Efficient Communication Strategies for Ad-Hoc Wireless Networks (Technical Report)

Micah Adler*
Department of Computer Science
University of Toronto
10 King's College Rd.
Toronto ON, M5S3G4, Canada

Christian Scheideler†
Heinz Nixdorf Institute and
Dept. of Math. and Computer Science
Paderborn University
33095 Paderborn, Germany

January 23, 2000

Abstract

An *ad-hoc* wireless network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. This type of network is of great importance in situations where it is very difficult to provide the necessary infrastructure, but it is a challenging task to enable fast and reliable communication within such a network. In this paper, we model and analyze the performance of so-called *power-controlled* ad-hoc wireless networks: networks where the mobile hosts are able to change their transmission power. We concentrate on finding schemes for routing arbitrary permutations in these networks. In general, it is NP-hard even to find a $n^{1-\epsilon}$-approximation for any constant ϵ to the fastest possible strategy for routing a given permutation problem on n mobile hosts. However, we here demonstrate that if we allow ourselves to consider slightly less general problems, efficient solutions can be found.

We first demonstrate that there is a natural class of distributed schemes for handling node-to-node communication on top of which online route selection and scheduling strategies can be constructed such that the performance of this class of schemes can be exploited in a nearly optimal way for routing permutations in any static power-controlled ad-hoc network. We then demonstrate that if we restrict ourselves to the important case of routing between nodes distributed randomly in a Euclidean space, we can route in a time that is asymptotically optimal for any routing scheme.

[*] Email: micah@cs.toronto.edu. Supported by an operating grant from the Natural Sciences and Engineering Research Council of Canada, and by ITRC, an Ontario Centre of Excellence. This research was conducted in part while he was at the Heinz Nixdorf Institute Graduate College, Paderborn, Germany.

[†] Email: chrsch@uni-paderborn.de. Supported in part by the DFG-Sonderforschungsbereich 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen" and by the EU ESPRIT Research Project 20244 (ALCOM-IT). Part of the research was done while staying at the Weizmann Institute, supported by a scholarship from the Minerva foundation.

1 Introduction

At the moment a dramatic growth in wireless networks can be observed. Mobile hosts and wireless networking hardware are becoming widely available, and extensive work has been done recently in integrating these elements into traditional networks such as the Internet. There are, however, important scenarios in which no fixed wired infrastructure such as the Internet is available, either because it may not be economically practical or physically possible to provide the necessary infrastructure or because the expediency of the situation does not permit its installation (for instance, networks formed by satellites, ships or airplanes, or networks connecting rescue teams in case of an earthquake or flood). In such situations, a collection of mobile hosts with wireless network interfaces may form a temporary network without the aid of any established infrastructure or centralized administration. This type of wireless network is known as an *ad-hoc* network [19, 21]. Ad-hoc wireless networks have been recognized as an important form of wireless network. The IEEE 802.11 recommended standard for wireless LANs, for instance, requires an access protocol to have the ability to support ad-hoc networking [7].

Ad-hoc networks can be divided into two categories: *simple*, where the mobile hosts use fixed transmission powers, and *power-controlled*, where the mobile hosts are able to change their transmission power. Power-controlled networks have several advantages over simple ad-hoc networks:

- Contention among the hosts can be significantly reduced.
- Energy consumption can be significantly reduced.
- Security of transmissions can be increased.

The inflexibility of simple ad-hoc networks can be best demonstrated by considering human conversation. In everyday life we naturally use our ability to control our voice level in order to communicate with other people. Imagine the problems caused if everyone could only speak with one voice level!

In this paper, we develop strategies for efficient communication in power-controlled ad-hoc networks. The most important communication primitives for these networks are broadcasting and point-to-point communication. We here concentrate on coordinating sets of point-to-point communication problems that can be described as permutations. The majority of traffic on the Internet today consists of point-to-point communication, and it is reasonable to assume that traffic on ad-hoc wireless networks will be similar. However, to the best of our knowledge, no strategy or algorithm has been presented so far that can solve permutation routing problems by a method better than a sequence of broadcasts. This is a waste of communication resources, which is undesirable, since bandwidth in wireless networks is very scarce compared to wired networks. Instead, we here demonstrate much more time- and resource-efficient techniques that are close to optimal in certain situations.

1.1 Previous Work

Despite the advantages of allowing variable transmission powers, much of the existing work on ad-hoc networks considers hosts that have a fixed transmission range. However, the idea of allowing mobile hosts to vary their transmission power and transmission rate is already used in so-called power-controlled CDMA systems, where the base station can direct mobiles to reduce their power and data rate to reduce interference and allow more users on the system. This approach is employed in TIA IS-95 with respect to the time-varying voice activity on cellular voice channels [22, 19]. The advantage of adjusting the power and data rate of mobile data users to the current interference level has already been studied experimentally (see, *e. g.*, [22]).

To our knowledge, no results for an abstract model exist on how to select routes or schedule the transmissions of messages for power-controlled ad-hoc networks. However, connectivity problems have been studied for both simple ad-hoc networks [31], as well as power-controlled ad-hoc networks [39, 25]. Kirousis *et al.* [25] present a polynomial time algorithm for finding the minimum cost transmission power assignment that maintains connectivity for arbitrarily distributed collinear points, where the cost of an assignment is defined as total power usage. They also provide a 2-approximation algorithm for finding the minimum cost assignment in the case where the hosts are arbitrarily distributed in three dimensions.

The remainder of the previous work we discuss has studied only simple ad-hoc wireless networks. A common model for such networks is called the *packet radio network* (or PRN) model. A PRN is represented as an undirected graph $G = (V, E)$, where two nodes i and j are connected if and only if i is within the transmission radius of j and vice versa. One time step is defined as the time it takes to transmit a packet. Since it is usually assumed that all nodes can only transmit at one frequency, a *transmission conflict* occurs at node i if two of its neighboring nodes want to transmit at the same time. Communication problems have been studied by many authors for both the case that a conflict can be detected and the case that it cannot be detected.

Studies of multi-hop PRNs have mainly concentrated on broadcasting problems. Lower bounds on this problem are considered in [1, 26, 3, 20, 6]. A number of papers also deal with constructing efficient broadcast protocols [9, 41, 11, 3, 17, 36]. Let n denote the size, D denote the diameter, and Δ denote the maximum degree of a given PRN. Bar-Yehuda *et al.* [3], for instance, present a randomized distributed broadcast protocol that completes in expected $O(D \log n + \log^2 n)$ steps. Gaber and Mansour [17] present a centralized deterministic protocol that works in time $O(D + \log^5 n)$. Point-to-point communication problems have also been studied, for instance, by Bar-Yehuda *et al.* [4], who present a randomized distributed protocol that performs k point-to-point transmissions in $O((k + D) \log \Delta)$ time steps, on average.

Another problem that has been studied is how to schedule transmissions in a PRN to enable neighboring nodes to successfully exchange information. For early work see, *e. g.*, [10, 12, 32]. Recent results can be found in [8, 5]. Also, the problem of selecting routes and updating routes when mobile hosts move has been considered. See, *e. g.*, [28, 23, 16] for different route selection strategies and further references. Similar problems have been studied for various dynamic network models. See Dolev *et al.* [15] for a survey of results in this area.

1.2 The Model

It is a challenging task to develop a model for wireless communication that is simple enough to enable the design and analysis of algorithms, but is also detailed enough to ensure that efficient algorithms derived in this model also perform well in practice. As in previous approaches such as the PRN model, we here choose a high level of abstraction. We also note at the end of this section additional details that could be added to the model, and how they would affect our results.

We model power-controlled ad-hoc networks with the help of the following graph:

Definition 1.1 *Let a* transmission graph $G = (V, \tau)$ *be a complete undirected graph with node set* V *and edge labels determined by the function* $\tau : V \times V \to \mathbb{R}^+$. *For any edge* $\{u, v\}$, $\tau(\{u, v\})$ *represents the lowest transmission power that allows* u *to send a message to* v *and vice versa.*

In this paper, we only deal with *static* transmission graphs, *i. e.*, the situation where the positions of the mobile hosts and the environment do not change. Although dynamic transmission graphs are more interesting for many wireless networking problems, static graphs are useful for scenarios where a large amount of communication can be performed between updates to the transmission graph. This is the case for fixed wireless networks in buildings or connecting several buildings, or low-mobility radio networks such as networks formed by ships. A thorough understanding of static graphs also provides a starting point for the dynamic case, for example, by providing a measure with which to compare performance.

The remainder of the model is defined as follows:

- A packet needs one time step to make a single hop in the network, regardless of the distance to the destination node. In many situations it is more time- and resource-efficient for a message to perform a sequence of hops instead of one single hop to its final destination. The sequence of nodes used for the hops is called the *route* of a message.
- Only one frequency is available for transmitting packets, which implies that:
  - A node can send out at most one packet at a time.
  - If a node v attempts to send a packet with transmission power t, then all nodes that need less than αt power to receive a packet from v are *blocked*, where $\alpha > 1$ is some fixed constant. Any information transmitted to a blocked node is not received. Note that this means that it is possible for no transmission during a single time step to be successful.
- A transmission conflict cannot be detected by the sending node.
- All nodes work in a synchronized way. (For simplification reasons, this paradigm is commonly used in the design and analysis of wireless communication strategies that are robust to slight differences in the speeds at which the nodes operate [3, 18, 37].)

As noted above, the level of abstraction of this model is high. We here address three of the above assumptions: (a) all hops in the network take the same amount of time, (b) the cause of messages being blocked and (c) the maximum transmission power of the mobile stations is unlimited. For (a), in practice, signal propagation time does cause the time required for a hop to depend on the distance traveled. However, even though incorporating this into our model does make our proofs slightly more complicated, it does not change any of our results in a qualitative manner. For (b), we point out that in practice, a message m can be blocked even when no single node is transmitting with enough power to block m, but the cumulative effect of the transmissions by several nodes might block m. The relevant measure is actually the strength of the interference caused by all possible sources of signals (the so-called *signal to interference ratio* or SIR) and not only one. See, for instance, the model developed by Ulukus and Yates [39]. However, in practice it turns out that only signals with strength above some threshold value contribute to blocking a node, since all other signals tend to cancel each other out rather than to add up, or may even be insignificant compared to the white Gaussian noise that is always present. Furthermore, incorporating the SIR into our model in the manner proposed by [39] makes our proofs considerably more complicated, but has no qualitative effect on the results of Chapter 2 and only an insignificant qualitative effect on the results of Chapter 3. For (c), limitations on the transmission power can be easily incorporated in Chapter 2 by restricting the class of schemes considered accordingly. This also does not present any difficulties for the strategies considered in Chapter 3, since they only require communication between nodes of close proximity.

1.3 On the Hardness of Routing in Ad-Hoc Networks

Several authors have already presented NP-hardness results for various wireless communication problems, such as the problem of finding an optimal broadcast schedule [9] or scheduling transmissions in the case where every node wants to send a message to one of its neighbors [38]. With a similar reduction from Chromatic Number, the following result can be shown.

Remark 1.2 *Given a transmission graph and a collection of n routes, even finding a schedule with a runtime that is within a factor of $n^{1-\epsilon}$ of optimal, for any constant $\epsilon > 0$, is NP-hard.*

Hence, finding efficient communication strategies for wireless networks is much harder than for networks with fixed node-to-node connections, even if a fixed collection of routes is already given along which the packets have to be sent. The hardness result above also holds if we do not constrain ourselves to specific routes. The reduction is similar.

