_i(e)$. Summing up over all the edges in the cut of Z and the requests in R_T, the expected number of paths in R_T that cross the cut of Z equals the flow of the request in R_T, which, in turn, is bounded by the capacity $\lambda \cdot (x + y)$. As the paths of the requests are independent random variables, we obtain:[9] $\mathbf{Pr}[\mathsf{dem}(Z) > \mathsf{cap}(Z)] \leq \mathbf{Pr}\left[\sum_{i \in R_T} |p_i \cap cut(Z)| > (x + y)\right] \leq (\lambda \cdot e)^{x+y}$.

For each x, y, each source (a_i, t_i) is contained in at most $x \cdot y$ rectangles with side lengths $x \times y$. By applying a union bound, the probability that (a_i, t_i) is contained in an overloaded rectangle is bounded from above by

$$\begin{aligned}\mathbf{Pr}[\exists \text{ overloaded rectangle } Z : (a_i, t_i) \in Z] &\leq \sum_{x=1}^{\infty}\sum_{y=1}^{\infty} xy \cdot (\lambda \cdot e)^{x+y} \\ &\leq \frac{(\lambda \cdot e)^2}{(1 - \lambda \cdot e)^4} \leq 0.07, \end{aligned} \tag{2}$$

and the theorem follows. ◀

Routing within the first tile (see Section 5.4) requires however a stronger upper bound on the number of requests that emanate from each side of the quadrant, namely that at most $k/3$ paths reach each side of the quadrant. Using a simple procedure (e.g., taking the solution and greedily eliminating paths) one can have a solution for which this condition holds, and with cardinality only a constant fraction smaller.

[9] Using the following version of the Chernoff bound: $\mathbf{Pr}[X \geq \alpha \cdot \mu] \leq \left(\frac{e}{\alpha}\right)^{\alpha \cdot \mu}$.

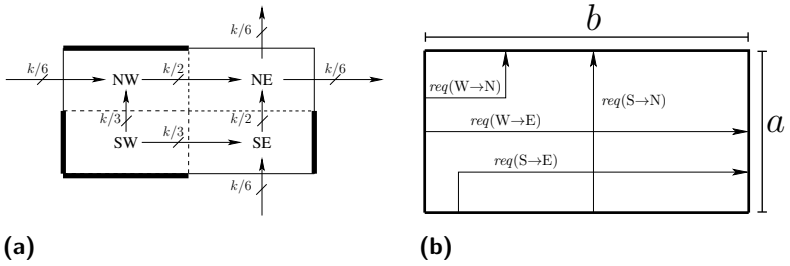

Figure 1 (a) Partitioning of a tile to quadrants [9]. Thick lines represent "walls" that cannot be crosses by paths. Sources may reside only in the SW quadrant of a tile. Maximum flow amounts crossing quadrant sides appears next to each side. Final destinations of paths are assumed (pessimistically) to be in the top row of the NE quadrant. (b) Crossbar routing: flow crossing an $a \times b$ grid [9].

▶ **Corollary 12.** *Let R'_{quad} be the set of quadrant-feasible paths such that at most $k/3$ paths reach each side of each quadrant. Then, $\mathbf{E}_\tau\left[|R'_{quad}|\right] \geq \Omega(1) \cdot \mathbf{E}_\tau\left[|R_{fltr}|\right]$, where τ is the probability space induced by the randomized rounding procedure.*

5.4 Detailed Routing

In this section we deal with computing paths for requests $r_i \in R_{quad}$ starting from the boundary of the SW quadrant that contains the source (a_i, t_i) till the destination row b_i. These paths are concatenated to the paths computed in the first quadrant to obtain the *final* paths of the accepted requests. Detailed routing is based on the following components: (1) The projections of the final path and the path p_i to the sketch graph must coincide. (2) Each tile is partitioned to quadrants and routing rules within a tile are defined. (3) Crossbar routing within each quadrant is applied to determine the final paths (except for routing in SW quadrants in which paths are already assigned).

Sketch paths and routing between tiles. Each path p_i computed by the randomized rounding procedure is projected to a sketch path p_i^s in the sketch graph. The final path $\hat{p}_i$ assigned to request r_i traverses the same sequence of tiles, namely, the projection of $\hat{p}_i$ is also p_i^s.

Routing rules within a tile [6]. Each tile is partitioned to quadrants as depicted in Figure 1a. The bold sides (i.e., "walls") of the quadrants indicate that final paths may not cross these walls. The classification of the requests ensures that source vertices of requests reside only in SW quadrants of tiles. Final paths may not enter the SW quadrants; they may only emanate from them. If the endpoint of a sketch path p_i^s ends in tile T, then the path $\hat{p}_i$ must reach a copy of its destination b_i in T. Reaching the destination is guaranteed by having $\hat{p}_i$ reach the top row of the NE quadrant of T (and thus it must reach the row of b_i along the way).

