# Near-Shortest Path Routing in Hybrid Communication Networks

**Sam Coy**
University of Warwick, Coventry, UK

**Artur Czumaj**
University of Warwick, Coventry, UK

**Michael Feldmann**
Paderborn University, Germany

**Kristian Hinnenthal**
Paderborn University, Germany

**Fabian Kuhn**
University of Freiburg, Germany

**Christian Scheideler**
Paderborn University, Germany

**Philipp Schneider**
University of Freiburg, Germany

**Martijn Struijs**
TU Eindhoven, The Netherlands

## Abstract

Hybrid networks, i.e., networks that leverage different means of communication, become ever more widespread. To allow theoretical study of such networks, [Augustine et al., SODA'20] introduced the HYBRID model, which is based on the concept of synchronous message passing and uses two fundamentally different principles of communication: a *local* mode, which allows every node to exchange one message per round with each neighbor in a *local communication graph*; and a *global* mode where *any pair* of nodes can exchange messages, but only *few such exchanges* can take place per round. A sizable portion of the previous research for the HYBRID model revolves around basic communication primitives and computing distances or shortest paths in networks. In this paper, we extend this study to a related fundamental problem of *computing compact routing schemes for near-shortest paths* in the local communication graph. We demonstrate that, for the case where the local communication graph is a *unit-disc graph* with $n$ nodes that is realized in the plane and has *no radio holes*, we can deterministically compute a routing scheme that has constant stretch and uses labels and local routing tables of size $O(\log n)$ bits in only $O(\log n)$ rounds.

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms

**Keywords and phrases** Hybrid networks, overlay networks

**Digital Object Identifier** 10.4230/LIPIcs.OPODIS.2021.11

**Related Version** *Full Version*: https://arxiv.org/abs/2202.08008

**Funding** *Sam Coy*: Supported by the Centre for Discrete Mathematics and its Applications (DIMAP) and by an EPSRC studentship.
*Artur Czumaj*: Supported by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/1, and by an IBM Award.
*Christian Scheideler*: Supported by the German Research Foundation (DFG) within the Collaborative Research Center 901 "On-The-Fly Computing" under the project number 160364472-SFB901.

## 1 Introduction

Humans naturally communicate in a hybrid fashion by making use of broadcast services, emails, phones, or simply face-to-face communication. Thus, it seems natural to study hybrid communication also in distributed systems. But fundamental research in this area is still in its infancy, even though there are several examples where hybrid communication is already exploited in practice. For instance, in modern data centers, wired communication networks are combined with high-speed wireless communication to reduce wire length or increase bandwidth without adding congestion to the wired network [12]. This paper focuses on hybrid *wireless* networks: networks that combine ad-hoc, WLAN-based connections (the



*local* network) with connections via a cellular or satellite infrastructure (the *global* network). These can be realized, for instance, by smartphones, since they support both communication modes and solutions for smartphone ad hoc networks have been around for almost a decade. Connections in the local network can transfer large amounts of data cheaply, but have limited range, while global connections can transmit data between any pair of devices, but typically with bandwidth restrictions and additional costs. So ideally, global communication should be reserved for exchanging control messages while the data should be sent via the local edges, which *necessitates the computation of a routing scheme for the local network*.

The simplest solution to compute a routing scheme would be to use the global mode to collect all local device connections and/or positions in a centralized server and do the computation there. However, a centralized solution would represent a bottleneck and single point of failure, or it would not be for free when making use of a cloud service. We avoid these problems by only relying on the devices themselves. Interestingly, even without any central service, we vastly improve the results over what is possible with just the local network. More specifically, we demonstrate that with a hybrid wireless network one can significantly speed up the computation of compact routing schemes under certain natural circumstances, thereby opening up a new research direction for wireless networks.

## 1.1 Model and Problem Definition

We assume a set $V$ of $n$ nodes with unique IDs. Each node is associated with a fixed, distinct point in the 2-dimensional Euclidean plane, (i.e., $V \subseteq \mathbb{R}^2$), and every node $v \in V$ knows the global coordinates of its point. We assume the standard *synchronous message passing model*: time proceeds in synchronous time slots called *rounds*. In each round, every node can perform an arbitrary amount of local computation and then communicate with other nodes.

In the HYBRID model, communication occurs in one of two modes: the *local mode* and the *global mode*. The connections for the local mode are given by a fixed graph. In our case, this graph is represented by a *unit-disk graph* $\mathsf{UDG}(V)$: for any $v, w \in V$, $\{v, w\} \in \mathsf{UDG}(V)$ if and only if $v$ and $w$ are at distance at most 1. For the local mode, we use the CONGEST model for simplicity: in each round, for all edges $\{v, w\} \in \mathsf{UDG}(V)$, node $v$ can send a message of $O(\log n)$ bits to node $w$. However, our algorithms still work if instead the more restrictive (and more natural) Broadcast-CONGEST model is used (see the full version). We assume that each message can carry a constant number of node locations (this is analogous to the *Real RAM model*, a standard model of sequential computation).

For the global mode, we are using a variant of the *node-capacitated clique (*NCC*)* model called $\mathsf{NCC}_0$ [2] that captures key aspects of overlay networks. In this model, any node $u$ can send messages to any other node $v \in V$ provided $u$ knows $v$'s ID. Initially, the set of IDs known to each node is just limited to its neighbors in $\mathsf{UDG}(V)$. Each node is limited to sending $O(\log n)$ messages of $O(\log n)$ bits via the global mode in each round. W.l.o.g., we assume that whenever a node $v$ knows the ID of some node $w$, it also knows $w$'s location (since this can be sent together with its ID).

**Model Motivation.** The assumption that the nodes know their global coordinates is motivated by the fact that smartphones can nowadays accurately determine their location using GPS or wireless access point or base station information. However, it would also be sufficient if the nodes can determine the distance and relative angles to their neighbors in the UDG (this can be obtained via kown localization methods [18]), though with some precision loss.

The use of the $\mathsf{NCC}_0$ model in the global communication mode is motivated by the fact that nodes can communicate with any other node in the world via the cellular infrastructure given its ID (e.g., its phone number). Note that $\mathsf{NCC}_0$ is weaker than $\mathsf{NCC}$, which assumes a clique from the start, but it is known that once the right topology has been set up in $\mathsf{NCC}_0$ (which can be done in $O(\log n)$ rounds [16]), any communication round in $\mathsf{NCC}$ can be simulated by $O(\log n)$ communication rounds in $\mathsf{NCC}_0$ [25].

**Problem Definition.** Our goal is to compute a *compact routing scheme* for $\mathsf{UDG}(V)$, in hybrid networks where $\mathsf{UDG}(V)$ is connected and does not contain radio holes. $\mathsf{UDG}(V)$ is said to contain a *radio hole* if, roughly speaking, there is an internal cycle in $\mathsf{UDG}(V)$ that cannot be triangulated. A precise definition will be given in Section 2.

Let $\mathcal{G}$ be a class of graphs. A stateless[1] routing scheme $\mathcal{R}$ for $\mathcal{G}$ is a family of labeling functions $\ell_G : V(G) \to \{0,1\}^+$ for each $G \in \mathcal{G}$, which assigns a bit string to every node $v$ in $G$. The label $\ell_G(v)$ serves as the address of $v$ for routing in $G$: it contains the identifier of $v$, and may also contain information about the topology of $G$.

While the identifier is given as part of the input, the label is determined in the *preprocessing*. Additionally, the preprocessing has to set up a routing function $\rho_G : V(G) \times \{0,1\}^+ \to V(G)$ for the given graph $G$ that, given the current node of a message and the label of the destination, determines the neighbor of $v$ in $G$ to forward the message to[2]. A routing scheme must satisfy various properties.

First of all, it must be *correct*, i.e., for every source-destination pair $(s,t)$, $\rho_G$ determines a path in $G$ leading from $s$ to $t$. Second, it must be *local*, in a sense that every node $v$ can evaluate $\rho_G(v,\ell)$ locally. Third, the routing should be *efficient*, i.e., the ratio of the length of the routing path and the shortest path – also known as the *stretch factor* – should be as close to 1 as possible. In our case, the length of a routing path is simply determined by the number of edges used by it. Note that whenever we have a constant stretch w.r.t. the number of edges in $\mathsf{UDG}(V)$, we also have a constant stretch w.r.t. the sum of the Euclidean lengths of its edges, so we achieve a constant stretch for *both* types of metrics (see Section 2). Finally, the routing scheme should be *compact*, i.e., the labels $\ell_G(v)$ of the nodes $v$ and the amount of space needed at each node $v$ to evaluate $\rho_G(v,\ell)$ should be as small as possible.

**Problem Motivation.** There are various reasons for developing fast distributed algorithms for compact routing schemes in hybrid wireless networks. First of all, computing routing schemes for the local ad-hoc network is useful even in the presence of a cellular infrastructure since ad-hoc connections are comparatively cheap to use and typically offer a much larger bandwidth. Also, the ability to quickly compute compact routing schemes allows for frequent adaption in case of topological changes in the wireless ad-hoc network with low overhead.

## 1.2 Our Contributions

In Appendix A we show that it is impossible to set up a compact routing scheme with constant stretch in time $o(\sqrt{n})$ when just relying on the UDG for communication even if the geometric location of all nodes is known and the UDG is hole-free. This poses the question of whether limited global communication can overcome this. We answer this question by

[1] In a stateless routing scheme a packet can not accumulate information along the routing path and is thus oblivious to the routing path that the packet took so far (as opposed to stateful routing).

[2] More general definitions of routing functions exist, but we do not require the additional power afforded by stateful routing (for instance), to compute near-constant routing schemes in logarithmic time.

showing the following result, which demonstrates the impact that a modest amount of global communication has when applied to problems which are challenging to solve locally.

▶ **Theorem 1.** *For a* HYBRID *network with a hole-free* UDG($V$)*, a compact, stateless routing scheme can be deterministically computed for* UDG($V$) *in* $O(\log n)$ *rounds. The scheme uses node labels of* $O(\log n)$ *bits and a mapping* $\rho$ *that (i) can be evaluated locally with* $O(\log n)$ *bits of information in each node and (ii) such that for every source-destination pair* $s, t \in V$*,* $\rho$ *determines a routing path of constant stretch from* $s$ *to* $t$ *in* UDG($V$)*.*

Technical novelties of this work include a *grid graph abstraction* of any UDG which serves to sparsify the UDG while preserving its geometric structure. Computations on the grid graph can be simulated efficiently in UDG($V$). Furthermore, we can transform a routing scheme on the grid graph to one in the UDG, increasing the stretch only by a constant.

We show how to construct this abstraction in a distributed setting based entirely on local communication. This could potentially make it of interest when studying routing or distance approximation problems on UDGs in the CONGEST or Broadcast-CONGEST models or for simplification of existing algorithms. We also believe that the grid graph abstraction and its properties will be useful for future work in the HYBRID setting, making it a springboard for the case of UDGs *with* radio-holes.