1.4 New Results

Nearly nothing is known about how permutation routing problems can be handled in wireless communication networks, and the hardness results above do not look very promising. However, we here show that if we allow ourselves to consider slightly less general problems, efficient solutions can be found. We demonstrate this with two approaches: first we consider the effect of restricting the class of routing strategies allowed, and then we restrict the class of allowed transmission graphs. Both restrictions we consider are natural and lead to permutation routing strategies that are close to optimal for their respective classes.

The class of strategies considered is defined by separating the routing problem into three layers: one layer that handles the scheduling of node-to-node transmissions of packets (we follow the experimental literature and refer to this as the *medium access control*, or MAC, layer [19]), one layer that is responsible for finding efficient routes for the packets, (called the *route selection* layer) and one layer for selecting which packets are to be transmitted next by the nodes (called the *scheduling* layer). We introduce a class of protocols for the MAC layer called *local probabilistic control* protocols. These protocols allow every node to make scheduling decisions in a randomized fashion without requiring any coordination between the nodes. Furthermore, they ensure that each attempted transmission is successful with constant probability.

On top of these protocols we develop schemes for the other two layers that almost optimally exploit the power of the MAC layer protocols for *any* transmission graph. For this we introduce a parameter $R(G, S)$, similar to the routing number of [2, 30], that represents a lower bound on the expected time required to route a random permutation in G using the MAC scheme S. We present online strategies for the route selection layer and scheduling layer that are able to route any permutation in G using S in time $O(R(G, S) \cdot \log n)$, and in many cases even in time $O(R(G, S) \cdot (\log \log n)^2)$, where n is the size of G. The techniques developed include efficient simulations of strategies for deterministic communication models by probabilistic communication models. This might be of independent interest.

We then study the effect of restricting the class of allowed communication graphs to those representing mobile hosts distributed in a Euclidean space. We assume that n points are distributed uniformly and independently at random in a *domain space*, a convex region of $\mathbb{R}^d$ that is known *a priori*. This random distribution models the situation where each of the nodes moves within the domain space independently and with equal probability in any direction. In the case where the domain space is a 2 dimensional square, we show that we can route any permutation online in time $O(\sqrt{n})$ with probability at least $1 - O(1/n)$. We also demonstrate that the techniques for a square domain space can be extended to an arbitrarily shaped convex domain space.

Furthermore, we show that in the scenario considered, it is not possible to use the ability to transmit over long distances to perform permutation routing faster using wireless communication than it is using an array: we prove a lower bound demonstrating that when the domain space is a 2 dimensional square, the time required to route a random permutation is $\Omega(\sqrt{n})$ with probability at least $1 - O(1/n)$. We also show that the result for routing in an arbitrarily shaped convex domain space is within a constant factor of optimal.

Finally, we note that our routing strategies developed for the two approaches above (*i. e.*, restricting the class of MAC layer protocols or the class of allowed transmission graphs) are not limited to supporting the exchange of messages, but can also be used to perform efficient distributed computations in ad-hoc networks such as sorting and matrix multiplication.

2 A Universal Framework for Wireless Communication

In this section we propose strategies that are universally applicable in a sense that they can be reliably used for arbitrary static ad-hoc networks and arbitrary routing problems. Our approach basically consists of three layers:

(1) the medium access control layer (or MAC layer),

(2) the route selection layer, and

(3) the scheduling layer.

The MAC layer is responsible for enabling node-to-node communication. The task of the route selection layer is to choose suitable routes (*i. e.*, suitable sequences of nodes to visit) for the packets, and the task of the scheduling layer is to resolve conflicts between packets that want to be transmitted by the same node at the same time step.

A communication scheme has to fulfill several demands to be strictly online, such as: The routes should be chosen independently of each other, and the nodes should be able to decide locally when and how often to try to get access to other nodes within some given time interval, and which packet to prefer if several packets want to be sent out by a node at the same time step. On the other hand, the strategy should (at least in important scenarios) ensure that the node-to-node communication, the selection of the routes and the scheduling are as efficient as possible. As we have seen in the previous section, to fulfill even a part of these demands is impossible in general (unless $P = NP$). However, we will show that if we restrict ourselves to considering only communication strategies that use the class of MAC layer protocols described below, suitable strategies can be constructed that (nearly) fulfill these demands.

2.1 Strategies for the MAC layer

In this section, we propose a class of local probabilistic control protocols for the MAC layer (also called *LPC schemes* in the following) that allow all nodes to operate independently of each other without blocking any other node too much by access trials. Before we define this class, let us first give a brief overview of the most commonly used MAC layer strategies and describe, why they cannot be used efficiently in ad-hoc networks.

2.1.1 Currently used MAC layer protocols

In todays wireless communication technologies, there are basically three different ways of transmitting signals: FDMA, TDMA and CDMA.

In FDMA (frequency division multiple access), signals can be transmitted simultaneously by using different frequencies. In TDMA (time division multiple access), signals occupy the same frequency band, but are separated by using non-overlapping time slots for transmissions. CDMA (code division multiple access) is a technique that contains both FDMA and TDMA. The basic idea of CDMA is that the senders use codes for their transmissions that are orthogonal to each other, *i. e.*, their cross-correlation is 0. Thus, with the help of a suitable correlator a receiver is able to filter out a signal from a specific sender without getting interference from others.

All current access schemes use either TDMA or CDMA (both often in combination with FDMA to increase the bandwidth and/or reduce interference). Whereas TDMA is used in most of todays standards such as DECT, GSM (Europe), PDC (Japan) and IS-54 (USA), CDMA is regarded as the access mode for future wireless communication standards such as UMTS (universal mobile communication system), currently under development in Europe, or IMT-2000, currently under development by the ITU, the international body for communication standards.

Both TDMA and CDMA are not useful for ad-hoc wireless networks since they are very sensitive to even small differences in the speeds at which the mobile stations operate. When there is central control, for instance via a base station, this problem is solved with the help of a reference signal sent out by the base station at fixed time intervals.

In the special case of wireless LANs, most networks are based on carrier sense multiple access (CSMA), polling, and TDMA. These can also be found in the IEEE 802.11 recommended standard for wireless LANs.

In CSMA protocols, each node listens whether other nodes are transmitting. If not, it tries to send its message. If it recognizes a collision with another message, it backs off for a random period of time before it tries again to send out its message. CSMA has been used with great success in the Ethernet. Unfortunately, this strategy is not applicable for ad-hoc networks, since a node cannot detect whether the destination of its message is currently blocked or not.

Hence, we propose a different class of MAC layer protocols for ad-hoc networks. The advantage of these protocols is that they seem to have the potential to adjust quickly to changing situations and therefore could support efficient communication also in dynamic ad-hoc networks.

2.1.2 A new class of MAC layer protocols

In this section we propose a class of simple local probabilistic control protocols for the MAC layer that allow each node to decide independently of the other nodes when to send a packet.

Let us start with describing how to perform a hop in an ad-hoc network. Suppose v tries to transmit a packet P to w. We say that the hop of P is *successful* if v knows that P reached w. Since in our model a transmission conflict cannot be detected by the sending node, a successful hop requires sending data without a conflict from v to w and from w to v. For our purposes it would suffice to have only two steps: Sending P from v to w, and sending an acknowledgement from w to v. We do not require w to know whether v knows that P reached it, since our route selection strategies will be based on choosing fixed routes. Once w receives the packet it can immediately start to send it further along its route.

Note that we do not consider here the possibility of transmission errors or node failures. In these cases, four communication steps may be used for a hop to ensure a correct transmission, and limits for the number of transmission attempts may be used to detect node failures.

Now, consider any transmission graph $G = (V, \tau)$. We define an LPC scheme for G to be a MAC layer protocol that can be described by an *access matrix* $P = (p_{v,w})_{v,w \in V}$ with $p_{v,w} \in [0, 1)$ for all $v, w \in V$. This access matrix is used in the following way:

Assume that some node v wants to send a packet Q to node w (which is allowed if $p_{v,w} > 0$). As long as v has not received an acknowledgment, v decides at each time step with probability $p_{v,w}$ to start a hop for Q to w.

For all $u, v \in V$ with $u \neq v$, let $b_{u,v} = \max_{w \in V}\{p_{v,w}\colon \text{a hop attempt from } v \text{ to } w \text{ blocks } u\}$. (Recall the definition of a node to be blocked in our model.) By convenience, we define $b_{u,u} = 0$. We demand for each LPC scheme that the $p_{v,w}$'s are set in such a way that for every $u \in V$, $\sum_{v \in V} b_{u,v} \leq 1/4$, *i. e.*, the probability that u is blocked at some time step is at most $1/4$ for any situation in which every node has at most one packet. This ensures that for any LPC scheme, no coordination among the nodes is necessary. Each node can make an arbitrary decision of where to try to send a packet, but we still have for all nodes $v, w \in V$ that

$$\Pr[\text{hop } v \to w \text{ not successful}] \leq \Pr[v \text{ blocked}] + \Pr[w \text{ blocked}] \leq 1/2 \; .$$

That is, a hop attempt has a success probability of at least $1/2$.

We want to use this class of schemes as a basic building block for the rest of our communication strategies. We will demonstrate that we can exploit the power of these schemes in a nearly optimal way for routing arbitrary permutations in *any* static ad-hoc network. Hence, possible inefficiencies are confined mostly to the MAC layer. An interesting open problem is finding more powerful classes of MAC layer protocols and developing efficient routing strategies on top of them.

2.1.3 Transforming the problem of routing in transmission graphs to routing in PCGs

The drawback of our schemes is that they are only guaranteed to work if every node tries a transmission only for one packet (*i. e.*, only one packet is *active*) at any time. Hence, if more than one packet is currently stored at a node, the others have to wait until the packet chosen by the node for hop attempts is successful. To see why this is a problem, consider the scenario where v_1 has $p_{1,2} = 1/10$ and $p_{1,3} = 1/1000$. If v_1 has to send a packet P_3 to v_3, it will (probably) require many attempts before P_3 is successfully sent. Say in the meantime, that v_1 receives another packet P_2, where P_2 needs to be sent to v_2. Forcing P_2 to wait until P_3 is successfully sent will probably force P_2 to wait at v_1 much longer than expected. On the other hand, giving P_2 priority over P_3 can lead to "starvation" of P_3 if many packets destined for v_2 arrive.

In order to avoid this drawback, we show in the following how to refine any LPC scheme to a scheme that allows several packets to be active at the same time, without reducing the $p_{v,w}$ by more than a constant. This transformation not only helps to make our schemes much more flexible, but also helps to interpret the problem of routing in transmission graphs using our LPC schemes as a routing problem in a network model called PCG (see Definition 2.2). This allows us to draw on the large body of literature in the field of interconnection networks to propose suitable path selection and scheduling strategies. The analysis of these strategies, however, is much more involved than previous analyses because of the probabilistic and non-uniform nature of our network model.

Consider any LPC scheme S, and let P be its access matrix. For each node v and $i \geq 1$, let the *i-zone* of v contain all nodes w with $p_{v,w} \in (\frac{1}{2^{i-1}}, \frac{1}{2^i}]$. For each i-zone, let us define the zone access probability for v as $q_{v,i} = \frac{1}{2^{i+2}}$. Furthermore, for all $u, v \in V$ let $b_{u,v}$ be defined as above. Assume that we allow now each node to have an active packet for every zone. Then the probability that node v is blocked by a hop attempt of a node $w \neq v$ is at most

$$\sum_{w \in V} \sum_{i \geq \lceil \log b_{v,w}^{-1} \rceil} \frac{1}{2^{i+2}} \leq \frac{1}{2} \sum_{w \in V} b_{v,w} \leq \frac{1}{8} \; .$$

Since for every pair of nodes $v, w \in V$ we require that $p_{v,w} \leq 1/4$ in order to ensure the bounds on the blocking probability of the original LPC scheme, the lowest i for which hop attempts to an i-zone are allowed is at least 2. Thus, the probability that node v is blocked by one of its own hop attempts is at most

$$\sum_{i \geq 2} \frac{1}{2^{i+2}} \leq \frac{1}{8} \; .$$

Hence, altogether the probability that a node is blocked using the zone access scheme is at most 1/4. Thus, the probability that a hop attempt is successful is at least 1/2, as desired.