Crossbar routing. [9]. Routing in each quadrant is simply an instance of routing in a (uni-directional) 2D grid where requests enter from two adjacent sides and exit from the opposite sides. Figure 1b depicts such an instance in which requests arrive from the left and bottom sides and exit from the top and right side. The following claim characterizes when crossbar routing succeeds.

▶ **Claim 13** ([9]). *Consider a 2-dimensional directed $a \times b$ grid. A set of requests can be routed from the bottom and left boundaries of the grid to the opposite boundaries if and only if the number of requests that should exit each side is at most the length of the corresponding side.*

We conclude with the following claim.

▶ **Claim 14.** *Detailed routing succeeds in routing all the requests in R_{quad}.*

Proof sketch. The sketch graph is a directed acyclic graph. Sort the tiles in topological ordering. Within each tile, order the quadrants also in topological order: SW, NW, SE, NE. Prove by induction on the position of the quadrant in the topological ordering that detailed routing up to and including the quadrant succeeds. The claim for all SW quadrants follows because this routing is done along the path that result of the randomized rounding step of the requests in R_{quad}. Now note that filtering ensures that the number of paths between tiles is at most $2\lambda k < k/6$. Routing in the first quadrant ensures that the number of paths emanating from each side of a SW quadrant is at most $k/3$. The induction step follows by applying Claim 13. ◀

5.5 Approximation Ratio

▶ **Theorem 15.** *The approximation ratio of the algorithm for packet requests in $R_{d_{\min},d_{\max}}$, for $d_{\min} = 3 \cdot \ln d_{\max}$, is constant in expectation.*

Proof. We follow the algorithm, as defined in Section 5.1, stage by stage.

Stage 2 computes a fractional maximum multi-commodity flow on a network with reduced edge capacities, and with the requirement that all flows have bounded diameter. By Lemma 3, bounding path lengths in the MCF results in a solution of at least a 1/3 fraction of the unrestricted one, and the scaling of the capacities in the space-time grid results in a solution which is at least a $\lambda = \Omega(1)$ fraction of the latter.

Stage 3 classifies the requests into 4 classes and picks only the one for which the multi-commodity flow solution is the highest, hence resulting in a solution of at least a 1/4 fraction of the solution of the previous stage.

Stage 4 applies a randomized rounding procedure to the flows that are picked in stage 3. The expected size of the solution is equal to the total flow left from the previous stage (but the solution might not be feasible).

Stage 5 applies a filtering procedure to the solution of the previous stage, in order to get a feasible solution on the sketch graph. By Claim 9, the expected size of the feasible solution is at least a $1 - O(1/k)$ fraction of the solution given by stage 4. Observe that $1 - O(1/k) = O(1)$ (in fact $k = \Omega(\log \log n)$ in any relevant invocation of the algorithm).

Stage 6 further reduces the size of the solution when the algorithm selects a subset of the requests that have survived so far, using a maximum flow algorithm applied to each SW quadrant. This is done in order to allow for the solution to be feasible in the original space-time grid. By Corollary 12 the expected size of the solution after this stage is an $\Omega(1)$ fraction of the expected size before this stage.

Stage 7 gives the final routing without further reducing the size of the solution.

We conclude that the algorithm for medium and long requests is a randomized $O(1)$ approximation algorithm (in fact with respect to the fractional optimum). ◀

6 Extensions

6.1 Non-unit Capacities & Buffer Sizes

Our results extend to arbitrary values of B and c, with an (additional) multiplicative penalty in the approximation ratio of $O(B/c)$ in certain cases. In this section we outline the required changes in the algorithm when B/c does not depend on the input size, i.e., $B = \Theta(c)$. For this case, our results will give a randomized approximation algorithm with constant approximation ratio. We now outline the required modifications to handle this case, explaining the modifications in each component of the algorithm, and then the overall structure of the modified algorithm.

Adapting the algorithm for short packets (the exhaustive search algorithm). Exhaustive search for arbitrary B and c can be done in polynomial time provided that the distance of each packet request is at most $\ln(\ell_S)$ (see [11, Lemma 7]). This means that for general B and c there is an additional category of requests, called *very short* requests, on which the exhaustive search will be applied. The remaining packets are divided into short, medium, and long requests, as for the case of $B = c = 1$, and the distance of short requests is lower bounded by $\ln(\ell_S)$.