## 1.3 Overview

The first step is the computation of a simple, yet surprisingly useful abstract graph structure on UDG($V$), which we call a *grid graph* $\Gamma$. The vertices of $\Gamma$ are the centers of the cells of a regular square grid which intersect with an edge of UDG($V$). Two vertices of $\Gamma$ share an edge iff their cells are vertically or horizontally adjacent (see Section 2.2, Figure 1). Subsequently, in Section 2.3, we tie the graphs UDG($V$) and $\Gamma$ together by defining a *representative* in $V$ for each vertex of $\Gamma$ that fulfills two main properties. First, two representatives of *adjacent* grid vertices are connected with a path of at most 3 hops in UDG($V$) (see Figure 2). Second, each node in $V$ has such a representative within 1 hop in UDG($V$).

We then turn to the algorithmic aspects of $\Gamma$. In Section 2.4 we define the *representation* $R$ of $\Gamma$, where grid vertices correspond to their aforementioned representatives and grid edges correspond to paths of 3 hops in UDG($V$), and we show that $R$ can be efficiently computed in UDG($V$). Furthermore, the representation $R$ can be used to efficiently *simulate* the HYBRID model on $\Gamma$, which is summarized in Theorem 8. In Section 3, we show that an optimal path in $\Gamma$ implies a path in $R$ with a constant approximation ratio (Theorem 10).

The final step of the first part is to construct a constant stretch routing scheme $\mathcal{R}$ for UDG($V$) assuming that we have an optimal one for $\Gamma$ (Section 3.2), which is encapsulated by Theorem 16. Since we can efficiently simulate the HYBRID model on $\Gamma$ (Theorem 8), the second part can be considered in isolation from the first part. Note that so far we did not exploit the fact that UDG($V$) is hole-free. In fact, the construction, simulation, and properties of $\Gamma$ hold without that assumption, which is only needed for the second part.

Requiring the UDG to be hole-free is a strong assumption. However, we believe that at least bounding the number of holes is necessary in order to compute a compact, constant-stretch labelling scheme in $O(\log n)$ rounds. Doing this in time and space polylogarithmic in $n$ that also *scales well* in the number of radio holes in the UDG seems to be highly non-trivial, as these holes may intertwine in arbitrary ways, while there are exponentially many

possibilities of navigating around them.[3] While in our setting there still can be exponentially many simple paths between two points, we are able to exploit the lack of large holes between them to deal with arbitrarily complex boundaries of UDGs in a hybrid network setting.

To compute the routing scheme $\mathcal{R}_\Gamma$ on $\Gamma$, the first step (Section 4) is to arrange the grid nodes into maximal vertical lines, called *portals* (see Figure 3a). All portals with two horizontally adjacent nodes will add one such edge, resulting in a *portal-tree* $T_\Gamma$, which is cycle-free because $\mathsf{UDG}(V)$ is hole-free (see Figure 3b). In order to compute a labelling scheme we first perform a distributed depth-first traversal on $T_\Gamma$ (where the root is the node with min ID). This allows us to compute intervals $I_v$ for each node $v$ of $T_\Gamma$ that fulfill the parenthesis theorem: it is $I_w \supset I_v$ ($I_w \subset I_v$) for each ancestor (descendant) node $w$ of $v$ in $T_\Gamma$, or else $I_w \cap I_v = \emptyset$ when $v, w$ are in different branches of $T_\Gamma$ (see Figure 3d). Then all nodes of a portal will agree on interval $I_r$ of node $r$ that is closest to the root as their *portal label*. The challenge here is to carefully line up techniques for the more restrictive $\mathsf{NCC}_0$ model to obtain such a labelling in $O(\log n)$ rounds.

Finally, in Section 5, we use $T_\Gamma$ to route a packet from source $s$ to target node $t$ in $T_\Gamma$. Since the shortest path in $T_\Gamma$ may not necessarily follow the tree, we have to define a routing strategy that jumps over branches when needed, for which we can use the "tree information" encoded in the labels. We use the portal labels to prioritize jumping horizontally as soon as the next portal on a path is reachable via any edge in $\Gamma$. Vertical routing within portals is done as a second priority for which node labels $I_v$ are used. We prove that this strategy yields an exact routing scheme $\mathcal{R}_\Gamma$ for $\Gamma$ formalized in Theorem 23. Consequently, Theorem 1 is a corollary from the fact that we can emulate $\Gamma$ on $\mathsf{UDG}(V)$ (Theorem 8) and that $\mathcal{R}_\Gamma$ can be transformed into a constant stretch routing scheme $\mathcal{R}$ for $\mathsf{UDG}(V)$ (Theorem 16).

## 1.4 Related Work

An early effort to formalize hybrid communication networks by [1], combined the $\mathsf{LOCAL}$ model with a global communication mode that essentially allows a single node to broadcast a message to all others per round. Note that this conception of the global network is fundamentally different to ours, which manifests in the fact that solving a aggregations problem (e.g., computing the sum of inputs of each node) can take $\Omega(n)$ rounds (by contrast, it takes $O(\log n)$ rounds in the $\mathsf{NCC}$ model).

Recently, shortest path problems in general **hybrid networks** have been studied by various authors [3, 9, 21, 11], which provide approximate and exact solutions for the all-pairs shortest paths problem (APSP) and the single-source shortest paths problem (SSSP). These solutions all require $O(n^\varepsilon)$ rounds (for constant $\varepsilon > 0$) to achieve a constant approximation ratio, and this is tight in the case of APSP. $O(\log n)$-time algorithms to solve SSSP for some classes of sparse graphs (not including UDGs) are given in [11]. Shortest path problems have also been studied for hybrid wireless networks [8]. They show that for a bounded-degree $\mathsf{UDG}(V)$ with a convex outer boundary, where the bounding boxes of the radio holes do not overlap, one can compute an abstraction of $\mathsf{UDG}(V)$ in $O(\log^2 n)$ time so that paths of constant stretch between all source-destination pairs *outside of* the bounding boxes can be found (a simple extension of their approach to outer boundaries of *arbitrary* shape seems unlikely).

Numerous **online routing strategies** have been proposed for general UDGs, including FACE-I, FACE-II, AFR, OAFR, GOAFR and GOAFR+ [5, 24, 22, 23]. In [24, 22] it is proven that GOAFR and GOAFR+ are asymptotically optimal w.r.t. path length compared

[3] The number of simple *st*-paths that cannot be continuously deformed into each other without crossing a hole (i.e., non-homotopic paths) is $2^h$, where $h$ is the number of radio holes.

to any geometric routing strategy. However, the achieved stretch is linear in the length of a shortest path. When a UDG contains the Delaunay graph of its nodes, one can exploit the fact that the Delaunay graph is a 2-spanner of the Euclidean metric [29], and MixedChordArc has been shown to be a constant-competitive routing strategy for Delaunay graphs [4]. This is only applicable in UDGs where the line segment connecting two nodes of the UDG does not intersect a boundary, which is the case if it has a convex outer boundary *and* is hole-free.

**Centralized constructions**[4] for compact routing schemes have been heavily investigated for general graphs (see, e.g., [28]) as well as UDGs. Here, we just focus on UDGs. Bruck et al. [6] present a medial axis based naming and routing protocol that does not require geographical locations, makes routing decisions locally, and achieves good load balancing. The routing paths seem near-optimal in simulations, but no rigorous results are given. Gao and Goswami [13] propose a routing algorithm that achieves a constant approximation ratio for load balanced routing in a UDG of arbitrary shape, but the question of near-optimal routing paths is not addressed. Based on work by Gupta et al. [17] for planar graphs, Yan et al. [30] show how to assign a label of $O(\log^2 n)$ bits to each node of the graph such that given the labels of a source $s$ and of a target $t$, one can locally compute a path from $s$ to $t$ with constant stretch. Using the well-separated pair decomposition (WSPD) for UDGs [14], Kaplan et al. [19] present a local routing scheme with stretch $1+\varepsilon$ with node labels, routing tables and headers of size polynomial in $\log n, \log D$, and $1/\varepsilon$, where $D$ is the diameter of $\mathsf{UDG}(V)$. Later, [26] shows how to achieve a stretch of $1+\varepsilon$ without using dynamic headers.

Our routing scheme for the grid graph abstraction extends the routing scheme proposed by Santoro and Khatib [27], who presented a labelling along with an optimal routing scheme for trees by computing a minimum-distance spanning tree and labelling of that tree via a depth-first search.[5] In our scheme, we provide optimal paths between any source-target pair in the grid graph, because we allow using edges that are not part of the spanning tree for routing in order to jump between the branches of the spanning tree.

Our study is also related to routing problems in sparse graphs in **parallel models** [20, 10]. For example, the algorithm of Kavvadias et al. [20] can be used to compute routing tables in planar graphs in time $\widetilde{O}(1)$ and work $\widetilde{O}(n)$. Together with the simulation framework of Feldmann et al. [11], the algorithm could in principle be used to solve our problem. However, for the simulation to work, one would need to construct a suitable global network, sparsify the graph, and, together with the simulation overhead, one would obtain a polylogarithmic runtime much higher than $O(\log n)$. Further, the size of the routing tables may be $\Theta(n^2)$.

## 2 Grid Graph

Let $G := \mathsf{UDG}(V)$. The goal of this section is to construct a grid abstraction of $G$ which makes finding routing protocols in the subsequent section manageable. In particular (but still suppressing some details), we want to simulate a bounded degree *grid graph* on $G$ such that shortest paths in the grid graph represents only a constant factor detour in $G$. The way we obtain such a grid representation of $G$ in a distributed fashion is by simulating grid nodes

[4] Note that in this paper, we allow ourselves just $O(\log n)$ rounds for pre-computation and each node can learn only $\operatorname{polylog} n$ bits per round given that it has small ($\operatorname{polylog} n$) degree, which can be true for every node. The local network has size $\Omega(n)$, meaning no single node can learn it completely. This inhibits solving the problem locally at some node, i.e., by *direct* use of some centralized algorithm.

[5] While the routing scheme in [27] guarantees a 2-approximation for general graphs regarding the worst-case optimal cost when routing over all possible source-target-pairs, their scheme does not guarantee constant stretch when routing a message between two specific nodes $s, t$ in the grid graph.

with real nodes of $V$ that are close by, where edges between grid nodes correspond to paths of constant length in $G$. We start by introducing some notations we require in the following.

## 2.1 Preliminaries

**Graphs and Polygons in $\mathbb{R}^2$.** Since each node in $V$ is associated with a point in $\mathbb{R}^2$, we can associate each edge $\{u, v\} \in \mathsf{UDG}(V)$ with the line segment with endpoints $u$ and $v$, i.e., the set $\{x \cdot u + (1-x) \cdot v \mid x \in [0, 1]\}$. We use the names of vertices and edges to refer to their associated subsets of $\mathbb{R}^2$ when no ambiguity arises.

A *polygonal chain* is a finite sequence of points where consecutive points are connected by segments. A polygonal chain is *closed* if the first point in the sequence is equal to the last. A *polygon* is a closed, connected, and bounded region in $\mathbb{R}^2$ where the boundary consists of a finite number of (not necessarily disjoint) closed polygonal chains (this implies the edges in these polygonal chains have no proper intersections).