Let us call the scheme S extended by zone accesses S_Z. Furthermore, let P_Z be the access matrix of S_Z with entries $p^Z_{v,w}$ for all $v, w \in V$. The construction above implies that, for all $v, w \in V$ with $p_{v,w} \in (\frac{1}{2^{i-1}}, \frac{1}{2^i}]$, $p^Z_{v,w} = \frac{1}{2^{i+2}} > 1/8 \cdot p_{v,w}$. Thus the following fact holds.

Fact 2.1 *For any LPC scheme S with access matrix P it holds that every entry of the access matrix P_Z of S_Z is at least $1/8$ the corresponding entry of P. However, S_Z allows one packet to participate in hop trials for each node and zone while guaranteeing that any attempted hop is still successful with probability at least $1/2$.*

In the following we show that for all communication strategies using a zone access scheme we can transform the problem of routing packets in a transmission graph (which is very difficult to handle, as we saw in the introduction) to the problem of routing packets in the following graph model (which, as we will see, is easier to handle).

Definition 2.2 *Let the* probabilistic communication graph *(or PCG in short) $G = (V, \varphi)$ be defined as a complete directed graph with node set V and edge labels determined by the function $\varphi : V \times V \to [0, 1]$. Every edge e can forward a packet in one time step, but only succeeds in doing this with probability $\varphi(e)$.*

Next we define how to transform transmission graphs to PCGs.

Definition 2.3 *Given a transmission graph $G = (V, \tau)$ and a zone access scheme Z with maximum zone number k, let the corresponding PCG be defined as $G_Z = (W, \varphi)$ with*

- $W = V \cup (V \times \{1, \ldots, k\})$,
- $\varphi(v, (v, i)) = \frac{1}{2^{i+2}}$ *for all* $v \in V$ *and* $i \in \{1, \ldots, k\}$ *(this represents the probability that a hop is attempted to the i-zone of v in G),*
- $\varphi((v, i), w) = 1$ *if w is in the i-zone of v and 0 otherwise for all* $v, w \in V$ *and* $i \in \{1, \ldots, k\}$*, and*
- $\varphi(v, w) = \varphi((v, i), (w, j)) = 0$ *for all* $v, w \in V$ *and* $i, j \in \{1, \ldots, k\}$.

Assuming the multi-port model for PCGs (*i. e.*, in each time step a transmission can be attempted for each edge), the following theorem holds.

Theorem 2.4 *Given any transmission graph G and zone access scheme Z it holds that, for any strategy on top of Z that can route some given routing problem $\mathcal{R}$ in G in expected time T, there is an offline strategy in G_Z that routes $\mathcal{R}$ in expected time at most $2T$.*

Proof. Given a strategy S on top of Z, let $\mathcal{T}_S$ denote the probability tree of all possible outcomes for sending the packets in G to their destinations, using S on top of Z. The root of $\mathcal{T}_S$ represents the starting point, and each path in $\mathcal{T}_S$ from the root to a leaf represents a possible outcome for S. Each outcome represents a protocol containing for each time step t, node v, and zone z the event whether

(1) no packet at v tried to use zone z at time t, or

(2) packet P at v tried to use zone z, but it randomly chose not to perform a hop, or

(3) packet P at v tried to perform a hop to node w in zone z, but failed due to a collision event, or

(4) packet P at v successfully performed a hop to node w in zone z.

Each time step of the protocol is represented by a node in $\mathcal{T}_S$.

Our aim is to contruct a (centralized) offline protocol for G_Z that simulates $\mathcal{T}_S$. Clearly, events of type 1 do not need any transformation (apart from, maybe, holding packets artificially back in the offline protocol). Events of type 2 can also be used without any change for G_Z, since the probability of attempting a hop is equal for both models. Since collisions of events cannot happen in G_Z, events of type 3 do not appear there. To incorporate them in the offline protocol, they are simulated in a way that the branching probabilities in $\mathcal{T}_S$ are preserved. Finally, any event of type 4 appearing in G can be taken over as it is to the offline protocol. As a hop takes two steps in G_Z instead of one step in G, the theorem follows. ∎

Hence, a lower bound for routing some given routing problem $\mathcal{R}$ in the PCG G_Z is also (up to a constant factor) a lower bound for routing $\mathcal{R}$ in the transmission graph G using Z. In addition, it is easy to see that for any analysis of a routing strategy, in which only probabilities for one node and zone at a time or expectations are considered for bounding the number of successful hops, any upper time bound for sending packets in the PCG G_Z also holds for sending the packets in the transmission graph G using Z. However, if probabilities for several nodes and/or zones are considered for the same time step, this is no longer true because of possible transmission conflicts. The proofs for our upper time bounds, presented in the following, will take care of this problem, so that all results presented in the following for PCGs also hold for transmission graphs using zone access schemes.

2.2 Path selection strategies for PCGs

In this section, we show how to select online (*i. e.*, in a distributed way) efficient path collections for routing arbitrary permutations in arbitrary PCGs. For this we need some notation. Let the *edge latency* L of a PCG G be defined as the maximum expected time $< \infty$ to cross an edge in G. Given a collection $\mathcal{P}$ of *simple* (*i. e.*, loop-free) paths in some PCG G,

- the *dilation* D of $\mathcal{P}$ is defined as the maximum over all paths in $\mathcal{P}$ of the sum of $1/\varphi(e)$ over all edges e used by it (that is, D denotes the maximum expected time a packet needs to traverse a path in $\mathcal{P}$), and
- the *congestion* C of $\mathcal{P}$ is defined as the maximum over all edges e of $1/\varphi(e)$ times the number of paths in $\mathcal{P}$ that cross it (that is, C denotes the maximum expected time spent at an edge e to forward all packets which contain e in their path).

In order to have a measure for the quality of the dilation and congestion of a path collection, we use the so-called *routing number*. This number is defined as follows (see, *e. g.*, [2, 30]):

Consider an arbitrary PCG $G = (V, \varphi)$ with N nodes. Let S_N represent the set of all permutations over $[N] = \{1, \ldots, N\}$. For a permutation $\pi \in S_N$, let $R(G, \pi)$ be the minimum possible expected number of steps required to route packets offline (*i. e.*, the whole routing problem is known to all nodes) in G according to π under the assumption that only one copy per packet is sent through the network. Then the *routing number* $R(G)$ of G is defined by $R(G) = \max_{\pi \in S_N} R(G, \pi)$. When there is no risk of confusion about the network G we will write R instead of $R(G)$. The routing number has the following nice property.

Theorem 2.5 *For any PCG G with routing number R and any routing strategy, the average, over all permutations, expected number of steps to route a permutation in G is bounded by $\Omega(R)$.*

Proof. The proof follows directly from a proof in [30]. ∎

Hence, asymptotically the routing number is not only an upper bound, but also a lower bound for the average permutation routing time using optimal routing strategies in G. This demonstrates that the routing number is a robust measure for the routing performance of graphs within our model. With the help of the routing number, we can prove Theorem 2.6. The main problem in the proof of this theorem is that a best possible path selection strategy for a PCG may not be a strategy using fixed paths, but maybe a strategy in which paths are chosen adaptively according to the outcome of transmission trials.

Theorem 2.6 *For any PCG G with routing number R and edge latency L and any permutation routing problem $\pi \in S_N$ in G, there exists a collection of simple paths with congestion and dilation at most $R + O(\sqrt{R\ell \log(R/\ell)}) = \Theta(R)$, where $\ell = \min[R, L]$.*

Proof. Consider any strategy $\mathcal{S}$ for sending packets in G to destinations prescribed by π. Let the random variable T denote the time the packets need to reach their destinations using $\mathcal{S}$. In order to have a bound on the edge latencies used by $\mathcal{S}$, we first show the following lemma.

Lemma 2.7 *If $\mathcal{S}$ makes use of edges with latency beyond $\mathrm{E}[T]$, $\mathcal{S}$ can be reduced to a strategy $\mathcal{S}'$ with expected runtime at most $2\mathrm{E}[T]$ that makes use of edges with latency at most $2\mathrm{E}[T]$.*

Proof. Consider the situation that $\mathcal{S}$ successfully uses an edge e with latency $\ell \geq 2\mathrm{E}[T]$, that is, at least one packet is sent along e. Since the expected routing time for this situation is at least ℓ, $\mathcal{S}$ can only successfully use such an edge with probability at most $\mathrm{E}[T]/\ell \leq 1/2$, because otherwise the overall expected routing time would exceed $\mathrm{E}[T]$. This means that with probability at least $1/2$ no edge successfully used by $\mathcal{S}$ has a latency of more than $2\mathrm{E}[T]$. Hence, the expected runtime for $\mathcal{S}$ restricted in a way that it is only allowed to make use of edges with latency at most $2\mathrm{E}[T]$ is at most $2\mathrm{E}[T]$, because otherwise the overall expected routing time would again exceed $\mathrm{E}[T]$. This completes the proof. ∎

In the following we therefore assume that $\mathcal{S}$ makes use of edges with maximum latency $L \leq 2\mathrm{E}[T]$. Let the random variables C and D be defined such that C denotes the congestion and D denotes the dilation of the path collection chosen by $\mathcal{S}$. Let $B = \max[C, D]$. In order to establish a relationship between B and T, we need a Chernoff-type bound for sums of geometrically distributed random variables.

Lemma 2.8 *Let $X_1, \ldots, X_n$ be $n \geq 1$ independent geometrically distributed random variables with parameters $p_1, \ldots, p_n \in [0, 1]$ (that is, $\Pr[X_i = j] = (1-p_i)^{j-1} p_i$ for all $j \in \mathbb{N}$). Let $X = \sum_{i=1}^n X_i$ and $p = \min[p_1, \ldots, p_n]$. Then it holds for any $0 \leq \epsilon \leq 1$ that*

$$\Pr[X \leq (1-\epsilon)\mathrm{E}[X]] \leq \mathrm{e}^{-\epsilon^2 \mathrm{E}[X] p/2} .$$

Proof. According to the Markov Inequality it holds that

$$\Pr[X \leq (1-\delta)\mu] \leq \mathrm{e}^{h(1-\delta)\mu} \cdot \mathrm{E}[\mathrm{e}^{-h \cdot X}] \qquad (1)$$

for any $h > 0$. Since $X_1, \ldots, X_n$ are independent, it follows that

$$\mathrm{E}[\mathrm{e}^{-h \cdot X}] \;=\; \prod_{i=1}^{n} \mathrm{E}[\mathrm{e}^{-h \cdot X_i}] = \textstyle\prod_{i=1}^n \left[\sum_{j \geq 1} (1-p_i)^{j-1} p_i \mathrm{e}^{-hj}\right]$$

$$\begin{aligned} &= \prod_{i=1}^{n} \left[\frac{p_i}{\mathrm{e}^h} \sum_{j \geq 0} \left(\frac{1-p_i}{\mathrm{e}^h} \right)^j \right] = \prod_{i=1}^{n} \frac{p_i}{\mathrm{e}^h} \cdot \frac{1}{1-(1-p_i)/\mathrm{e}^h} \\ &= \prod_{i=1}^{n} \frac{1}{\mathrm{e}^h/p_i + 1 - 1/p_i} = \prod_{i=1}^{n} \frac{1}{1-\mu_i+\mu_i \mathrm{e}^h} \ . \end{aligned} \tag{2}$$

for all $h > 0$. Plugging this into inequality (1) yields

$$\Pr[X \leq (1-\delta)\mu] \leq \prod_{i=1}^{n} \left(\frac{\mathrm{e}^{h(1-\delta)\mu_i}}{1-\mu_i+\mu_i\mathrm{e}^h} \right) \tag{3}$$