Adapting the Algorithm for $R_{d_{\min},d_{\max}}$ (for short, medium, and long packets). We adapt the algorithm to the case where $B = c = \gamma$, where $\gamma \in \mathbb{N}^{>0}$. Note that while we consider here non-unit capacities, the flow demands (packets) are not changed, i.e., they remain unit demands. Consider a 3-dimensional view of the grid where there are γ "floors", each floor for a single capacity slice out of γ. This view of the grid partitions the routing problem at hand to γ problems, where at each of them the capacities are unit. Now, apply the algorithm for $R_{d_{\min},d_{\max}}$ on each of these floors. The approximation ratio of the revised algorithm follows from linearity of expectation.

Putting things together. We now describe the general structure of the algorithm for arbitrary B and c. The packets are partitioned into four classes *very short, short, medium,* and *long*. As given above, we have a constant approximation algorithm for very short packets for arbitrary B and c.

For the other three categories, we set both the buffer sizes and link capacities to $\min\{B, c\}$, and apply the (modified) $R_{d_{\min},d_{\max}}$ for the case $B = c$. Observe that the *fractional* optimum incurs a penalty of at most a factor of $O(B/c)$ as a result of this capacity change. Since this algorithm gives a constant approximation compared to the *fractional* optimum, we get an $O(B/c)$-approximation algorithm compared to the optimum on the network with non-modified capacities.

We can now conclude with the following theorem.

▶ **Theorem 16.** *There exists a randomized algorithm for the Max-Pkt-Line problem, such that if $B = O(c)$, then its approximation ratio is a (different) constant.*

6.2 Supporting Soft Deadlines

In this section we argue that if packets have deadlines, we achieve a constant approximation ratio when the produced solution is allowed to miss deadlines by at most $O(\log n)$ time units. This is achieved, simply, by reducing a request with deadlines to a path request which

its destination set is a copy of the destination vertex with a time index which is less than the time of the deadline. Since the tiling has "resolution" of at most $O(\log n)$, the detailed routing might "miss" the deadline by the tile's side length.

Note that the algorithm for short requests (or very short for non-unit capacities) can handle requests with deadlines, and achieve the same performance while respecting hard deadlines, because it employs exhaustive search[10].

References

1 Micah Adler, Arnold L. Rosenberg, Ramesh K. Sitaraman, and Walter Unger. Scheduling time-constrained communication in linear networks. *Theory Comput. Syst.*, 35(6):599–623, 2002. doi:10.1007/s00224-002-1001-6.

2 William Aiello, Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Dynamic routing on networks with fixed-size buffers. In *SODA*, pages 771–780, 2003. doi:10.1145/644108.644236.

3 Stanislav Angelov, Sanjeev Khanna, and Keshav Kunal. The network as a storage device: Dynamic routing with bounded buffers. *Algorithmica*, 55(1):71–94, 2009. (Appeared in APPROX-05). doi:10.1007/s00453-007-9143-1.

4 Baruch Awerbuch, Yossi Azar, and Amos Fiat. Packet routing via min-cost circuit routing. In *ISTCS*, pages 37–42, 1996.

5 Yossi Azar and Rafi Zachut. Packet routing and information gathering in lines, rings and trees. In *ESA*, pages 484–495, 2005. (See also manuscript in http://www.cs.tau.ac.il/~azar/). doi:10.1007/11561071_44.

6 Guy Even and Moti Medina. An $O(\log n)$-Competitive Online Centralized Randomized Packet-Routing Algorithm for Lines. In *ICALP (2)*, pages 139–150, 2010. doi:10.1007/978-3-642-14162-1_12.

7 Guy Even and Moti Medina. Online packet-routing in grids with bounded buffers. In *Proc. 23rd Ann. ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 215–224, 2011. doi:10.1145/1989493.1989525.

8 Guy Even and Moti Medina. Online packet-routing in grids with bounded buffers. *CoRR*, abs/1407.4498, 2014.

9 Guy Even, Moti Medina, and Boaz Patt-Shamir. Better deterministic online packet routing on grids. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 284–293, 2015. doi:10.1145/2755573.2755588.

10 Jon M Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Massachusetts Institute of Technology, 1996.

11 Harald Räcke and Adi Rosén. Approximation algorithms for time-constrained scheduling on line networks. In *SPAA*, pages 337–346, 2009. doi:10.1145/1583991.1584071.

12 Prabhakar Raghavan. Randomized rounding and discrete ham-sandwich theorems: provably good algorithms for routing and packing problems. In *Report UCB/CSD 87/312*. Computer Science Division, University of California Berkeley, 1986.

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

14 Adi Rosén and Gabriel Scalosub. Rate vs. buffer size-greedy information gathering on the line. *ACM Transactions on Algorithms*, 7(3):32, 2011. doi:10.1145/1978782.1978787.

15 Neal E Young. Randomized rounding without solving the linear program. In *SODA*, volume 95, pages 170–178, 1995.

[10] Apply the exhaustive search outlined in [11, Lemma 7] for this case.