A *hole* of a polygon $P$ is an open region in $\mathbb{R}^2$ that is a maximal bounded and connected component of $\mathbb{R}^2 \setminus P$. Note that the boundary of each hole of $P$ is equal to one of the polygonal chains bounding $P$. A polygon is *simple* if it has no holes.

**Distance Metrics.** We use the notation $\|\cdot\|$ for the Euclidean metric on $\mathbb{R}^2$. Consequently, for $p, q \in \mathbb{R}^2$, $\|p-q\|$ denotes the Euclidean distance from $p$ to $q$. For sets of points $A, B \subseteq \mathbb{R}^2$ we define the distance between those sets as $\operatorname{dist}(A, B) := \min_{a \in A, b \in B} \|a-b\|$.

Let $P \subseteq \mathbb{R}^2$ be a polygon in the Euclidean plane and let $p, q \in P$. We define the *geometric* distance between $p$ and $q$ in $P$, $\operatorname{dist}_P(p, q)$, to be the length of the shortest path between $p, q$ in $P$. Note that because $P$ is a polygon, there is a polygonal chain $\Pi = (v_1, \ldots, v_k)$ from $p$ to $q$ inside $P$ such that $\operatorname{dist}_P(p, q) = \sum_{i=1}^{k-1} \|v_{i+1} - v_i\|$.

Let $G = (V, E)$ be an embedded graph. Let $\Pi \subseteq E$ be a path, i.e., a sequence of incident edges of $G$. Then we define $\operatorname{dist}_G(\Pi) = \sum_{(u,v) \in \Pi} \|u-v\|$. Let $|\Pi|$ be the number of edges (or *hops*) of a path $\Pi$ in $G$. The *hop-distance* between two nodes $u, v \in V$ is defined as $\operatorname{hop}_G(u, v) := \min_{u\text{-}v\text{-path } \Pi} |\Pi|$.

## 2.2 Grid Graph Definition

We first give some definitions to formalize the notion of an UDG having radio-holes. A *triangle of* $\mathsf{UDG}(V)$ is a region in $\mathbb{R}^2$ that is bounded by the edges of a 3-cycle in $\mathsf{UDG}(V)$ (including both the boundary and interior of the triangle). We define the *contour polygon* $P$ *of* $\mathsf{UDG}(V)$ as the union of all triangles and edges of $\mathsf{UDG}(V)$. Since $\mathsf{UDG}(V)$ is connected, $P$ is indeed a polygon. We call the holes in $P$ *radio-holes* of $\mathsf{UDG}(V)$. We say an UDG has *no radio-holes* if the contour polygon of that UDG has no holes, i.e., the polygon $P$ is simple.

Next we partition the plane into an axis-parallel square grid with side-length $c = \frac{1}{10}\sqrt{15}$ and a fixed origin corresponding to origin of the coordinate system. Note that due to knowledge of coordinates, all nodes are aware of their position relative to the grid.

Define a square grid-cell to be *active* if it has a non-empty intersection with $P$. Based on this grid, we define the grid graph $\Gamma = (V_\Gamma, E_\Gamma)$, where $V_\Gamma$ has a node positioned at the center of each active cell in our grid, and we have an edge in $E_\Gamma$ between every pair of nodes of $V_\Gamma$ that lie in adjacent cells in the grid (i.e., the square cells share an edge). The grid graph $\Gamma$ will be simulated in the routing protocol. We will also define the cell graph $\Gamma' = (V_{\Gamma'}, E_{\Gamma'})$ in the analysis of our protocol, but do not simulate it. We call a vertex of the grid *loose* if it is a corner of exactly 2 active cells that are not adjacent. $\Gamma'$ is composed of the boundaries of the square grid, with $V_{\Gamma'}$ the set of all corners of each active grid-cell that are not loose, with a pair of vertices in $V_{\Gamma'}$ having an edge in $E_{\Gamma'}$ if they are ends of an edge of a grid-cell. To define the *cell polygon* $P'$, first take the union of all active grid-cells. Then, for every

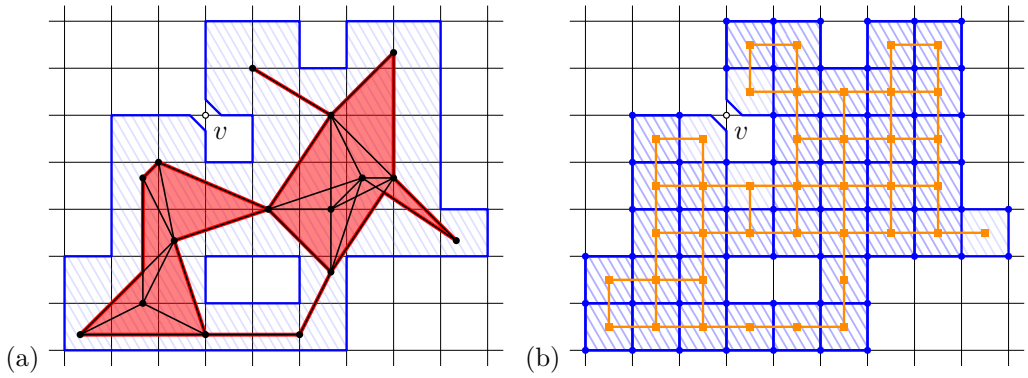



**Figure 1** (a): $G := \mathsf{UDG}(V)$ (black), polygon $P$ (red), and cell polygon $P'$ (blue). (b): grid graph $\Gamma$ (orange), active grid-cells and cell graph $\Gamma'$ (blue). The grid vertex $v$ is loose. Note that $\boldsymbol{G}$ and $\boldsymbol{\Gamma}$ have a hole. Our routing algorithm on $\boldsymbol{\Gamma}$ would not work for this UDG, but we can still construct $\boldsymbol{\Gamma}$.

loose vertex $v$ in the grid, remove a triangle from $P'$ at every active grid-cell incident to $v$ that is small enough to be disjoint from $P$, such that $P'$ no longer contains $v$. Note that since a loose vertex does not lie in $P$ (otherwise, all 4 cells incident to it would have been active), such a triangle exists. See Figure 1 for an example of these definitions. Next, we define a *representative* $r$ for each grid node $g \in V_\Gamma$, which simulates $g$ throughout the rest of the protocol. We apply one of the following rules to assign a grid node to a node $u \in V$.

▶ **Definition 2.** *Let $g \in V_\Gamma$, and let $C$ be the grid cell of which $g$ is the center. We define $\mathcal{C}_1(g)$ as the set of vertices of all triangles of $\mathsf{UDG}(V)$ that contain the point $g$. We define $\mathcal{C}_2(g)$ as the set of vertices incident to an edge that intersects $C$.*

*We define the* set of candidate representatives *$\mathcal{C}(g)$ as $\mathcal{C}_1(g) \cup \mathcal{C}_2(g)$.*

*The representative of $g$ is defined as $r = \arg\min_{v \in \mathcal{C}_1(g)} \|v - g\|$ if $\mathcal{C}_1(g)$ is non-empty, and $r = \arg\min_{v \in \mathcal{C}_2(g)} \|v - g\|$ otherwise. In either case, we break ties by smallest node ID.*

## 2.3 Properties of the Grid Graph

The next step is to show that the grid abstraction introduced in Definition 2 represents the UDG well. In this section we prove several properties to this effect: we show that nodes are adjacent to the representative of the cell which they are in (Lemma 3); that representatives for adjacent grid cells are close (Lemma 4); and that the cell polygon $P'$ is simple (Lemma 5).

For brevity, all proofs in this section are delegated to the full version .

▶ **Lemma 3.** *Let $u, r \in V$. If $r$ is the representative of the cell $C$ containing $u$, then $\mathrm{hop}_G(u, r) \leq 1$*

Intuitively, this is true because $u$ must be close to the centre of $C$, as must $r$: even if these nodes are different they cannot be too far apart.

▶ **Lemma 4.** *Let $(g_1, g_2) \in E_\Gamma$ be an edge in $\Gamma$. Let $u, v \in V$ be representatives of $g_1, g_2$ respectively. Then $\mathrm{hop}_G(u, v) \leq 3$.*

We show that the edges which define $\mathcal{C}(g_1)$ and $\mathcal{C}(g_2)$ are at most the diagonal of a $2 \times 1$ block of grid cells apart. We conclude that this distance is small enough that an edge connects an endpoint of one edge with an endpoint from the other, and so the representatives of adjacent cells have distance at most 3 from each other.

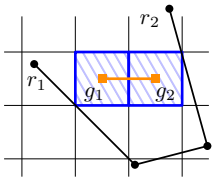



**Figure 2** Representatives $r_1, r_2$ of adjacent grid nodes $g_1, g_2$ are connected by a path of 3 hops.

Finally, we show that $P'$ is simple, i.e., it has no holes. We show this by observing that if there is a hole in $P'$, there is a cycle of active cells with an inactive cell in its interior. We show this cycle of cells contains a cycle of $G$, which implies $G$ contains a radio-hole.

▶ **Lemma 5.** *If $G$ has no holes, then $P'$ is simple.*

## 2.4 Grid Graph Representation, Computation and Simulation

Building on the previous subsections, we show that we can efficiently simulate the grid graph $\Gamma$ with a sub-graph $R = (V_R, E_R)$ of the UDG $G$ which we call a *representation* of $\Gamma$ in $G$ which closely approximates the structure of $\Gamma$. In a nutshell: the set of nodes $V_R$ contains the set of representatives of all grid nodes $V_\Gamma$. On top of that, for each grid edge in $E_\Gamma$, we add a path in the UDG $G$ to $R$ between two representatives of the corresponding grid nodes (see example in Figure 2). Note that in the previous subsection we have shown the existence of such paths that have at most 3 hops.

The first goal of this subsection is to thoroughly define $R$ and to show that we can compute $R$ in $G$ according to that definition in $O(1)$ rounds. The second goal is to give an interfacing theorem for later sections that purely work with $\Gamma$, showing that a round of HYBRID in the grid graph $\Gamma$ can be simulated in $O(1)$ rounds by the nodes in $R$ (the proof is also given in the full version). By *simulation*, we mean that one round of local communication between adjacent grid nodes in $\Gamma$ can be performed using $O(1)$ rounds of local communication in $G$ to route messages between the representatives of adjacent grid nodes. An analogous property holds for the global communication.