The fraction on the right-hand side of (3) is minimal for $h = h_1$, where

$$h_1 = \ln\left(1 + \frac{\delta}{\mu_i(1-\delta)}\right) \ .$$

Thus we set $h = \ln(1+\delta p/(1-\delta))$. For this we have

$$\Pr[X \leq (1-\delta)\mu] \leq \prod_{i=1}^{n} \left(\frac{(1+\delta p/(1-\delta))^{(1-\delta)\mu_i}}{1+\delta p \mu_i/(1-\delta)} \right) \ .$$

Since $p\mu_i \leq 1$ it holds that

$$(1+\delta p/(1-\delta))^{(1-\delta)\mu_i} \leq 1 + \delta p \mu_i \sum_{i \geq 0} \tfrac{\delta^i}{(i+1)!} \ .$$

Thus,

$$\begin{aligned} \frac{(1+\delta p/(1-\delta))^{(1-\delta)\mu_i}}{1+\delta p\mu_i/(1-\delta)} &\leq \frac{1+\delta p \mu_i \sum_{i\geq 0} \frac{\delta^i}{(i+1)!}}{1+\delta p \mu_i \sum_{i \geq 0} \delta^i} \\ &\leq 1-(1-\delta)\delta p \mu_i \left[\left(\textstyle\sum_{i \geq 0} \delta^i \right) - \left(\textstyle\sum_{i\geq 0} \frac{\delta^i}{(i+1)!} \right) \right] \\ &\leq \mathrm{e}^{-(1-\delta)\delta p \mu_i \left[\left(\sum_{i \geq 0} \delta^i \right) - \left(\sum_{i\geq 0} \frac{\delta^i}{(i+1)!} \right) \right]} \\ &\leq \mathrm{e}^{-(1-\delta)\delta p \mu_i \cdot \frac{1}{2} \sum_{i \geq 1} \delta^i} = \mathrm{e}^{-\delta^2 p \cdot \mu_i/2} \ . \end{aligned}$$

Plugging this into inequality (3) yields

$$\Pr[X \leq (1-\delta)\mu] \leq \mathrm{e}^{-\delta^2 p \mu/2} \ .$$

■

In order to bound the probability for a large deviation of B from T, we show the following lemma.

Lemma 2.9 *For any $0 \leq \epsilon \leq 1$ it holds*

$$\Pr[T \leq (1-\epsilon)B] \leq \mathrm{e}^{-\epsilon^2 B/2L} \ .$$

Proof. We consider the following two cases.

- $B = C$: Then there exists an edge e that has congestion B. Let k be the number of packets traversing this edge, *i. e.*, the latency of e is $\ell = B/k$. For all $i \in \{1, \ldots, k\}$, let the random variable X_i denote the number of time steps it takes for the ith packet to cross e. Then $X = \sum_i X_i$ denotes the time all packets need to traverse e. Since the X_i are independent and geometrically distributed with parameters $p_i = 1/\ell$, Lemma 2.8 yields that, for all $0 \leq \epsilon \leq 1$,

  $$\Pr[X \leq (1-\epsilon)B] \leq \mathrm{e}^{-\epsilon^2 B/2\ell} .$$

  Since $X \leq T$ and $\ell \leq L$, it therefore also holds that

  $$\Pr[T \leq (1-\epsilon)B] \leq \mathrm{e}^{-\epsilon^2 B/2L} .$$

- $B = D$: Then there exists a packet p that traverses a path with dilation B. The runtime for this can be bounded as above by viewing each edge as an independent geometrically distributed random variable. This implies that also in this case

  $$\Pr[T \leq (1-\epsilon)B] \leq \mathrm{e}^{-\epsilon^2 B/2L} .$$

Combining the two cases yields the lemma. ■

Now we can start to bound $\mathrm{E}[B]$. It holds

$$\mathrm{E}[B] \;=\; \sum_{b \geq 1} b \Pr[B = b] \;=\; \sum_{t \geq 1} \Pr[T = t] \sum_{b \geq 1} b \Pr[B = b \mid T = t] .$$

Let the function $\epsilon : \mathrm{I\!N} \to \mathrm{I\!R}^+$ be defined as

$$\epsilon(i) = \max\left[\sqrt{\frac{4L(\ln(\mathrm{E}[T]/L) + i)}{t}}, \frac{4L(\ln(\mathrm{E}[T]/L) + i)}{t}\right] .$$

It holds that

$$\Pr[b \geq (1+\epsilon(i))t] \quad \Leftrightarrow \quad \Pr\left[t \leq \left(1 - \tfrac{\epsilon(i)}{1+\epsilon(i)}\right) b\right]$$

and, according to Lemma 2.9,

$$\Pr\left[t \leq \left(1 - \tfrac{\epsilon(i)}{1+\epsilon(i)}\right) b\right] \leq \mathrm{e}^{-\left(\frac{\epsilon(i)}{1+\epsilon(i)}\right)^2 b/2L} \leq \mathrm{e}^{-\frac{\epsilon(i)^2}{1+\epsilon(i)} \cdot t/2L} .$$

If $\epsilon(i) \leq 1$, we have

$$\mathrm{e}^{-\frac{\epsilon(i)^2}{1+\epsilon(i)} t/2L} \leq \mathrm{e}^{-\frac{4L(\ln(\mathrm{E}[T]/L)+i)}{t} \cdot t/4L} = \frac{L}{\mathrm{E}[T]} \cdot \mathrm{e}^{-i} ,$$

and if $\epsilon(i) > 1$, we have $\epsilon(i)^2/(1+\epsilon(i)) \geq \epsilon(i)/2$ and therefore

$$\mathrm{e}^{-\frac{\epsilon(i)^2}{1+\epsilon(i)} t/2L} \leq \mathrm{e}^{-\frac{4L(\ln(\mathrm{E}[T]/L)+i)}{t} \cdot t/4L} = \frac{L}{\mathrm{E}[T]} \cdot \mathrm{e}^{-i} .$$

So in any case,

$$\Pr[b \geq (1+\epsilon(i))t] \leq \frac{L}{\mathrm{E}[T]} \cdot \mathrm{e}^{-i} .$$

Thus, it follows that

$$\begin{aligned}
\mathrm{E}[B] &= \sum_{t\geq 1}\Pr[T=t]\sum_{b\geq 1} b\Pr[B=b \mid T=t] \\
&= \sum_{t\geq 1}\Pr[T=t]\left[\sum_{b=1}^{(1+\epsilon(1))t} b\Pr[B=b \mid T=t] + \sum_{i\geq 1}(1+\epsilon(i+1))t\cdot\Pr[b\geq(1+\epsilon(i))t]\right] \\
&\leq \sum_{t\geq 1}\Pr[T=t]\left[(1+\epsilon(1))t + \frac{L}{\mathrm{E}[T]}\sum_{i\geq 1}(1+\epsilon(i+1))t\cdot \mathrm{e}^{-i}\right] \\
&= \sum_{t\geq 1}\Pr[T=t]\left[(1+\epsilon(1))t + \frac{L}{\mathrm{E}[T]}\cdot O((1+\epsilon(1))t)\right] \\
&= \sum_{t\geq 1}\Pr[T=t]\left(t+O\left(\sqrt{tL\ln(\mathrm{E}[T]/L)}\right)\right)\left(1+\tfrac{L}{\mathrm{E}[T]}\right) \\
&= \left(1+\tfrac{L}{\mathrm{E}[T]}\right)\left(\mathrm{E}[T]+O\left(\sqrt{L\ln(\mathrm{E}[T]/L)}\right)\mathrm{E}\left[\sqrt{T}\right]\right)\ .
\end{aligned}$$

Let X be an arbitrary random variable. Since $\mathrm{E}[(X-\mathrm{E}[X])^2]=\mathrm{E}[X^2]-\mathrm{E}[X]^2\geq 0$, it holds $\mathrm{E}[X]^2\leq \mathrm{E}[X^2]$. Substituting X by $\sqrt{T}$ and taking the root on both sides yields $\mathrm{E}[\sqrt{T}]\leq\sqrt{\mathrm{E}[T]}$. Hence, since $L\leq 2\mathrm{E}[T]$,

$$\begin{aligned}
\mathrm{E}[B] &\leq \left(1+\tfrac{L}{\mathrm{E}[T]}\right)\left(\mathrm{E}[T]+O\left(\sqrt{\mathrm{E}[T]L\ln(\mathrm{E}[T]/L)}\right)\right) \\
&= \mathrm{E}[T]+O\left(\sqrt{\mathrm{E}[T]L\ln(\mathrm{E}[T]/L)}\right)\ .
\end{aligned}$$

If we assume that $\mathcal{S}$ is a best possible strategy for routing π, then $\mathrm{E}[T]=R$. (Or $\mathrm{E}[T]\leq 2R$ in case that Lemma 2.7 had to be used. But then $L=\Theta(R)$, which means that $\sqrt{RL\ln(R/L)}=\Theta(R)$.) In this case, $\mathrm{E}[B]=R+O(\sqrt{RL\ln(R/L)})$, which implies that there exists a path collection for routing π with congestion and dilation at most $R+O(\sqrt{RL\ln(R/L)})$. This proves Theorem 2.6. ∎

Once we have this theorem, the question is whether it is possible to find such a path collection in an efficient way for every permutation routing problem. An option we want to consider in this paper is to construct a *path system*, from which suitable paths for routing the packets can be easily chosen. A path system is defined as a set of paths in G that contains exactly one path for every pair of nodes in G. The dilation of a path system is defined as for a path collection. The following theorem shows that we can construct efficient path systems ("w.h.p." means with probability at least $1-N^{-c}$ for any constant c).

Theorem 2.10 *For any PCG G of size N in which every permutation can be routed along a path collection with congestion and dilation at most B and edge latency L, a simple path system $\mathcal{P}$ with dilation at most B can be constructed in polynomial time such that the congestion of routing one packet from every node v to a node chosen uniformly at random by v is bounded by $B+O(\sqrt{(B+L\log N)L\log N})$, w.h.p.*

Proof. First we show the existence of an efficient path system. According to the definition of B, for any permutation $\pi_i:[N]\to[N]$ with $\pi_i(x)=(x+i) \bmod N$ for all $i,x\in[N]$ there is a path collection $\mathcal{P}_i$ with congestion and dilation at most B. We then choose $\mathcal{P}=\bigcup_{i=0}^{N-1}\mathcal{P}_i$ to be the path system for G. Clearly, this path system has congestion at most $N\cdot B$ and dilation at most B.

Next we bound the congestion that holds w.h.p. for routing a random function in G using paths in $\mathcal{P}$. Consider some fixed edge e in G, and let ℓ be its latency. Let the random variable C_e be defined as the congestion at e. For any node v in G, let the binary random variable X_v be one if and only if the packet with source v contains e in its routing path. Then $C_e = \ell \sum_{v \in V} X_v$. Since each node sends out a packet to a destination chosen uniformly at random, we have $\mathrm{E}[C_e] \leq B$. As the X_v are independent, the Chernoff-Hoeffding bounds yield that

$$\Pr[C_e \geq (1+\epsilon)B] \leq \begin{cases} e^{-\epsilon^2 B/3\ell} & : \text{ if } 0 \leq \epsilon < 1 \\ e^{-\epsilon \cdot B/3\ell} & : \text{ if } \epsilon \geq 1 \end{cases}$$

This probability is polynomially small in N for $\epsilon = O(\max[\sqrt{\frac{\ell \log N}{B}}, \frac{\ell \log N}{B}])$ sufficiently large. Hence, w.h.p. all edges have a congestion of at most $B + O(\sqrt{(B + L \log N) L \log N})$. It remains to be shown how to construct an efficient path system.

Constructing an efficient path system

First, let us assume that all probabilities attached to the edges in G are of the form $1/\ell$, where $\ell \in \mathbb{N}$. (If this is not the case then divide all probabilities by N^α, where α is a constant chosen for a sufficiently good precision. Then round all probabilities to the nearest $1/\ell$, where $\ell \in \mathbb{N}$ and continue with the construction as described below. It is easy to see that in this case the relative rounding error per edge is below $1 + N^{-\alpha}$. This ensures that the amount by which the bounds on the congestion and dilation below have to be increased is at most $(1 + 1/N^\alpha)$, which is insignificantly small.) The following algorithm will serve as a basic building block for our algorithm to construct an efficient path system in G.

For any $d \in \mathbb{N}$, let G_d denote a leveled network of depth d, *i. e.*, a network consisting of $d+1$ sets of nodes called *levels*. Let its levels be numbered from 0 to d. For each level, let its node set represent the set of all nodes in G. Two nodes v and w are connected if

- v and w are in consecutive levels and represent the same node in G, or
- v and w are in levels i and $i+\ell$ for some $i \geq 0$ and the nodes represented by v and w are connected via an edge with latency ℓ in G.

Let all edges be directed from the lower to the higher level. We consider the problem of sending one unit of flow from every node at level 0 to every node at level d. (That is, we have a so-called multicommodity flow problem.) There are certain capacity limits that have to be kept in order for a solution to our problem to be valid: We demand that for every edge e with latency ℓ in G, all edges in G_d representing copies of e together are allowed to forward a flow of at most $N \cdot d/\ell$. There are no restrictions on how this flow is distributed among these edges. All edges in G_d that connect copies of the same node in G are assumed to have infinite capacity.

If we allow fractional flows then linear programming can be used to find a solution (or stop with the answer that no solution exists) in polynomial time.

Now we are ready to formulate our algorithm for finding an efficient path system.

(1) Find the minimum d for which there is a solution to the multicommodity flow problem for G_d as stated above.

(2) Transform the multicommodity flow for G_d into a multicommodity flow for G by identifying nodes in G_d representing the same node in G. Since all copies of an edge e in G with latency ℓ together have a capacity of $d \cdot N/\ell$, the fractional congestion of e is at most $d \cdot N$.

(3) Use Raghavan's method [34] for converting fractional flows into integral flows. This results in a path system for G with dilation at most d and congestion at most $d \cdot N + O(\sqrt{(d \cdot N + L \log N) L \log N})$.

Clearly, the algorithm runs in polynomial time. It remains to bound d. According to the existence proof, there is a path system $\mathcal{P}$ with dilation at most B and congestion at most $C = B \cdot N$. This path system can be directly transformed into an integral solution to our multicommodity problem for G_B (the paths of $\mathcal{P}$ represent the paths of the units of flow). Hence, there exists a fractional solution to our multicommodity flow problem with $d \leq B$. Raghavan's rounding method therefore produces a path system with congestion at most $B \cdot N + O(\sqrt{(B \cdot N + L \log N) L \log N})$.

Using the same analysis for bounding the congestion of routing random functions as in the existence proof yields Theorem 2.10. ■

Hence, for $L = O(\frac{R}{\log N})$, the congestion and dilation of routing a randomly chosen function using paths in $\mathcal{P}$ is bounded by $O(R)$, w.h.p. Using Valiant's trick [40] of routing packets first to randomly chosen intermediate destinations before they are routed to their original destinations, we can get this congestion bound for arbitrary permutations, w.h.p. So if there were an offline protocol that can route packets along an arbitrary simple path collection with congestion C and dilation D in expected time $O(C + D)$, then the *optimal* worst case (and average case, see Theorem 2.5) expected time to route packets according to an arbitrary permutation can be reached by using a fixed path system. We will address this question in the next section.

2.3 Scheduling strategies for PCGs

In this section, we prove upper bounds for offline and online scheduling of packets along a fixed collection of paths in a PCG. Recall that in offline scheduling it is assumed that all nodes know the entire scheduling problem from the beginning (and are therefore able to compute an optimal schedule since internal computations are usually not counted for the time complexity of scheduling algorithms), whereas in online scheduling every node initially only knows the packets it stores.

2.3.1 Offline scheduling

In order to simplify the development of efficient offline schedules for PCGs, we compare these communication graphs with their deterministic equivalents. For this, we introduce the following type of communication graph.

Definition 2.11 *Let the* deterministic communication graph *(DCG) $G = (V, \gamma)$ be defined as a complete directed graph with node set V and a function $\gamma : V \times V \to \mathrm{IR}^+ \cup \{\infty\}$. For any edge (u, v), $\gamma(u, v)$ represents the time it takes to send a packet from u to v.*

Consider any PCG $G = (V, \varphi)$. Let the DCG of G be defined as $\bar{G} = (V, \gamma)$ with $\gamma(e) = 1/\varphi(e)$ for all $e \in V \times V$. Consider the problem of simulating an offline protocol for $\bar{G}$ by G. The next theorem shows that this can be done efficiently. The proof can be found in the appendix.

Theorem 2.12 *Consider any offline protocol $\mathcal{S}$ that requires T time steps to send all packets along a given path collection of size n in $\bar{G}$. Let L denote the maximum edge latency and ℓ denote the minimum edge latency of this path collection. Then $\mathcal{S}$ can be taken to construct a protocol with runtime $O(T \log(L/\ell) + L \log n)$, w.h.p., for sending the packets along the same path collection in G.*

Note that Theorem 2.12 is true for *any* (even non-simple) path collection. This may allow to use it for a wider range of problems than just communication problems.

2.3.2 Online scheduling

With the help of the idea behind the online protocol in [27], we will show the following result.

Theorem 2.13 *There is an online protocol for sending packets along an arbitrary path collection of size n with dilation D, congestion C, maximum edge latency L and minimum edge latency ℓ in time $O(C + D\log(n \cdot L/\ell))$, w.h.p.*

Proof. We assume in the following that the minimum edge latency, ℓ, is equal to 1. It is easy to modify the proof such that it also holds for any minimum edge latency.

Let us first present a protocol for a DCG, called *random delay protocol*, before we show how to convert it to a protocol for a PCG. The protocol assumes that all links have bandwidth B (fixed later), that is, up to B packets can traverse a link at one time step. Let us call an interval of t consecutive time steps a *t-interval*. The following algorithm is used as a basic building block for the random delay protocol.

> **Algorithm Route(s):**
> Each packet is assigned an initial delay, chosen uniformly and independently at random from the range $[C/\log(Ln)]$. A packet that is assigned a delay of δ waits in its initial buffer for δ steps and then moves on without waiting again until it reaches its destination or traversed a path with dilation at least s. If a packet arrives at an edge that is currently traversed by B other packets, then it stops and stays at the link for the rest of Route(s).

The random delay protocol works as follows.

```
repeat
    execute Route(min[$D, L \cdot n$])
until all packets reached their destinations
```

Lemma 2.14 *Suppose we are given an arbitrary simple path collection $\mathcal{P}$ of size n with congestion C, dilation D and edge latency L in some DCG G. Then the random delay protocol needs at most $O(C + D\log(Ln))$ time steps to finish routing in $\mathcal{P}$, w.h.p.*

Proof. Let us consider some fixed edge e and time step t during the execution of Route(s). Let ℓ be the latency of e. Since at most C/ℓ packets want to traverse e and each of these packets chooses an initial delay independently at random from a range of size $C/\log(Ln)$, the probability that at least $B = \max[\alpha+3, 2\mathrm{e}]\log(Ln) + 3$ packets arrive at e within some ℓ-interval is at most

$$\begin{aligned} &(\min[D, L \cdot n] + C/\log(Ln)) \binom{C/\ell}{B} \left(\frac{\ell}{C/\log(Ln)}\right)^B \\ \le\ & 2Ln \left(\frac{\mathrm{e}\log(Ln)}{B}\right)^B \le 2Ln \left(\frac{1}{2}\right)^{(\alpha+3)\log(Ln)+3} = \frac{1}{4(Ln)^{\alpha+2}} \ . \end{aligned}$$

We say that a packet P *fails* at edge e if at that time it reaches it, e is already used by B other packets. Clearly, if P fails then at least B arrivals of other packets fall in an ℓ-interval ending with P's arrival time. Hence, the probability that P fails at least $k = \lceil D/(Ln) \rceil$ times during the execution of the random delay protocol is bounded by

$$\binom{D+k}{k} \left(\frac{1}{4(Ln)^{\alpha+2}}\right)^k \le \left(\frac{4D}{k}\right)^k \left(\frac{1}{4(Ln)^{\alpha+2}}\right)^k \le \left(\frac{1}{(Ln)^{\alpha+1}}\right)^k \le \frac{1}{(Ln)^{\alpha+1}} \ .$$

Since there are n packets to consider, the probability that there exists a packet with at least k failures is at most $n \cdot (Ln)^{-\alpha-1} \le n^{-\alpha}$. Hence, w.h.p. the random delay protocol successfully routes all packets along the given path collection in time

$$\begin{aligned} & B\cdot(\min[D,L\cdot n]+C/\log(Ln))\cdot(D/(\min[D,L\cdot n])+k) \\ = \quad & O((C+\min[D,L\cdot n]\cdot\log(Ln))\cdot(D/(\min[D,L\cdot n])+\lceil D/(Ln)\rceil)) \\ \overset{C\le Ln}{=} \quad & O(C+D\log(Ln)) \;. \end{aligned}$$

This completes the proof of Lemma 2.14. ∎

It remains to be shown how to convert this protocol into a protocol for a PCG. For each time step of the protocol for a DCG with bandwidth B, let us use $4B$ time steps for the PCG.

Consider the simulation of some fixed call of Route(s). Let each packet have a rank denoting, for each edge it currently waits at, the time it arrives there in a DCG with bandwidth B. A packet currently staying at an edge with latency ℓ is declared *active* if the current time step t is within the time interval $[4\ell B\cdot r, 4\ell B(r+1))$ of its rank r. Only active packets are allowed to try to cross a link. If several packets are active at the same time at some edge then the packet with lowest rank is chosen for attempts to cross the edge. (In case that several packets have the same rank, an arbitrary packet may be chosen.) If a packet is not able to traverse an edge within its active period, it stops and stays at that edge for the rest of Route(s). Also if, for some packet P and edge e with latency ℓ along P's path, more than B packets with lower ranks than P are active at e when P arrives there, P stops and stays at that edge for the rest of Route(s). If a packet is involved in one of these two cases, we say that it *fails* Route(s).

The rules above imply that every active packet is competing with at most $B-1$ other active packets at any time step during the simulation. Hence, the probability that a packet with rank r fails to traverse its current link e within the time interval $[4\ell B\cdot r, 4\ell B(r+1))$ is at most the probability that e has less than B successes within $4\ell B$ time steps. Let the random variable X denote the number of successes e has within $4\ell B$ time steps. Clearly, $\mathrm{E}[X] = 4B$. Applying the Chernoff bounds with $\epsilon = 1 - \frac{1}{4}$, we obtain

$$\Pr[X \le B] = \Pr[X \le (1-\epsilon)\mathrm{E}[X]] \le \mathrm{e}^{-\epsilon^2 E[X]/2} = \mathrm{e}^{-(1-1/4)^2 4B/2} \le \mathrm{e}^{-B} \;.$$

Using this together with the proof of Lemma 2.14 yields Theorem 2.13. ∎

With a more involved proof we obtain the following result.