▶ **Definition 6.** *Let $\Gamma = (V_\Gamma, E_\Gamma)$ be the grid graph as defined in Section 2. A* representation *$R = (V_R, E_R)$ of $\Gamma$ in $G$ is a sub-graph of $G$ defined as follows. For every grid node $g \in V_\Gamma$ with representative $r \in V$ we define: $r \in V_R$. For each edge $\{g_1, g_2\} \in E_\Gamma$ let $r_1, r_2 \in V$ be the corresponding representatives. Then $R$ contains all nodes and edges of one $r_1$-$r_2$-path $\Pi_{r_1,r_2}$ in $G$ such that $|\Pi_{r_1,r_2}| \leq 3$. We call $\Pi_{r_1,r_2}$ the representation of the edge $\{g_1, g_2\}$. Note that such a path always exists due to Lemma 4.*

▶ **Lemma 7.** *A representation $R = (V_R, E_R)$ of $\Gamma$ can be computed in $O(1)$ rounds.*

▶ **Theorem 8.** *A round of the* HYBRID *model in $\Gamma$ can be simulated in $O(1)$ rounds.*

# 3 Constant Stretch Routing Scheme for the UDG

It remains to show how to leverage the grid graph constructed in the previous section for the computation of routing schemes for the UDG assuming that an exact routing scheme for the grid graph is known. We start with the analysis of the approximation factor.

## 3.1 From Shortest Paths in $\Gamma$ to Approximate Paths in $G$

The goal of this subsection is to show that shortest paths in the simulated grid graph $\Gamma$ represent good paths in the UDG $G$. In particular, paths in $G$ that are obtained via the representation $R$ of $\Gamma$ are constant approximations of *optimal* paths in $G$, both in terms of hop-length and Euclidean distance. We start by defining a *representative path*.

▶ **Definition 9.** *Let $s, t \in V$. Let $g_s, g_t \in V_\Gamma$ be the two grid nodes which are located in the same grid cell as $s, t$ respectively. Let $r_s, r_t$ be the representatives of $g_s$ and $g_t$. Note that $\{s, r_s\}, \{r_t, t\} \in E_G$ due to Lemma 3. Consider an optimal $g_s$-$g_t$-path $\Pi^*$. For $e \in \Pi^*$ let $\Pi_e$ be the representation of the grid edge $e \in E_\Gamma$ (see Definition 6). Let $\Pi_{r_s,r_t} := \bigcup_{e\in\Pi^*} \Pi_e$. We define the* representative $s$-$t$-path *as $\Pi_{s,t} := \{\{s, r_s\}\} \cup \Pi_{r_s,r_t} \cup \{\{r_t, t\}\}$.*

We will show that our routing scheme routes packets from $s$ to $t$ along the representative path $s$-$t$-path $\Pi_{s,t}$. First, we show that these paths achieve constant stretch in $G$.

▶ **Theorem 10.** *Let $s, t \in V$. Let $\Pi_{s,t}$ be the $s$-$t$-path given in Def. 9. If $\{s, t\} \notin E_G$ Then $\text{dist}(\Pi_{s,t}) \leq |\Pi_{s,t}| \leq 36 \cdot \text{dist}_G(s, t)$.*

Note that if $\{s, t\} \in E_G$ then we can send the packet directly along this edge and the distance and number of hops is guaranteed to be optimal. If $\{s, t\} \notin E_G$ then $dist_G(s, t) > 1$, a fact which we use in the proof of Theorem 10. We prove this theorem in stages represented by the subsequent lemmas. In the first stage we upper bound the number of hops of the representative path $\Pi_{s,t}$ with the distance of a corresponding $g_s, g_t$-path in $\Gamma$.

▶ **Lemma 11.** *Let $s, t \in V$. Then $|\Pi_{s,t}| \leq \frac{3}{c} \cdot \text{dist}_\Gamma(g_s, g_t) + 2$.*

**Proof.** We exploit the fact that edges in $\Gamma$ have distance at least $c$ and that $\Pi_{s,t} = \{\{s, r_s\}\} \cup \Pi_{r_s,r_t} \cup \{\{r_t, t\}\}$ is constructed from an optimal path in $\Gamma$ (see Definition 9). We combine this with Lemmas 3, 4 to obtain the following $|\Pi_{s,t}| \overset{\text{Lem. } 3}{\leq} |\Pi_{r_s,r_t}| + 2 \overset{\text{Lem. } 4}{\leq} 3 \cdot \text{hop}_\Gamma(g_s, g_t) + 2 \leq \frac{3}{c} \text{dist}_\Gamma(g_s, g_t) + 2$. ◀

Since the cell-polygon $P'$ completely covers $P$ (the smallest polygon containing all edges of $\Gamma$ does not, in general), we relate paths in the grid graph $\Gamma$ to paths in the cell-graph $\Gamma'$. This allows us to relate paths in $P$ to $\Gamma$. Note that comparisons of hop-distance in $\Gamma$ and $\Gamma'$ correspond to equal comparisons of distances, since both graphs have the same granularity $c$.

▶ **Lemma 12.** *Let $g_1, g_2 \in V_\Gamma$ be located in cells $C_1, C_2$, respectively. There exist nodes $g_1', g_2' \in V_{\Gamma'}$ that are corners of $C_1, C_2$ respectively, such that $\text{dist}_\Gamma(g_1, g_2) \leq 2 \cdot \text{dist}_{\Gamma'}(g_1', g_2')$.*

**Proof.** Choose $g_1', g_2'$ such that $\text{hop}_G(g_1', g_2') \geq 1$. Let $\Pi'$ be a shortest $g_1'g_2'$-path in $\Gamma'$. Since all edges and vertices of $\Gamma'$ are part of the boundary of an active grid cell and $\Gamma'$ contains no loose vertices, there is a sequence $A$ of active grid-cells from $C_1$ to $C_2$, where consecutive cells share a side and each cell has an edge or vertex of $\Pi'$ on its boundary. There are two kinds of cells in $A$: the first kind has an edge of $\Pi'$ on its boundary, the second kind does not have an edge of $\Pi'$ on its boundary, but has a vertex of $\Pi'$ on its boundary. The number of cells of the first kind is at most $|\Pi'|$, because each edge in $\Pi'$ is adjacent to at most one cell of $A$. The number of cells of the second kind is at most $|\Pi'| + 1$, because each vertex of $\Pi'$ is adjacent to at most one cell of this type (since $\Pi'$ has at least one edge.). So, $|A| \leq 2|\Pi'| + 1$.

We obtain a $g_1g_2$-path $\Pi$ of length $|A|-1$ in $\Gamma$ from the chain $A$ by taking the vertex centered at each cell in $A$. So, we have $\text{hop}_\Gamma(g_1, g_2) \leq |\Pi| \leq |A|-1 \leq 2|\Pi'| = 2 \text{ hop}_{\Gamma'}(g_1', g_2')$. Since all edges in $\Gamma$ and $\Gamma'$ have length $c$, we have $\text{dist}_\Gamma(g_1, g_2) \leq 2 \text{ dist}_{\Gamma'}(g_1', g_2')$. ◀

We follow up on the previous stage, and bound the distance of an optimal path in the *graph* $\Gamma'$ with that of an optimal *geometric* path in the *polygon* $P'$. The resulting approximation factor of $\sqrt{2}$ stems from a segment-wise comparison of Euclidean distance of a shortest polygonal chain in $P'$ to the Manhattan distance in the graph $\Gamma'$.

▶ **Lemma 13.** *Let* $g_1, g_2 \in V_{\Gamma'}$. *Then* $\text{dist}_{\Gamma'}(g_1, g_2) \leq \sqrt{2} \cdot \text{dist}_{P'}(g_1, g_2)$.

**Proof.** Let $\Pi$ be the shortest *geometric* path from $g_1$ to $g_2$ in $P'$. Since $P'$ is a polygon, $\Pi$ is a polygonal chain connecting vertices $g_1 =: v_1, \ldots, v_n := g_2$, where $v_i$ are reflex vertices (i.e., vertices with an internal angle of at least $\pi$) of $P'$. Note that by construction of $P'$, all reflex vertices of $P'$ are vertices of $\Gamma'$, so we have $v_1, \ldots, v_n \in V_{\Gamma'}$.

Consider one such segment $s_i$. Each point of $s_i$ lies in some gridcell belonging to $P'$, because the path $\Pi$ lies in $P'$. Therefore, there is a monotone chain of gridcells connecting $v_i$ and $v_{i+1}$. Consider the axis aligned bounding rectangle $R_i$ defined by the two opposite corners $v_i, v_{i+1} \in V_{\Gamma'}$. The width and length of $R_i$ sum up to $\|v_i - v_{i+1}\|_1$ (where $\|(x, y)\|_1 = x + y$ for some $(x, y) \in \mathbb{R}^2$ denotes the $L_1$-norm).

Traversing the boundary of the monotone chain of gridcells between $v_i$ and $v_{i+1}$ in the shortest possible way represents a shortest path between $v_i$ and $v_{i+1}$ in $\Gamma'$. On one hand, the length of this path equals the sum of side-lengths of $R_i$, i.e., $\text{dist}_{\Gamma'}(v_i, v_{i+1}) = \|v_i - v_{i+1}\|_1$. On the other hand the geometric distance equals the length of $s_i$ which is $\text{dist}_{P'}(v_i, v_{i+1}) = \|v_i - v_{i+1}\|$. We have

$$\text{dist}_{\Gamma'}(v_i, v_{i+1}) = \|v_i - v_{i+1}\|_1 \leq \sqrt{2} \cdot \|v_i - v_{i+1}\|_2 = \sqrt{2} \cdot \text{dist}_{P'}(v_i, v_{i+1}),$$

using the equivalence property of $L_1$ and $L_2$-norms: $\|x\|_1 \leq \sqrt{2}\|x\|_2$ for any $x \in \mathbb{R}^2$. So, for each segment $S_i$ of $\Pi$, there exists a path in $\Gamma'$ with stretch at most $\sqrt{2}$ connecting the endpoints. Concatenating these paths gives the required $g_1$-$g_2$-path in $\Gamma'$. ◀

Next we observe that an optimal path between two nodes in the UDG $G$ can not be any shorter than a corresponding shortest geometric path in $P'$.

▶ **Lemma 14.** *Let* $s, t \in V$. *Then* $\text{dist}_{P'}(s, t) \leq \text{dist}_G(s, t)$.

**Proof.** Let $\Pi$ be a shortest $st$-path in $G$. By definition, each cell that is intersected by an edge of $\Pi$ is active and therefore this edge lies in $P'$. So, $\Pi$ is an $st$-path in $P'$. ◀

We now use the inequalities proven in the lemmas above to prove Theorem 10.

**Proof of Theorem 10.** Let $s, t \in V$ and let $g_s, g_t \in \Gamma$ be their cell representatives. Let $g'_s, g'_t \in V_{\Gamma'}$ be two corner-nodes of $g_s, g_t$ for which Lemma 12 holds. Then we get

$$\begin{aligned}
|\Pi_{s,t}| &\leq \tfrac{3}{c} \cdot \text{dist}_{\Gamma}(g_s, g_t) + 2 && \textit{Lemma 11} \\
&\leq \tfrac{6}{c} \cdot \text{dist}_{\Gamma'}(g'_s, g'_t) + 2 && \textit{Lemma 12} \\
&\leq \tfrac{6\sqrt{2}}{c} \cdot \text{dist}_{P'}(g'_s, g'_t) + 2 && \textit{Lemma 13} \\
&\leq \tfrac{6\sqrt{2}}{c} \cdot \big(\text{dist}_{P'}(g'_s, s) + \text{dist}_{P'}(s, t) + \text{dist}_{P'}(t, g'_t)\big) + 2 && \textit{triangle ineq.} \\
&= \tfrac{6\sqrt{2}}{c} \cdot \big(\|g'_s - s\| + \text{dist}_{P'}(s, t) + \|g'_t - t\|\big) + 2 && \textit{sg}'_s \textit{ and } \textit{tg}'_t \textit{ in same cell} \\
&\leq \tfrac{6\sqrt{2}}{c} \cdot \big(\|g'_s - s\| + \text{dist}_{G}(s, t) + \|g'_t - t\|\big) + 2 && \textit{Lemma 14} \\
&= \tfrac{6\sqrt{2}}{c} \cdot \big(\text{dist}_G(s, t) + \sqrt{2} \cdot c\big) + 2 = \tfrac{6\sqrt{2}}{c} \cdot \text{dist}_G(s, t) + 14
\end{aligned}$$

In the equality in the fourth step we use that the segments $sg'_s$ and $tg'_t$ are both contained in a single grid cell, hence the distance in the cell-polygon equals the Euclidean distance.

Since a grid cell has side length $c$, we have $\|g'_s - s\|, \|g'_t - t\| \leq \frac{1}{2}\sqrt{2} \cdot c$ in the second last step. As $\Pi_{s,t}$ is a path in a UDG each edge has distance at most 1, thus

$$\text{dist}_G(\Pi_{s,t}) \leq |\Pi_{s,t}| \leq \tfrac{6\sqrt{2}}{c} \cdot \text{dist}_G(s,t) + 14 \leq 22 \cdot \text{dist}_G(s,t) + 14.$$

Since we have a direct edge to targets with distance at most 1, the additive error can be accounted for by increasing the multiplicative stretch by the additive error for targets at distance more than 1. Consequentially, we obtain $\text{dist}(\Pi_{s,t}) \leq |\Pi_{s,t}| \leq 36 \cdot \text{dist}_G(s,t)$. ◀

## 3.2 Transforming Routing Schemes for the Grid Graph to the UDG

We provide an interface to transform a routing scheme $\mathcal{R}_\Gamma$ for the grid graph $\Gamma$ (for which an exact routing scheme is provided in the subsequent section) into a routing scheme $\mathcal{R}$ for the UDG $G$ with constant stretch. The idea is to construct $\mathcal{R}$ from $\mathcal{R}_\Gamma$ using the representation $R$ of $\Gamma$ (see Definition 6). Theorem 16 provides approximation guarantees by leveraging the insights on representative paths from the previous subsection (for a proof see the full version).

▶ **Definition 15** (UDG Routing Scheme). *Let $\mathcal{R}_\Gamma$ be an* exact *routing scheme for $\Gamma$ consisting of $\ell_\Gamma : V_\Gamma \to \{0,1\}^+$ and $\rho_\Gamma : V_\Gamma \times \{0,1\}^+ \to V_\Gamma$. Let $R = (V_R, E_R)$ be the representation of $\Gamma$ (see Def. 6). The routing scheme $\mathcal{R}$ for $G$ is defined on the basis of grid cells. Let $C$ be a cell with grid node $g \in V_\Gamma$ and let $r \in V_R$ be the representative of $g$. For each $v \in C$ we set $\ell_G(v) := \ell_\Gamma(g) \circ ID(v)$ (where "$\circ$" represents the concatenation of bit strings). The routing function $\rho_G$ is defined as follows. Let $v \in V_G$ be the current node and let $\ell_t := \ell_{\Gamma,t} \circ ID(t)$ be the label of the target node $t \in V$, where $\ell_{\Gamma,t}$ is the label of the representative in $t$'s cell w.r.t. $\mathcal{R}_\Gamma$. We assume $t \neq v$, as otherwise the packet has already arrived.*

1. *If $\{v,t\} \in E_G$, then we can directly deliver to $t$: $\rho_G(v, \ell_t) := t$.*
2. *Else, if $v \in C \setminus V_R$ is the source we directly route to the representative of $C$: $\rho_G(v, \ell) := r$.*
3. *Else, if $v = r$ is the representative of this grid cell $C$, let $g' := \rho_\Gamma(g, \ell_{\Gamma,t})$ be the next grid node suggested by $\mathcal{R}_\Gamma$. Let $u$ be the first node on the path $\Pi_{\{g,g'\}} \subseteq E_R$ that represents the edge $\{g,g'\} \in E_\Gamma$. Then $\rho_G(v, \ell_t) := u$.*
4. *Else, if $v \in V_R$ but $v$ is not the representative of $C$, then $v$ must be a "transitional node" on $\Pi_{\{g,g'\}} \in E_R$ that represents $\{g,g'\} \in E_\Gamma$. W.l.o.g. let $g' := \rho_\Gamma(g, \ell_{\Gamma,t})$ be the next grid node suggested by $\mathcal{R}_\Gamma$ and $u$ be the next node on $\Pi_{\{g,g'\}}$ towards $g'$. Then $\rho_G(v, \ell_t) := u$.*

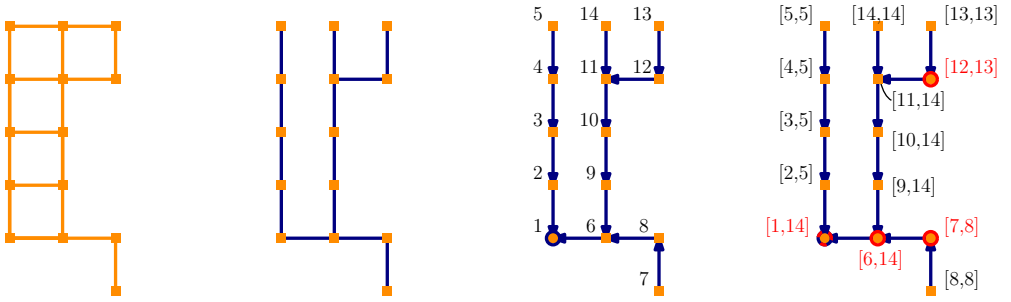
▶ **Theorem 16.** *Let $\mathcal{R}_\Gamma$ be a local, correct, exact routing scheme for $\Gamma$ with labels and local routing information of $O(\log n)$ bits. Then the routing scheme $\mathcal{R}$ from Definition 15 is local, correct, has constant stretch, labels and local routing information of size $O(\log n)$ bits and can be computed in $O(1)$ rounds.*

# 4 Computing a Labelling for the Grid Graph

This section is dedicated to computing the labelling $\ell_\Gamma : V_\Gamma \to \{0,1\}^+$ for the grid graph by first constructing a particular tree structure $T_\Gamma$ and then computing a labelling on it in $O(\log n)$ rounds leveraging various HYBRID (and in particular $\mathsf{NCC}_0$) model techniques. For the tree-labelling we use a similar approach as presented in [27], but slightly adapt the labelling which later allows jumping over branches of our specifically constructed tree, facilitating an optimal routing scheme in grid graphs. Afterwards, Section 5 will deal with computing the routing function $\rho_\Gamma : V_\Gamma \times \{0,1\}^+ \to V_\Gamma$ leading to the routing scheme $\mathcal{R}_\Gamma$.

We assume the HYBRID model on the grid graph $\Gamma$ that represents the network which we constructed and simulated in the previous sections (Theorem 8). The goal is to divide the

grid nodes into sets of vertically connected grid nodes called *portals*. Connecting neighboring portals with a single edge gives us a spanning tree of $\Gamma$, which we call *portal tree*. We then root the portal tree at the node with minimum identifier and compute a label for each grid node, leading to a well-defined labelling function $\ell_\Gamma$. Note that we require that the cell polygon $P'$ does not contain holes (Lemma 5), as otherwise there be a cycle after connecting neighboring portals.




**(a)** Initial grid graph $\Gamma$. **(b)** Portal tree $T_\Gamma$ (blue). **(c)** Rooted portal tree, values $\ell_v$. **(d)** Label $[l_v,r_v]$, portal label red.

**Figure 3** Example for creation and labelling of the portal tree.

We first define the set of portals as follows:

▶ **Definition 17** (Portals). *Let $\Gamma = (V_\Gamma, E_\Gamma)$ be the grid graph as constructed in the last section. The* set of portals *are the connected components of $(V_\Gamma, E_{vert})$, where $E_{vert} \subset E_\Gamma$ are the vertical edges of the grid graph.*

For convenience, assume that the grid nodes $v_1, \dots, v_k$ within a portal $\mathcal{P}$ are sorted by their $y$-coordinates in descending order, i.e., $v_1$ is the northernmost node.

To construct the portal tree $T_\Gamma$ of the grid graph $\Gamma$ we connect neighboring portals via a single edge. Each grid node $v$ checks whether it has an edge to the left and communicates this to its northern and southern neighbors $v_N$ and $v_S$. Assume that $v$ has an edge $\{v, v_W\}$ to the left. Then $v$ checks if $v_S$ also has a horizontal edge to the left. If that is not the case, $v$ adds the edge $\{v, v_W\}$ to the portal tree. We refer to Figures 3a and 3b for an example. This gives us the following lemma, the proof of which is delegated to Appendix C.1:

▶ **Lemma 18.** *The portal tree $T_\Gamma$ of a grid graph $\Gamma$ can be computed in $O(1)$ rounds.*

Given the portal tree $T_\Gamma$, we want to compute a unique label for each grid node that reflects its structure as portal tree. First, we root $T_\Gamma$ at the grid node $r$ whose representative is the UDG node $u$ with minimal identifier, using pointer jumping (Appendix C.2) on the cycle of all grid nodes that corresponds to an Euler tour (Appendix C.3).

Now we compute the labelling for the (rooted) portal tree. For each grid node $v$ in $T_\Gamma$, we aim to assign an interval $I_v = [l_v, r_v] \in \mathbb{N}^2$ to $v$, such that $I_v \supset I_w$ for any child node $w$ of $v$ in $T_\Gamma$. To obtain the left interval border $l_v$ for each grid node $v$ in the portal tree, we perform a depth-first traversal (DFS) on $T_\Gamma$ in $O(\log n)$ rounds, using Lemma 28 (see Appendix C.4). The value $l_v$ is then the preorder number of $v$ according to the DFS. Note that $l_v < l_u$ for any node $u$ lying in the subtree of $v$. We then compute the number $r_v$, corresponding to the maximum left interval border among all nodes in $v$'s subtree. In a nutshell, we first let all nodes compute some value $d \in O(\log n), d \geq \log D(T_\Gamma)$, where $D(T_\Gamma)$ is the depth of the portal tree. Then we generate additional edges in $T_\Gamma$ for $d$ iterations, by performing pointer-jumping on the paths from the leaf nodes of $T_\Gamma$ to the root. We perform the pointer-jumping technique in a condensed way to ensure that the node degrees do not exceed $O(\log n)$. With the help of these additional edges, we let each node $v \in T_\Gamma$ compute the value $r_v$ as an aggregate of the $l_u$-values of all nodes $u$ that are contained in the subtree $T_\Gamma(v)$ of $T_\Gamma$ with root $v$. We elaborate on this approach in Appendix C.5 (see Lemma 29).