Theorem 2.15 *There is an online protocol for sending packets along an arbitrary simple path collection of size n with dilation D, congestion C, maximum edge latency L and minimum edge latency ℓ in time*

$$O\left(\log(L/\ell)\left(C + D\log\left(L/\ell\cdot\log n\right) + L\cdot\left(\frac{\log(n\cdot L/\ell)}{\log\left(L/\ell\cdot\log n\right)}\right)^2\right)\right) \;,$$

w.h.p.

Proof. The proof basically combines the ideas in [14] and [33] with Theorem 2.12.

Let us assume in the following that $D \le L\cdot n$. The case $D > L\cdot n$ can be dealt with similar to the proof of Theorem 2.13 (see the way the random delay protocol calls Route(s)). Furthermore, let us assume for the moment that the minimum edge latency, ℓ, is 1. We again first present a protocol for a DCG, called *advanced random delay protocol* (or advanced RDP), before we show how to convert it to a protocol for a PCG. The protocol assumes that all links have bandwidth B (fixed later), that is, up to B packets can traverse a link at one time step. In the following, let

- $p = \frac{\log(Ln)}{\log(L\log n)}$,
- $C_b = b \cdot L\log(Ln) \cdot p$ for some constant $b > 0$ to be set later and
- $D_b = \frac{C_b}{\log(L\log n)}$.

The following algorithm is used as a basic building block for our protocol.

> **Route-Block:**
> In a first pass, each participating packet chooses an initial delay uniformly and independently at random from the range $[D_b]$. A packet that is assigned a delay of δ waits in its initial buffer for δ steps. Afterwards it moves on in the following way until it reaches its destination or traversed a path with dilation at least D_b: For each edge e with latency ℓ at which it arrives, it waits for ℓ many steps before traversing e. If, for some edge e with latency ℓ, more than B packets arrive at e during an ℓ-interval then all of them stop. Packets to which this happens are declared *unsuccessful.*
> After the first pass (which takes at most $3D_b$ time steps) $2p$ further passes are performed for the unsuccessful packets. For each pass, every unsuccessful packet chooses an initial delay uniformly and independently at random from the range $[D_b/p]$. The forwarding and stopping of packets in case of a too high contention is done as above. Packets that traversed a path with dilation at least D_b/p stop for this pass. Packets that reach their destination or altogether traversed a path with dilation at least D_b stop for the rest of Route-Block.

The advanced RDP then works as follows.

> Execute Route-Block $2C/C_b + D/D_b$ times. Each packet is assigned an integer, chosen uniformly and independently at random from the range $[2C/C_b]$. A packet that is assigned a value of d awaits d executions of Route-Block before participating in (at most) D/D_b consecutive calls of Route-Block.

First, let us bound the runtime of the protocol. If all edges have a bandwidth of B and the minimum latency is 1, this is at most

$$\left(\frac{2C}{C_b} + \frac{D}{D_b} + 1\right)(3D_b + 2p \cdot (3D_b/p)) = \left(\frac{2C}{C_b} + \frac{D}{D_b} + 1\right) 9D_b = \frac{18C}{\log(L\log n)} + 9D + 9D_b \ .$$

Hence, for $B = O(\log(L\log n))$ this would result in a runtime of

$$O\left(C + D\log(L\log n) + L \cdot \left(\frac{\log(Ln)}{\log(L\log n)}\right)^2\right) \ .$$

In case that the minimum latency, ℓ, is not 1, we scale the parameters C, D, and L to $C' = C/\ell$, $D' = D/\ell$, and $L' = L/\ell$, respectively. In this case we arrive at a time bound of

$$O\left(C' + D' \cdot \log(L'\log n) + L' \cdot \left(\frac{\log(L'n)}{\log(L'\log n)}\right)^2\right) \ .$$

Multiplying this bound with ℓ to obtain the original runtime, we obtain the bound stated in Theorem 2.15. We next show that the probability that a packet fails to reach its destination is very low.

Lemma 2.16 *For any constant $c > 0$ there is a constant $b > 0$ such that, for $B = b\log(L\log n)$ and C_b and D_b chosen as above, the probability that the advanced RDP fails to send a packet to its destination is at most n^{-c}.*

Proof. First we bound the probability that the congestion within some call of Route-Block exceeds C_b (Claim 2.17). Then we show that under the assumption that the congestion in Route-Block is at most C_b, the probability is very small that, for any edge e along the path of a participating packet, there are more than C_b/p packets left with paths containing e that were unsuccessful in the first pass (Claim 2.18). Finally, we show that if the congestion at any edge e caused by unsuccessful packets is at most C_b/p after the first pass, then the probability is very small that one of the unsuccessful packets fails in so many of the $2p$ other passes that it is not able to traverse a path of dilation D_b (Claim 2.19). Combining the three claims yields the lemma.

Claim 2.17 *For any $c > 0$ there is a $b > 0$ such that the probability that there is a call of Route-Block with a congestion of more than C_b is at most n^{-c}.*

Proof. Consider some fixed execution of Route-Block and some fixed edge e. Let the random variable X denote congestion caused by the packets that intend to cross e in this call. That is, if e has a latency of ℓ then each packet that intends to cross e in this call contributes ℓ to X. Since the total congestion at e is at most C and the packets choose at random one of $2C/C_b$ possible starting blocks, $\mathrm{E}[X] \leq C \cdot C_b/(2C) = C_b/2$. Because the packets choose their starting blocks independently at random, we can use Chernoff bounds to obtain, for $\epsilon = 1$,

$$\Pr[X \geq C_b] = \Pr\left[X \geq (1+\epsilon)\tfrac{C_b}{2}\right] \leq \mathrm{e}^{-\epsilon\frac{C_b/2}{3\ell}} \leq \mathrm{e}^{-\frac{b\log^2(Ln)}{6\log(L\log n)}} \ .$$

Clearly, there are at most $D \cdot n$ different edges and at most $2C/C_b + D/D_b \leq n$ calls of Route-Block to consider per edge. Since we assume that $D \leq L \cdot n$, the probability that there is an edge with congestion more than C_b is at most

$$D \cdot n^2 \mathrm{e}^{-\frac{b\log^2(Ln)}{6\log(L\log n)}} \leq (L \cdot n)^3 \cdot \mathrm{e}^{-b\log(Ln)/6} \leq n^{-c}$$

for $b \geq 5(c+3)$. ∎

Given some fixed call of Route-Block and a packet P, a subpath q of P's path is called its *active path segment* if q represents the path P tries to traverse during this call. Then we can show:

Claim 2.18 *Assume that the congestion in any call of Route-Block is at most C_b. Then, for any $c > 0$ there is a $b > 0$ such that the probability is at most n^{-c} that, for any edge e along an active path segment, the congestion at e caused by packets that were unsuccessful in the first pass is more than C_b/p.*

Proof. Consider some fixed call of Route-Block. A *site* is defined as an ordered pair (e, I), where e is an edge and I is a time interval within the first pass of Route-Block. A packet *aims for* a site (e, I) if it selects a random delay such that it intends to arrive at e within I. For any edge e, let I_e denote an interval of length the latency of e. Given a link bandwidth of B and an edge e, a site (e, I_e) is declared *bad* if more than B packets arrive at e within I_e.

Consider some fixed edge e, and let P_e be defined as the set of all edges contained in the active path segments that cross e. Consider marking any m sites along the edges of P_e. According to our assumptions, e is crossed by at most C_b active path segments and each of these segments contains at most D_b edges. Furthermore, at most $3D_b$ sites have to be considered per edge. Hence, there are at most $C_b D_b \cdot 3D_b$ possibilities of marking a site. Let the random variable X denote the total number of packets that aim for any of the m marked sites. Since the number of active segments containing an edge e' with latency ℓ in P_e is at most C_b/ℓ and each of the corresponding packets chooses a random delay from a range of size D_b, the expected number of packets in any one site of e' is at most $\frac{C_b}{\ell} \cdot \frac{\ell}{D_b} = \log(L \log n)$. Therefore, $\mathrm{E}[X] \leq m \log(L \log n)$. Consider the marked sites to consist of all bad sites. Since a packet can only participate in one bad site, X can be regarded as the sum of independent binary random variables. Hence, we can use the Chernoff bounds to bound the probability that X is at least $B \cdot m$ with $B = b \log(L \log n)$:

$$\Pr[X \geq B \cdot m] \leq \left(\frac{\mathrm{e}^{b-1}}{b^b}\right)^{m \log(L \log n)} .$$

Let $m = \max[\frac{\log(Ln)}{\log(L \log n)}, 3\mathrm{e}]$. Then we get that, for $b = \max[c + 3 \log b + 3, \mathrm{e}^2]$, the probability that there exist m bad sites along the active segments containing e is at most

$$\begin{aligned}\binom{3C_b D_b^2}{m} \left(\frac{\mathrm{e}^{b-1}}{b^b}\right)^{m \log(L \log n)} &\leq \left(\frac{3\mathrm{e}(bL \log^2(Ln))^3}{m}\right)^m \cdot \mathrm{e}^{-bm \log(L \log n)} \\ &\leq \mathrm{e}^{m \cdot 6 \log(\sqrt{bL} \log(Ln)) - bm \log(L \log n)} \\ &\leq \mathrm{e}^{(6 \log \sqrt{b} - b) m \log(L \log n)} = \mathrm{e}^{-(c+3) \log(Ln)} \leq (Ln)^{-(c+3)} .\end{aligned}$$

This bounds the number of bad sites in P_e and therefore the number of packets that fail and whose active path segments contain e to most $b \log(Ln)$ with probability at least $1 - (Ln)^{-(c+3)}$.

Since there are at most $D_b \cdot n$ edges to consider in a call and there are at most $2C/C_b + D/D_b \leq L \cdot n$ calls, the probability that some edge has more than $b \log(Ln)$ packets that fail the first pass is at most

$$D_b \cdot L \cdot n^2 (Ln)^{-(c+3)} \leq n^{-c} .$$

Hence, the maximum congestion at any edge after the first pass is at most $Lb \log(Ln) = C_b/p$ with probability at least $1 - n^{-c}$. ■

Claim 2.19 *Assume that the congestion at any edge e caused by unsuccessful packets is at most C_b/p after the first pass of any call. Then for any $c > 0$ there is a $b > 0$ such that the probability is at most n^{-c} that one of the unsuccessful packets fails in so many of the $2p$ other passes that it is not able to traverse a path of dilation D_b in a call of Route-Block.*

Proof. Consider some fixed execution of Route-Block. Within this execution, consider some fixed packet P and edge e along P's active path segment. Let ℓ be the latency of e. The probability that P runs at e into a bad site is at most

$$\binom{C_b/(\ell \cdot p)}{B} \left(\frac{\ell}{D_b/b}\right)^B \leq \left(\frac{\mathrm{e}}{b}\right)^B .$$

Let $B = b \log(L \log n)$ and $b \geq 4\mathrm{e}$. Then the probability that P fails a pass is at most

$$\frac{D_b}{p} \cdot \left(\frac{\mathrm{e}}{b}\right)^B \leq bL \log^2(Ln) \cdot \left(\frac{1}{2}\right)^{2b \log(L \log n)} \leq (L \log n)^{-(b+1)} .$$

P fails the call of Route-Block if it fails in more than p passes. The probability for this is at most

$$\binom{2p}{p} \left(\frac{1}{L \log n}\right)^{(b+1)p} \leq \left(\frac{1}{L \log n}\right)^{bp} \leq n^{-b} .$$

Counting over all possible packets and calls of Route-Block yields the claim for $c \leq b - 2$. ■

The claims complete the proof of Lemma 2.16. ∎

It remains to be shown how to convert this protocol into a protocol for a PCG. For this, let us assume that an offline protocol $\mathcal{S}$ is given for a DCG $\bar{G}$ with link bandwidth B, that is, an edge can forward up to B packets simultaneously in each direction. This offline protocol has to be simulated on a PCG G with link bandwidth 1.

An easy modification of the proof of Theorem 2.12 shows that this can be done in $O(B \cdot T \log(L/\ell) + L \log n)$ time steps. To ensure that this time bound also holds for our online protocol, we have to modify our protocol in the following way:

Consider some pass of a fixed call of Route-Block. Each packet is given a rank denoting the time step at which it is supposed to cross its actual link in $\bar{G}$ during that pass. All packets again wait for the latency of their current edge many time steps before trials are started to cross an edge. If, at some time step during the simulation, more than B packets with the same rank wait at an edge, all of them are deleted.

To be able to use the analysis above for our online protocol, the definition of bad sites has to be slightly changed. Now, a site is called bad if more than B packets not only intend to use it, but wait for transmission at the same time at the corresponding edge. This definition ensures that every packet can only be used once as a witness of a bad site, since it is deleted at the corresponding edge by the rule above. The independence assumptions in Claim 2.18 are therefore still correct. This allows us to use the proof of Lemma 2.16 to show Theorem 2.15. ∎

2.4 Putting everything together

Theorem 2.4 together with Theorem 2.5, Theorem 2.10 and Theorem 2.13 show that for *any* transmission graph using a scheme out of our class of local probabilistic control schemes there is an online protocol that can reach up to a factor of $O(\log N)$ for arbitrary permutations the optimal average case permutation routing time. Theorem 2.15 further shows that this can be reduced to a factor of $O((\log\log N)^2)$ if $L = \log^{O(1)} N$ and $L \leq \frac{R}{\log^2 N}$. It would be interesting to see, whether further improvements are possible.

3 Wireless Communication in a Euclidean space

We next turn to the case where the nodes that wish to communicate with each other are points in $\mathbb{R}^d$, and the weight of an edge that connects any two points p_1 and p_2 is the Euclidean distance between the two points in $\mathbb{R}^d$: this models the scenario where the required signal strength to send a message between a pair of nodes is only a function of distance. The cases where $d = 2$ and $d = 3$ are the most relevant from a practical standpoint; we here focus on the case where $d = 2$. Recall that we consider the case where the nodes are distributed independently and uniformly at random in a fixed region of $\mathbb{R}^2$, called the domain space.

3.1 Upper bounds

3.1.1 Square domain spaces

We start with the case where the domain space is exactly a square. We demonstrate how to take advantage of the considerable similarity between randomly distributed nodes in such a domain space and existing work on computing with faulty arrays [35, 24, 13]. For $Z = a^2$ for some integer a, let a *Z-partition* of a square domain space be the partition of the domain space into exactly Z equal sized square regions and let r_{ij} denote the region in the ith row and the jth column, counting from some fixed corner of the domain space. The following easy lemma is useful in many of our later proofs.

Lemma 3.1 *For any Z-partition and any communication pattern where the nodes in every region send at most h messages and every message sent from region r_{ij} has a destination node in region r_{ij}, or in a region that abuts region r_{ij}, there exists a schedule that sends every message in time $O(h)$.*

Proof. Recall that α, the ratio of the interference radius to the transmission radius, is a constant. Regardless of where in a region nodes are distributed, every message sent from region r_{ij} directly to its destination conflicts only with regions at most $k = \lceil 3\alpha \rceil$ away from region r_{ij}. Thus, all messages sent from any region r_{ij}, where $i = x \bmod k$ and $j = y \bmod k$ can be sent concurrently, where $0 \leq x, y < k$. Thus, all messages can be scheduled in k^2 waves, where in wave m, we schedule all messages sent from a region r_{ij}, where $i = m \bmod k$ and $j = \lfloor m/k \rfloor \bmod k$. Each wave requires time $O(h)$. ∎

This implies for example that if we have an n-partition with exactly one communication node located somewhere in each region, then in a constant number of steps, we can route any communication pattern performed in a single step on a (non-faulty) $\sqrt{n} \times \sqrt{n}$ array. Here, the node in region r_{ij} performs the communication performed by processor p_{ij} of the array.

When each node is distributed uniformly and independently at random, it is very unlikely that there will be exactly one node in each region of the n-partition. Let $S_{n,m}$ be a placement of n nodes into an m-partition. Given any $S_{n,m}$, there is a $\sqrt{m} \times \sqrt{m}$ faulty array $F(S_{n,m})$ such that processor p_{ij} of $F(S_{mn})$ is fault-free if $S_{n,m}$ has at least one node in region r_{ij} of the m-partition, and is faulty otherwise. A faulty processor cannot send or receive any messages. $S_{n,m}$ can emulate $F(S_{n,m})$ with constant slowdown. Note that in most cases the number of nodes in $S_{n,m}$ is strictly larger than the number of processors in $F(S_{n,m})$, since some of the regions of the domain space contain more than one node. When region r_{ij} contains more than 1 node, one arbitrarily chosen node in the region performs the communication performed by processor p_{ij} of the array.

Existing work on routing with faulty arrays considers the scenario where each node fails independently with probability p [35, 24, 13]. Thus, in order to simulate algorithms for the faulty array, we need to deal with the fact that the event that a given square contains some node is dependent on how many other squares also contain nodes. We do this by requiring that the algorithm for the array to be simulated is (p, k, m)*-guaranteed*, a technical condition described below. This technical condition is actually fairly natural, and most existing algorithms for faulty arrays adhere to it.

In order to describe an algorithm that is (p, k, m)*-guaranteed*, we first need to define some terms. Let A_s denote the processors in s, a subset of the processors of faulty array A, and let $f(A_s)$ denote the set of fault-free processors in A_s. An *array property* specifies a subset A_s and a function $P_s(f(A_s))$ from $f(A_s)$ to $\{0,1\}$.

Definition 3.2 *A* monotonic array property *is an array property $P_s(f(A_s))$ such that for two faulty arrays A and A', where $f(A_s) \subset f(A'_s)$, if $P_s(f(A_s)) = 1$, then $P_s(f(A'_s)) = 1$.*

Most algorithms for faulty arrays work correctly provided that a subset of the processors with a specified structure is fault-free: such a requirement is a monotonic array property. An example of an algorithm that might not be so guaranteed is one where the question of whether a processor is faulty or not is used to provide random bits.

Definition 3.3 *A set of k array properties $\{P_1(f(A_1)), \ldots, P_k(f(A_k))\}$ is said to* (p, k, m)-guarantee *an algorithm for the faulty array A if $\forall j$, $1 \leq j \leq k$, it holds that $\Pr[P_j(f(A_j)) = 0] \leq p$, and the number of processors in $A_j \leq m$, and the algorithm functions correctly whenever $\forall 1 \leq j \leq k, P_j(f(A_j)) = 1$.*

In other words, an algorithm is (p, k, m)-guaranteed if we can specify a set of k (not necessarily disjoint) subsets of the processors, each of size at most m, and each with an array property that is 1 with probability at least $1 - p$, such that if all the array properties are satisfied then the algorithm functions correctly.

Lemma 3.4 *Let A be a $\sqrt{n} \times \sqrt{n}$ faulty array where every processor is faulty independently with probability q, for any constant q. Let P_j be any monotonic array property on A_j, a subset of at most $\frac{n}{2}$ of the processors of A. There exists a constant c such that for $S_{cn,n}$ the distribution of cn wireless nodes into an n-partition,*

$$\Pr[P_j(f(F(S_{cn,n})_j)) = 1] \geq \Pr[P_j(f(A_j)) = 1].$$

Proof. For some fixed ordering of processors in A_j, let $x_i = 1$, $1 \leq i \leq |A_j|$ if processor i of A_j is fault-free and 0 otherwise. Let $y_i = 1$, $1 \leq i \leq |A_j|$ if processor i of $F(S_{cn,n})_j$ is fault-free and 0 otherwise. Also, for an ordering of all processors in $F(S_{cn,n})$, let $z_i = 1$, $1 \leq i \leq n$, if processor i of $F(S_{cn,n})$ is fault-free and 0 otherwise.

Claim 3.5 *There exists a c such that for y_i defined by $S_{cn,n}$,*

$$\Pr[y_i = 1 | y_1 \ldots y_{i-1}, y_{i+1} \ldots y_{|A_j|}] \geq q\ .$$

Proof. The process of determining which $z_i = 1$ can be viewed as the two part process where first we determine $N = \sum z_i$, the number of fault-free nodes in $F(S_{cn,n})$, and then we choose which nodes are fault free by choosing uniformly a set of N variables z_i, set them to 1, and set the remainder of the z_i to 0.

For $S_{cn,n}$, the expectation of N is $\geq (1 - e^{-c})n$, and using standard Martingale techniques, when c is a constant, we can show that

$$\Pr[N < (1 - 2e^{-c})n] \leq e^{-\Omega(n)}.$$

We first condition on the value of N. If we then condition on the values of $y_1 \ldots y_{i-1}, y_{i+1} \ldots y_{|A_j|}$, the setting of these random variables that minimizes $\Pr[y_i = 1]$ is when $y_1 = y_2 = \ldots y_{i-1} = y_{i+1} = \ldots y_{|A_j|} = 1$. Thus,

$$\Pr[y_i = 1 | N, y_1 \ldots y_{i-1}, y_{i+1} \ldots y_{|A_j|}] \geq \frac{N - |A_j|}{n - |A_j| - 1} \geq \frac{2N - n}{n + 2}.$$

By Bayes rule, $\Pr[y_i = 1 | y_1 \ldots y_{i-1}, y_{i+1} \ldots y_{|A_j|}]$ is at least this same probability conditioned on the fact that $N \geq (1 - 2e^{-c})n$, multiplied by $1 - e^{-\Omega(n)}$. However, for any constant q, there is a constant c such that

$$\frac{2(1 - 2e^{-c})n - n}{n}(1 - e^{-\Omega(n)}) > q.$$

■

Given this claim, we prove the lemma as follows. Deciding which $x_i = 1$ (*i. e.*, deciding which processors of the faulty array A_j are fault-free) can be viewed as traversing a complete binary tree T of depth $|A_j|$ from the root to a leaf, where at level i, we branch right if $x_i = 1$ and left otherwise. Let $A_j(x)$ be the xth leaf of T in the left to right ordering of the leaves of T. For some of the leaves of T, $P_j(f(A_j(x))) = 1$. We label every node N of T with a probability p_N, which is the probability of reaching any leaf $A_j(x)$ such that $P_j(f(A_j(x))) = 1$, given that we have already reached node N.

We label both outgoing edges from node N with the probability of taking that edge from N. In T, all right branches have label q, and all left branches have label $1 - q$.

For $S_{cn,n}$, we define an analogous tree T', where at level i of T' we branch right if $y_i = 1$, and we branch left otherwise. We label every node N' of T' with a probability $p_{N'}$, which is the probability of reaching a leaf $F(S_{cn,n})_j(x)$ such that $P_j(f(F(S_{cn,n})_j(x))) = 1$, given that we have already reached node N'. We label both outgoing edges from node N' with the probability of taking that edge from N'. If node r is the root of T and node r' is the root of T', then to prove the lemma, we need to show that $p_r \leq p_{r'}$.