After the algorithm has terminated, each node $v$ knows the correct value $r_v$ and thus its interval $I_v = [l_v, r_v]$. Observe that grid nodes which are in different branches of the portal tree have incomparable labels. We obtain the following lemma:

▶ **Lemma 19.** *Given a rooted portal tree $T_\Gamma$, each node $v \in T_\Gamma$ can compute an interval $I_v = [l_v, r_v]$ in $O(\log n)$ rounds, such that $I_v \supset I_w$ for any child node $w$ of $v$ in $T_\Gamma$.*

Now that each grid node $v$ knows its interval in $T_\Gamma$ we need to perform one final step. In addition to its own (unique) interval, a grid node $v$ needs to know the interval that has been assigned to the node $v_i$ which is closest to the root within its own portal. We call this label the *portal label* of $v$. The node on a portal which is closest to the root can determine this locally. Each portal label can then be broadcasted to all nodes within the respective portal in $O(\log |\mathcal{P}|)$ rounds (see Lemma 26), so we obtain the following lemma (cf. Figures 3c and 3d).

▶ **Lemma 20.** *After $O(\log n)$ rounds, each grid node $v$ in the portal $\mathcal{P} = (v_1, \ldots, v_k)$ knows the interval $I_{v_i}$ of the node $i \in P$ closest to the root of the portal tree.*

Observe that, the way we defined the portal labels we obtain the property that for portal labels of two neighboring portals, one portal's label is always a subset of the other. Combining Lemma 19 and Lemma 20 yields the main result of this section.

▶ **Theorem 21.** *Computing the labelling $\ell_\Gamma : V_\Gamma \to \{0,1\}^+$ for the grid graph $\Gamma$ as part of the routing scheme $\mathcal{R}_\Gamma$ can be done within $O(\log n)$ rounds.*

## 5 Compact Routing Scheme for the Grid Graph

Finally, we explain our routing strategy for transmitting a packet between two nodes $s, t \in V_\Gamma$ in the grid graph, leading to the routing function $\rho_\Gamma : V_\Gamma \times \{0,1\}^+ \to V_\Gamma$. At the start of the routing protocol, the node $s$ generates a message $m$ that contains the identifier of the target node $t$, as well as $t$'s label and portal label. The goal of our routing strategy is to route $m$ to $t$ along grid edges via an optimal path in the grid graph. To do so, each grid node receiving the message $m$ has to decide which of its grid neighbors to forward $m$ to, using only the information stored in $m$, and the information stored in its own local memory. Briefly, the strategy works as follows. While we are not at the portal containing $t$, we always try going left (west) or right (east) first by going to a portal whose label is closest to the portal label of the target node $t$. If going east or west is not possible, we go up (north) or down (south) instead by comparing $g$'s own label with the *actual* label of the target node $t$. Once we are at the portal that contains the target node, we only consider going up or down until we reach $t$.

**Detailed Description.** We describe the routing strategy in more detail now (see the full version for pseudocode). Assume we are at a grid node $g$ and want to route a message $m$ to a grid node $t$. We introduce the following notation for the information known to $g$. Note that grid nodes obtain this information in one communication round with their neighbors.

▶ **Definition 22.** *The information required to be stored by a grid node $g \in V_\Gamma$ are denoted by the following variables.*

- **(i)** *$g.L \in \mathbb{N}^2$: $g$'s own interval given to it by labelling of the portal tree.*
- **(ii)** *$g.P \in \mathbb{N}^2$: The portal label of the portal containing $g$.*
- **(iii)** *$g_N, g_S, g_E, g_W \in V_\Gamma \cup \{\perp\}$: $g$'s grid neighbors in north, south, east and west direction ($\perp$ denotes that there is no such neighbor). For each of these grid neighbors $g$ also knows the label of the grid node and the portal label of the grid node.*

Additionally, we store the label $t.L$ of $t$ and the portal label $t.P$ of $t$ in the message $m$, so $g$ knows these as well upon receipt of $m$. Note that storing this information at $g$ requires only $O(\log n)$ bits. Assuming that $g \neq t$, $g$ must decide which of its grid neighbors $g_W, g_E, g_N, g_S$

to forward $m$ to. Node $g$ first checks if it is in the same portal as $t$ by comparing $g.P$ and $t.P$. Assume that this is not the case. Then $g$ has to consider the following cases. We use the notation $a \nsim b$ to denote that label $a$ is *incomparable* to label $b$, i.e., $a \not\subseteq b \wedge b \not\subseteq a$.

We start by explaining how a message $m$ is routed in horizontal direction.

**(i)** $g.P \subset t.P$ or $g.P \supset t.P$. In case $g.P \subset t.P$ then $g$ checks if either $g.P \subset g_W.P \subset t.P$ or $g.P \subset g_E.P \subset t.P$ holds (only one of these conditions can be true). In the first case, $g$ forwards $m$ to $g_W$, in the second case $g$ forwards $m$ to $g_E$. If none of the conditions hold (for example, if $g_W = \perp$ or $g_E = \perp$), then $g$ routes $m$ vertically (see the description below). The case $g.P \supset t.P$ works analogously.

**(ii)** $g.P \nsim t.P$. In this case $g$ tries to forward $m$ horizontally to a node, whose portal label is a superset of $g.P$. By doing so, $m$ eventually reaches a node $g'$ whose portal label is also a superset of $t.P$ (at the closest "common ancestor portal"), and case (i) is considered. If neither $g_W$ nor $g_E$ satisfies this condition or does not exist, $g$ routes vertically.

We now explain how $m$ is routed in vertical direction. We do this if $g$ has not been able to route $m$ horizontally (either because its horizontal neighbors are not appropriate, or because they do not exist) or if it is already contained in the same portal as the target node $t$. Again, $g$ considers the following cases, this time for its own label $g.L$ instead for $g.P$ and for the actual label $t.L$ instead of the portal label $t.P$.

**(i)** $g.L \subset t.L$ or $g.L \supset t.L$. In the case $g.L \subset t.L$ node $g$ checks if either $g.L \subset g_N.L \subset t.L$ or $g.L \subset g_S.L \subset t.L$ holds. In the first case, $g$ forwards $m$ to $g_N$, in the second case $g$ forwards $m$ to $g_S$. The case $g.L \supset t.L$ works analogously.

**(ii)** $g.L \nsim t.L$. If the labels $g.L$ and $t.L$ are incomparable, $g$ tries to forward $m$ vertically to a node, whose label is a superset of $g.L$. This is the case for either $g_N$ or $g_S$, depending on the location of the root of the labeled tree.

**Analysis of the Routing Strategy.** We show that our routing strategy is local, efficient, and correct, so it fulfills all requirements for a routing scheme. Our routing strategy is local, as each node $v$ can determine the next node to forward the message $m$ to based solely on the $O(\log n)$ bits of local information, and the labels $t.L$ and $t.P$ given to $v$ upon receipt of $m$.

Regarding efficiency of our routing strategy, we prove in the full version that it is optimal. The idea is to show that in case the message is routed in a specific direction, there exists at least one optimal path that moves in the same direction. We conclude the following theorem.

▶ **Theorem 23.** *A local, correct and exact routing scheme $\mathcal{R}_\Gamma$ for $\Gamma$ using node labels and local space of $O(\log n)$ bits can be computed in $O(\log n)$ rounds in the* HYBRID *model.*

## 6 Conclusion

We showed that for any HYBRID network with a hole-free UDG($V$), a compact routing scheme can be computed for UDG($V$) in just $O(\log n)$ rounds. There are various interesting directions for follow-up research. For example, we suspect that our approach can be generalized to 3 dimensions (potentially more) where the corresponding "unit ball graph" implies a polyhedron of genus 0. In particular, some approach akin to multidimensional range trees might work: define $\Gamma$ analogously in a three dimensional grid; dissect $\Gamma$ along 2d-hyperplanes to obtain 2d-portals in $\Gamma$ – if one then comes up with a routing scheme to find the correct 2d-portal, then this can be applied alongside the 2d-routing algorithm presented here to find the correct node in that 2d-portal. There are unresolved issues, however. Another interesting direction is to efficiently compute compact routing schemes for *arbitrary* connected UDGs, or ideally,

to find efficient solutions for arbitrary planar graphs. This seems to be a daunting task; a simpler setting might be to consider UDGs with a small number of holes where our grid construction could be of help. Finally, it would be interesting to think about adaptations of our routing scheme to also minimize congestion, which should be possible in the special case of hole-free UDGs (see for example the case where the contour polygon is a square [7]).

## References


1. Yehuda Afek, Gad M. Landau, Baruch Schieber, and Moti Yung. The power of multimedia: Combining point-to-point and multiaccess networks. *Information and Computation*, 84(1):97–118, January 1990. doi:10.1016/0890-5401(90)90035-G.
2. John Augustine, Keerti Choudhary, Avi Cohen, David Peleg, Sumathi Sivasubramaniam, and Suman Sourav. Distributed graph realizations. In *Proc. of the 34th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2020)*, pages 158–167, 2020. doi:10.1109/IPDPS47924.2020.00026.
3. John Augustine, Kristian Hinnenthal, Fabian Kuhn, Christian Scheideler, and Philipp Schneider. Shortest paths in a hybrid network model. In *Proc. of the 31st ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*, pages 1280–1299, 2020. doi:10.1137/1.9781611975994.78.
4. Nicolas Bonichon, Prosenjit Bose, Jean-Lou De Carufel, Vincent Despré, Darryl Hill, and Michiel H. M. Smid. Improved routing on the Delaunay triangulation. In *Proc. of the 26th Annual European Symposium on Algorithms (ESA 2018)*, pages 22:1–22:13, 2018. doi:10.4230/LIPIcs.ESA.2018.22.
5. Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001. doi:10.1023/A:1012319418150.
6. Jehoshua Bruck, Jie Gao, and Anxiao Jiang. MAP: medial axis based geometric routing in sensor networks. *Wireless Networks*, 13(6):835–853, 2007. doi:10.1007/s11276-006-9857-z.
7. Antonio Carzaniga, Koorosh Khazaei, and Fabian Kuhn. Oblivious low-congestion multicast routing in wireless networks. In *Proc. of the 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2012)*, pages 155–164, 2012. doi:10.1145/2248371.2248395.
8. Jannik Castenow, Christina Kolb, and Christian Scheideler. A bounding box overlay for competitive routing in hybrid communication networks. In *Proc. of the 21st International Conference on Distributed Computing and Networking (ICDCN 2020)*, pages 14:1–14:10, 2020. doi:10.1145/3369740.3369777.
9. Keren Censor-Hillel, Dean Leitersdorf, and Volodymyr Polosukhin. Distance computations in the hybrid network model via oracle simulations. *CoRR*, abs/2010.13831, 2020. arXiv:2010.13831.
10. Hristo N. Djidjev, Grammati E. Pantziou, and Christos D. Zaroliagis. Computing shortest paths and distances in planar graphs. In *Proc. of the 18th International Colloquium on Automata, Languages and Programming (ICALP 1991)*, pages 327–338, 1991. doi:10.1007/3-540-54233-7_145.
11. Michael Feldmann, Kristian Hinnenthal, and Christian Scheideler. Fast hybrid network algorithms for shortest paths in sparse graphs. In *Proc. of the 24th International Conference on Principles of Distributed Systems (OPODIS 2020)*, pages 31:1–31:16, 2020. doi:10.4230/LIPIcs.OPODIS.2020.31.
12. Klaus-Tycho Foerster and Stefan Schmid. Survey of reconfigurable data center networks: Enablers, algorithms, complexity. *SIGACT News*, 50(2):62–79, 2019. doi:10.1145/3351452.3351464.
13. Jie Gao and Mayank Goswami. Medial axis based routing has constant load balancing factor. In *Proc. of the 23rd Annual European Symposium on Algorithms (ESA 2015)*, pages 557–569, 2015. doi:10.1007/978-3-662-48350-3_47.

14 Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM Journal on Computing*, 35(1):151–169, 2005. doi:10.1137/S0097539703436357.

15 Robert Gmyr, Kristian Hinnenthal, Christian Scheideler, and Christian Sohler. Distributed monitoring of network properties: The power of hybrid networks. In *Proc. of the 44th International Colloquium on Automata, Languages and Programming (ICALP 2017)*, pages 137:1–137:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.137.

16 Thorsten Götte, Kristian Hinnenthal, Christian Scheideler, and Julian Werthmann. Time-optimal construction of overlay networks. *CoRR*, abs/2009.03987, 2020. arXiv:2009.03987.

17 Anupam Gupta, Amit Kumar, and Rajeev Rastogi. Traveling with a pez dispenser (or, routing issues in MPLS). *SIAM Journal on Computing*, 34(2):453–474, 2004. doi:10.1137/S0097539702409927.

18 Fabian Höflinger, Joan Bordoy, Rui Zhang, Amir Bannoura, Nikolas Simon, Leonhard M. Reindl, and Christian Schindelhauer. Localization system based on ultra low-power radio landmarks. In *Proc. of the 7th International Conference on Sensor Networks (SENSORNETS 2018)*, pages 51–59, 2018. doi:10.5220/0006608800510059.

19 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. *Algorithmica*, 80(3):830–848, 2018. doi:10.1007/s00453-017-0308-2.

20 Dimitris J Kavvadias, Grammati E Pantziou, Paul G Spirakis, and Christos D Zaroliagis. Hammock-on-ears decomposition: A technique for the efficient parallel solution of shortest paths and other problems. *Theoretical Computer Science*, 168(1):121–154, 1996. doi:10.1016/S0304-3975(96)00065-5.

21 Fabian Kuhn and Philipp Schneider. Computing shortest paths and diameter in the hybrid network model. In *Proc. of the 39th Annual ACM Symposium on Principles of Distributed Computing (PODC 2020)*, pages 109–118, 2020. doi:10.1145/3382734.3405719.

22 Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *Proc. of the 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, pages 63–72, 2003. doi:10.1145/872035.872044.

23 Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proc. of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 2002)*, pages 24–33, 2002. doi:10.1145/570810.570814.

24 Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proc. of the 4th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, pages 267–278, 2003. doi:10.1145/778415.778447.

25 Frank Thomson Leighton, Bruce M. Maggs, Abhiram G. Ranade, and Satish Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17(1):157–205, 1994. doi:10.1006/jagm.1994.1030.

26 Wolfgang Mulzer and Max Willert. Compact routing in unit disk graphs. In *Proc. of the 31st International Symposium on Algorithms and Computation (ISAAC 2020)*, pages 16:1–16:14, 2020. doi:10.4230/LIPIcs.ISAAC.2020.16.

27 Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985. doi:10.1093/comjnl/28.1.5.

28 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proc. of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 2001)*, pages 1–10, 2001. doi:10.1145/378580.378581.

29 Ge Xia. The stretch factor of the Delaunay triangulation is less than 1.998. *SIAM Journal on Computing*, 42(4):1620–1659, 2013. doi:10.1137/110832458.

30 Chenyu Yan, Yang Xiang, and Feodor F. Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Computational Geometry: Theory and Applications*, 45(7):305–325, 2012. doi:10.1016/j.comgeo.2012.01.015.

## A Lower Bound Without Global Communication

The counterexample in Figure 4 (which follows the arguments of [23]) demonstrates that it is impossible to set up a compact routing scheme with constant stretch in polylogarithmic time when just relying on the unit-disk graph, even if it does not have radio holes and the geometric location of the destination is known: Suppose the destination is in the center. With a wheel of $\Theta(\sqrt{n})$ spikes of $\Theta(\sqrt{n})$ length each, no node on the wheel can guess the right spike with probability better than $\Theta(1/\sqrt{n})$, so routing information is required for a constant stretch, but in order to compute the information needed for a constant stretch, the starting point $w$ of the spike leading to the destination in the center needs to be identified, which requires $\Omega(\sqrt{n})$ communication rounds.

▶ **Theorem 24.** *There is no deterministic (randomized) distributed algorithm that can, within $o(\sqrt{n})$ rounds, compute a compact routing scheme that achieves (expected) $o(\sqrt{n})$ stretch when only communicating over the unit-disk graph. This claim holds even when the algorithm can use geometric information*[6] *and the unit-disk graph has no radio-holes.*

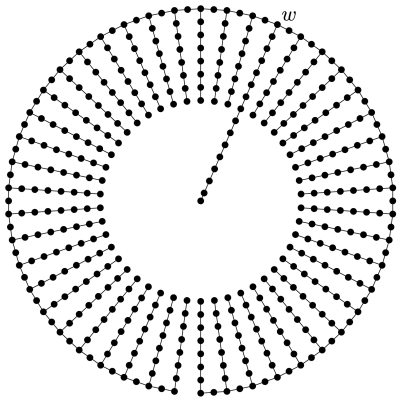



**Figure 4** Lower bound graph (slightly adapted from Figure 8 in [23]).

## B Proof of Theorem 16

**Proof of Theorem 16.** Given some $t \in V$, the label $\ell_t$ of $t$ is the concatenation of the label $\ell_{\Gamma,t}$ of the grid node which is in the same cell as $t$ and $\mathrm{ID}(t)$. Hence, the labelling $\ell_G$ requires $O(\log n)$ bits, given that the same is true for $\ell_\Gamma$. The information required to compute $\rho_G(v, \ell_t)$ is composed of the knowledge of neighbors of $v$ in $G$ which includes the representative of the cell of $v$ (due to Lemma 3) and the information required to evaluate $\rho_\Gamma(v, \ell_{\Gamma,t})$. Since nodes know their neighbors already as part of the problem input (local network equals the routing graph), we do not regard this as additional routing information. The information to evaluate $\rho_\Gamma(v, \ell_{\Gamma,t})$ is $O(1)$ bits by our presumption.

We continue with the correctness and the stretch of a path implied by $\rho_G$. Let $s \neq t \in V$ be the current node and the target node respectively. Consider the case that $\|s - t\| \leq 1$, i.e., the nodes are adjacent. Then, according to Definition 15 rule (1) the packet is delivered directly to $t$ which constitutes a correct and exact path.

[6] i.e., each node knows and can communicate its own location, the location of its neighbours, and each source will be given the location of its destination before routing.

Consider the case that $\|s-t\| \geq 1$. Let $g_s, g_t \in V_\Gamma$ and $r_s, r_g$ be the respective grid nodes and representatives of the cells of $s, t$. Let $\Pi^*$ be the optimal $g_s$-$g_t$-path in $\Gamma$ implied by $\rho_\Gamma$. Then the path implied by $\rho_G$ equals $\Pi_{s,t} = \{\{s, r_s\}\} \cup \Pi_{r_s,r_t} \cup \{\{r_t, t\}\}$ from Definition 9, where $\Pi_{r_s,r_t} := \bigcup_{\{g,g'\}\in\Pi^*} \Pi_{\{g,g'\}}$ and $\Pi_{\{g,g'\}}$ is the representation of the grid edge $\{g, g'\} \in \Pi^*$. This is due to rules (2),(3) and (4).

By Theorem 10, we have that $\text{dist}(\Pi_{s,t}) \leq |\Pi_{s,t}| \leq 36 \cdot \text{dist}_G(s, t)$, implying a stretch of $O(1)$.

The runtime of pre-computing $\mathcal{R}_G$ amounts to that of computing a representation $R$ of $\Gamma$, which takes $O(1)$ rounds due to Lemma 7. Note that in all four cases of the routing function $\rho_G$ can be evaluated locally using the representation $R$ of $\Gamma$ from the pre-computation step, information about local neighbors in $G$ and (local) evaluations of $\rho_\Gamma$. ◀

# C Additional Technical Details for Section 4

We provide the missing proof of Lemma 18 and additionally give an overview on some techniques for hybrid networks that are used by our tree labelling algorithm in Section 4. A more detailed description of these techniques can be found in [11, 16, 15].

## C.1 Proof of Lemma 18

**Proof of Lemma 18.** The runtime of $O(1)$ rounds is clear, as each node $v$ only needs to communicate for one round with its southern neighbor $v_S$ in the portal. We provide arguments on why the construction is a tree. Since the cell polygon $P'$ is simple (Lemma 5), the cells of vertices in a portal connect two points on the same polygonal boundary of $P'$. Thus, removing these cells disconnects $P'$ and therefore removing the vertices of a portal from $\Gamma$ disconnects $\Gamma$. This means the portal graph is acyclic, i.e., a tree. Since $\Gamma$ is connected, the portal graph is connected as well. ◀

## C.2 Pointer Jumping

We show how to construct a network with diameter $O(\log n)$ in time $O(\log n)$ out of a simple line graph $L$ with $O(n)$ nodes. Assume each node $v \in L$ knows its left and right neighbor in the line (except for the left- and rightmost node, who only know one neighbor). We let the nodes of $L$ generate *shortcut edges* via *pointer jumping*: In the first round, each node $v_i \in L$ that has two neighbors $v_{i-1}, v_{i+1} \in L$ establishes the edge $\{v_{i-1}, v_{i+1}\}$. Whenever in each subsequent round, a node $v \in L$ receives two new shortcut edges $\{u, v\}, \{v, w\}$ in the previous round, $v$ generates another shortcut edge $\{u, w\}$. It is easy to see that after $O(\log n)$ rounds, no further shortcut edges are created and the resulting structure has diameter $O(\log n)$, thus implying the following lemma.

▶ **Lemma 25.** *Given a line $L$ of $O(n)$ nodes, setting up additional edges to obtain a structure $L^+$ with diameter $O(\log n)$ and degree $O(\log n)$ takes $O(\log n)$ rounds.*

Performing pointer jumping on each of our portals in the portal tree, the grid nodes within each portal $\mathcal{P}$ are able to set up a structure on which they can quickly broadcast information to all grid nodes within $\mathcal{P}$. By doing so, we immediately obtain the following lemma.

▶ **Lemma 26.** *Any $O(\log n)$-bit message can be broadcast among all grid nodes within a single portal $\mathcal{P}$ in $O(\log |\mathcal{P}|)$ rounds.*

## C.3 Rooting Trees of Arbitrary Depth

Given a tree $T$ of $n$ nodes with arbitrary depth and constant node degree, we show how to root $T$ at the node $s$ with minimum identifier, such that every node in $T$ is aware of its parent node. To do so, we adapt the well-known *Euler tour* technique to a distributed setting. Every node $v \in T$ with neighbors $v(0), \ldots, v(\deg(v)-1)$ (sorted in ascending order by their identifiers) simulates a virtual node $v_i$ for each of its neighbor $v(i)$. We now connect all virtual nodes to a simple cycle $C$ as follows. For every node $v_i \in C$, there is an edge $(v_i, u_j) \in C$ such that $u = v((i+1) \mod \deg(v))$ and $v = u(j)$. Therefore, each virtual node $v_i$ that belongs to the node $v$ with identifier $id(v)$ is able to introduce itself to its predecessor in $C$ by sending its *virtual identifier* $\widetilde{id}(v_i) := id(v) \circ i$ for all $i \in [\deg(v)]$, where $\circ$ denotes the concatenation of two binary strings and $[k] = \{0, \ldots, k-1\}$. Since each node simulates only a constant number of virtual nodes, the number of virtual nodes in the cycle $C$ is $O(n)$.

We first describe how to determine the virtual node $s_i$ with minimal virtual identifier in $O(\log n)$ rounds.[7] Note that the node $s$ simulating $s_0$ is then the node with minimal identifier. Consider the cycle $C$ of virtual nodes and denote the edges of the cycle as *level*-1 edges. Our algorithm works in multiple iterations. Initially, each virtual node $v$ stores its own virtual identifier $\widetilde{id}(v)$ in some variable $v.I$. In the first iteration, each virtual node $v$ does the following. In the first step, $v$ sends $v.I$ to its left neighbor in the cycle[8]. Upon receipt of a virtual identifier $v.I$, each node $u$ updates its variable $u.I$ to $v.I$ in case that $v.I < u.I$. In the next step, each virtual node $v$ introduces its left neighbor $v_l$ to its right neighbor $v_r$ to create the edge $\{v_l, v_r\}$, a level-2 edge. In each subsequent iteration, say the $i$-th iteration, each node $v$ first sends $v.I$ along with its own identifier via *all* of its level-$j$ edges (for all $j \in \{1, \ldots, i\}$) and then creates level-$(i+1)$ edges, using its level-$i$ edges created in the previous iteration. Note that after the $i$-th iteration, $2^i$ nodes are aware of $v$'s virtual identifier and thus have stored $v$'s virtual identifier in their variables $u.I$, in case $v$'s virtual identifier is the minimal virtual identifier among all of these nodes. We proceed in this manner until a virtual node $v$ has received its own virtual identifier from its right neighbor in some iteration, as in this case all nodes have received $v$'s virtual identifier. This happens at the node $s_0$ with minimum virtual identifier after $O(\log n)$ rounds, because $C$ contains $O(n)$ virtual nodes. Thus, all that is left to do is to let $s_0$ announce itself as the root of the tree by broadcasting a message on the cycle with the generated shortcuts, indicating the termination of the algorithm and announcing itself as the node with minimum virtual identifier. This takes another $O(\log n)$ rounds. Then, each virtual node is now aware of the node $s_0$ with minimum virtual identifier and therefore also of the node $s$ with minimum identifier.

Now we want to root the tree $T$ at $s$. The virtual node $s_0$ starts broadcasting its virtual identifier via all of its outgoing edges to the left (including all of the generated shortcuts from before). During this broadcast, we keep track of the traversal distance of the message to be broadcasted, such that each virtual node is able to determine how many hops it is away from $s_0$ in the cycle. A real node $v$ can now determine its parent in the tree $T$ by looking at its virtual node with minimum traversal distance to $s_0$. Let this node be the node $v_i$ and let $u_i$ be the predecessor of $v_i$ in the cycle $C$. Then it is easy to see that $u$ is the parent node of $v$ in $T$, resulting in $T$ getting rooted at $s$ and implying the following lemma.

[7] The virtual node with minimal virtual identifier $\widetilde{id}(s_i) = id(s) \circ i$ is the node $s_i$ with $id(s) \leq id(v)$ for all $v \in T$ and $i = 0$.

[8] The nodes may have different perceptions on which direction is left, but this is of no concern for our algorithm.

▶ **Lemma 27.** *Let $T$ be a tree of $n$ nodes with constant node degree. $T$ can be rooted at the node $s$ with minimal identifier within $O(\log n)$ rounds.*

## C.4 Depth-First Search on Trees

Given a rooted tree $T$ of $n$ nodes with arbitrary depth and constant node degree, we compute for each node $v \in T$ the preorder number $l_v \in \mathbb{N}$ according to a depth-first search (DFS) of $T$. Let $s$ be the root of $T$. As the first step, we perform the distributed Euler-Tour technique described in the previous section with the exception that the virtual node $s_0$ refrains from introducing itself to its predecessor. It is easy to see that this results in the virtual nodes being arranged in a simple line $L$ instead of a cycle.

Next, we apply the pointer jumping technique from Lemma 25 to transform $L$ into a structure $L^+$ with diameter $O(\log n)$. Through a single broadcast from the leftmost node $s \in L^+$, we are now able to compute a number $l'_u$ for a virtual node $u$ indicating the number of (real) nodes $v$ for which at least one of $v$'s virtual nodes is left of $u$ on the line $L$. Each real node $u$ then sets $l_u$ to the minimum value out of all $l'_u$ values of its virtual nodes, which corresponds to $u$'s position in the DFS.

▶ **Lemma 28.** *Let $T$ be a rooted tree of $n$ nodes with constant node degree. A DFS on $T$ where each node $v \in T$ is assigned its number $l_v \in \mathbb{N}$ in the DFS can be computed in $O(\log n)$ rounds.*

## C.5 Computing the Maximum Preorder Number in a Rooted Tree

Assume we are given a rooted tree $T$ of $n$ nodes with arbitrary depth and constant node degree in which every node $v \in T$ possesses a preorder number $l_v \in \mathbb{N}$ according to a DFS. We compute for each node $v \in T$ the maximum preorder number possessed by a node in $v$'s subtree, i.e., we compute $r_v = \max\{l_u \in \mathbb{N} \mid u \in T(v)\}$, where $T(v)$ is the subtree of $T$ with $v$ as the root.[9]

Before we describe our algorithm, we let the nodes compute an upper bound of $\log n$, i.e., some value $d = O(\log n), d \geq \log n$ as follows. We compute the line $L$ via the Euler-tour described earlier on the rooted tree $T$ and apply Lemma 25 on $L$ to obtain the structure $L^+$. Then we perform a broadcast from the rightmost node $u$ in $L^+$ to the leftmost node $s_0$ in $L^+$, where each message generated by the broadcast contains a counter that is incremented by 1 once the message is forwarded. The node $s_0$ then maintains a variable $d$ that contains the maximum counter received by $s_0$. Since $L^+$ has diameter $O(\log n)$, the broadcast finishes after $O(\log n)$ rounds. Once the broadcast is finished, it is easy to see that $d = O(\log n)$. The node $s_0$ then broadcasts $d$ to all nodes in $L^+$, such that after another $O(\log n)$ rounds, each node knows $d$. Observe that $d \geq \log D(T)$, where $D(T)$ is the depth of the tree $T$.

We are now ready to describe the algorithm for computing the values $r_v$ for each node $v \in T$. Initially, each node $v$ sets $r_v$ to $l_v$. Denote the edges of $T$ as *level*-0 edges. The algorithms performs $i = 1, \ldots, d$ iterations, each iteration needing $O(1)$ rounds. Iteration

[9] A more general approach to this problem is presented in [16, Lemma 4.12], where the goal is to compute the value of a distributive aggregate function for each node $v$'s own subtree. An aggregate function $f$ is called *distributive* if there is an aggregate function $g$ such that for any multiset $S$ and any partition $S_1, \ldots, S_\ell$ of $S$, $f(S) = g(f(S_1), \ldots, f(S_\ell))$. Classical examples are MAX, MIN, and SUM. However, due to the generality of $f$, the authors had to make use of randomization, which results in a runtime of $O(\log n)$, w.h.p. for their algorithm. We present a deterministic $O(\log n)$-algorithm that is specifically tailored to the MAX function in this section.

$i$ works as follows at each node $v \in T$. First, if $v$ has a level-$(i-1)$ edge going up in the tree to some node $u$, then $v$ sends $r_v$ to $u$. Upon receipt of a value $r_w$ from node $w$ in the previous step, $v$ updates $r_v$ by setting $r_v \leftarrow r_w$ and *marks* the edge $\{v, w\}$.[10] As the final step of the iteration, $v$ checks whether it has a marked edge $\{v, u\}$ going up the tree and a marked edge $\{v, w\}$ going down the tree. If that is the case, $v$ creates a level-$i$ edge $\{u, w\}$ by introducing $u$ to $w$ and vice versa. If not, then $v$ marks itself as *ready*.

Let $T(v)$ be the subtree of $T$ with $v$ as the root and let $w \in T(v)$ be the leaf node with maximum preorder number. Consider the unique path $P$ up the tree from $w$ to $v$ in $T(v)$. It is easy to see that our algorithm transfers the preorder number $l_w$ to all nodes on this path within $\lceil \log k \rceil$ iterations, where $k$ is the length of $P$, because in each iteration $i$, new level-$i$ shortcuts are added to the nodes on the path in a manner similar to the pointer-jumping approach from Section C.2. Therefore, once a node $v$ has marked itself as ready in iteration $i$, $v$ has received the desired value for $r_v$ in iteration $i$. As each node $v \in T$ performs the algorithm in parallel, each node $v$ has determined $r_v$ after at most $d = O(\log n)$ iterations (recall that $d \geq \log D(T)$). Note that the node degree for each node $v$ does not exceed $O(\log n)$ throughout the algorithm, as in each iteration, $v$'s degree increases by at most 2.

We obtain the following lemma.

▶ **Lemma 29.** *Let $T$ be a rooted tree of $n$ nodes with constant node degree in which every node $v \in T$ possesses a preorder number $l_v \in \mathbb{N}$ according to a DFS on $T$ starting at its root. Each node $v \in T$ can compute the value $r_v = \max\{l_u \in \mathbb{N} \mid u \in T(v)\}$, where $T(v)$ is the subtree of $T$ with $v$ as the root, within $O(\log n)$ rounds.*

[10] Note that in the first iteration ($i = 1$), a node $v$ receives a value $r_w$ from each of its child nodes $w$. It then just sets $r_v$ to be the maximum value out of all received values $r_w$. It is easy to see that $v$ receives at most one message in any subsequent iterations in this step.