To do this, we use a series of trees $T_1 \ldots T_{|A_j|}$, where $T_1 = T$ and $T_{|A_j|} = T'$. T_j is the complete binary tree where all edge labels between nodes which both have height at most j (counting the root as height 1) are identical to the corresponding labels in the tree T', the remainder of the edge labels are identical to the corresponding labels in the tree T, and the node labels of the leaves are the same as both T and T'. We show that for every node in T_j, the node label of the corresponding node in T_{j+1} is at least as large, and the lemma follows directly by induction.

We use two facts. For any internal node N of T or T', let $l(N)$ and $r(N)$ be the left and right children of N respectively. Fact 1 is that for any node N' of T', the probability of proceeding from N' to $r(N')$ is at least q. This follows from Claim 3.5. Fact 2 is that for any node N of T, $p_{r(N)} \geq p_{l(N)}$. This follows from the fact $P_j(A_j)$ is monotonic and the fact that edge labels are identical in the subtrees rooted at $p_{r(N)}$ and $p_{l(N)}$

Note that in T_j, all node labels at levels greater than j are the same as their value in T. Thus, by Fact 2, every node J in level j of T_j is such that $p_{r(J)} \geq p_{l(J)}$. However, by Fact 1, when the edge labels between levels j and $j+1$ are changed from their values in T_j to their values in T_{j+1}, this cannot decrease the likelihood of going from J to $r(J)$. Thus, all node labels at level j of T_{j+1} are at least as large as the corresponding node label in T_j. It is easy to show by induction that this implies that all node labels of T_{j+1} are at least as large as the corresponding node label in T_j. ∎

This lemma and lemma 3.1 give us the following:

Theorem 3.6 *Let $\mathcal{A}$ be an algorithm for A, an $\sqrt{n} \times \sqrt{n}$ faulty array with each processor faulty independently with probability r, for some constant r, where $\mathcal{A}$ is $(p, k, \frac{n}{2})$-guaranteed by a set of monotonic array properties. With probability at least $1 - kp$, a random placement of $O(n)$ wireless nodes into a square domain space can simulate algorithm $\mathcal{A}$ with constant factor slowdown per step of $\mathcal{A}$.*

So, for example, we can use the results of [24] to obtain the following:

Corollary 3.7 *With probability $1 - O(1/n)$, a placement of n wireless nodes uniformly and independently at random into the domain space can perform any of the following in time $O(\sqrt{n})$:*
- *Route an arbitrary on-line permutation between the n nodes.*
- *Sort n keys.*
- *Multiply two $\sqrt{n} \times \sqrt{n}$ matrices.*

The algorithms for all of these are deterministic and require constant sized queues.

Proof. [24] provides algorithms that run in time $O(\sqrt{n})$ for all of these problems for the $\sqrt{n} \times \sqrt{n}$ faulty array. They show that these algorithms work correctly, provided that the array is $\sqrt{n}/4$-*gridlike*. An array is r-gridlike if every $r \times r$ subarray has at least $\frac{2}{3}r$ fault-free paths connecting the left and right sides of the subarray and at least $\frac{2}{3}r$ fault-free paths connecting the top and bottom of the subarray. Being r-gridlike is a monotonic array property, and is also decomposable: if the processors in columns 0 through $\frac{\sqrt{n}}{2}$ are r-gridlike, the processors in columns $\frac{\sqrt{n}}{4}$ through $\frac{3\sqrt{n}}{4}$ are r-gridlike, and the processors in columns $\frac{\sqrt{n}}{2}$ through $\sqrt{n}$ are r-gridlike, then the the entire array is r-gridlike, for $r < \frac{\sqrt{n}}{4}$. In [24], they also prove the following:

Theorem 3.8 *[24] A $\sqrt{n}\times\sqrt{n}$ array, where each processor is faulty with probability p, is $\frac{logn}{\log 1/p}$-gridlike with probability at least $1-\frac{1}{n}$.*

Thus the algorithms for the three problems are $(\frac{1}{n}, 3, \frac{n}{2})$-guaranteed by a set of monotonic array properties. For the routing result, the algorithm of [24] is only able to route permutations consisting of pairs of processors connected by a path of fault-free processors. However, we can use the extra power of wireless communication to route **any** permutation between all n nodes. We do this by using the $\frac{n}{\log^2 n}$-partition of the domain space. Call each square of the $\frac{n}{\log^2 n}$-partition a super-region. Using standard Chernoff bound techniques, we can show that every super-region has at most $O(\log^2 n)$ nodes w.h.p. It is shown in [24] that in any $\log n \times \log n$ region of the faulty array, there are $\Omega(\log^2 n)$ *active* processors, where each active processor is connected by a fault-free path to every other active processor. Thus, Lemma 3.1, gives us that in time $O(\log^2 n)$, we can redistribute the messages within each super-region so that every message is at an active node, and each active node has $O(1)$ messages. ∎

3.1.2 Arbitrary Convex Regions

We also demonstrate that these techniques can be extended to route a random permutation in time that is asymptotically optimal when the nodes are distributed independently and uniformly at random into any 2-dimensional convex region. In such a domain space, a Z-partition is defined as follows. Find the longest chord of the domain space, called the major axis, and the longest chord perpendicular to the major axis, called the minor axis. The Z-partition is the set of Z abutting squares with largest side length that fit entirely into the domain space, where the squares are aligned with the major and minor axes. The width of the domain space is defined to be the number of squares in the n-partition along the major axis, and the height is the number along the minor axis. As a first step, we consider the problem of nodes distributed randomly into any rectangular region.

Lemma 3.9 *When n nodes are distributed uniformly at random in a rectangular domain space with height H and width $\frac{n}{H}$, w.h.p. a random permutation can be routed in time $O(\frac{n}{H}\min[H, \lceil\frac{\log n}{H}\rceil])$.*

Proof. We use the n-partition of the rectangular domain space, and the corresponding faulty array. When $H \geq \log n$, the routing algorithm follows directly from the techniques of the previous subsection, since the rectangle can be shown to be $\log n$-gridlike. Also, when $H \leq \sqrt{\log n}$, each message can be sent directly from source to destination in the stated running time. Thus, the interesting case is when $\sqrt{\log n} < H < \log n$.

In this case, we use the $\frac{nH^2}{\log^2 n}$-partition of the rectangular domain space. Call each square of this partition a super-region. Each of these super-regions consists of a square of $\frac{\log n}{H} \times \frac{\log n}{H}$ regions of the n-partition. Let S' be the distribution of the nodes into the $\frac{nH^2}{\log^2 n}$-partition, and let $f(S')$ be the corresponding faulty $\frac{n}{\log n} \times \frac{H^2}{\log n}$ array, where a processor of $f(S')$ is fault-free iff the corresponding super-region of S' contains at least one node. When each region of the n-partition has a node in it independently with probability $\frac{1}{3}$, then each super-region has at least one node in it with probability $1-p$, where $p = (\frac{2}{3})^{\frac{\log^2 n}{H^2}}$. Let A' be the faulty $\frac{n}{\log n} \times \frac{H^2}{\log n}$ array, where each processor is faulty independently with probability p.

By Theorem 3.8, array A' is $\frac{\log n}{\log 1/p} = \frac{H^2}{\log n}$-gridlike w.h.p. Thus, A' can route a one-item per fault-free processor routing problem in time $O(\frac{n}{\log n})$. By Lemma 3.4, the array $f(S')$ is also $\frac{H^2}{\log n}$-gridlike, and thus by Lemma 3.1, the nodes in S' can also perform, in time $O(\frac{n}{\log n})$, any routing problem where there is at most one packet for each super-region that contains at least one node. However, the distribution of wireless nodes actually has an average of $\frac{\log^2 n}{H^2}$ packets per super-region. Since every set of $\frac{H^2}{\log n}$ super-regions has $O(\log n)$ packets w.h.p., the packets can be redistributed in time $O(\log n)$ so that in fact each super-region has $O(\frac{\log^2 n}{H^2})$ packets. Thus, by solving $O(\frac{\log^2 n}{H^2})$ separate routing problems, the total time required is $O(\frac{n \log n}{H^2})$. ∎

We now turn to the case of domain spaces that are not rectangles.

Theorem 3.10 *For n nodes distributed uniformly and independently at random into a convex domain space with height H, there is an algorithm that, with probability $1-O(1/n)$, routes a random permutation in time $O(\frac{n}{H}\min[H, \lceil\frac{\log n}{H}\rceil])$.*

Proof. Any node that is not contained in a region of the n-partition uses as an intermediary a node in the partition. Routing to and from such nodes requires time $O(\log n)$, and thus we here assume that every node is contained in a region of the partition. This only effects the analysis by constant factors.

When $H \geq \log^2 n$, we use super-regions consisting of $\log n \times \log n$ squares that are all contained in the domain space. The only regions of the n-partition that are more than $\log n$ regions away from such a super-region are the regions at the extreme end-points of the major axis. By the convexity assumption, the number of such regions is $O(\frac{n \log^2 n}{H^2})$, and thus routing directly to and from the nodes in those regions can be performed in time $O(\frac{n}{H})$. For the remainder of the nodes, we route to the nearest node in a super-region by routing either first to the correct row and then to the correct column, or visa versa (depending on which path does not leave the domain space). Then, from there, the packet is forwarded to the correct node.

In the case that $H = O(\log^2 n)$, the number of nodes that are not within distance $\log n$ of a super-region is high enough that we need to be more careful. We partition the domain space as follows: where the local height of the space is $> \log n$, we partition the space into the same $\log n \times \log n$ super-regions as before. When the local height is $< \log n$, we partition the domain space into rectangles. At any point along the major axis, there is at most one such rectangle. By the convexity assumption, there are at most two portions of the domain space where the local height is between 2^i and 2^{i+1}, for any integer i. Each of these portions consists of one rectangle of height 2^i. Note that if we examine the arrangement of rectangles along the major axis, then this results in rectangles of increasing size until we possibly reach a portion of the domain space of height $> \log n$, after which the rectangles will be of decreasing size.

We here describe how to route packets that travel in the direction of one endpoint of the major axis (called without loss of generality the left endpoint of the domain space). We assume that the packets start at uniformly distributed nodes in the highest point of the domain space. With the ability to do this, we can route any permutation of packets, since routing in the opposite direction is just the reverse process.

We number the rectangles between the left endpoint and the highest point of the domain space from smallest to largest: the jth smallest rectangle is R_j, and let R_m, $m = \min[\log H, \log\log n]$ be the largest such rectangle. By using our results for rectangular domain spaces, there are $\Theta(\lceil\frac{2^{2j}}{\log n}\rceil)$ paths in rectangle R_j from the left edge to the right edge. There are 4 times as many paths in R_{j+1} as in R_j. These paths can be connected so that every path in R_j is connected to exactly 4 paths in R_{j+1}, and every path in R_{j+1} is connected to a path in R_j. To do this, we assume that the width of the rectangle is $\Omega(\log n)$, but when the width of any rectangle R_j is $o(\log n)$, then by the convexity assumption, the number of nodes in the rectangles R_i, $i \leq j$ is at most $o(\log^2 n)$, which means that any packets destined for these nodes can be delivered directly to their destinations in the stated time.

With such a set up of paths, we route as follows. Packets are sent using the previous faulty array like routing techniques to the rectangle R_m, and are then sent along a randomly chosen path of the

$\lceil \frac{2^{2m}}{\log n} \rceil$ paths in R_m. Each packet is routed along the chosen path to the correct column of the domain space. Since 4 paths in rectangle R_{j+1} connect to a single path in rectangle R_j, there may be conflicts between two packets trying to use the same path. When there is a conflict between two packets, the packet that has the furthest to go is sent first. After all the packets have reached the correct column, the packets are sent directly to their destinations.

It is easily shown that the total number of times a packet is transmitted is at most $\frac{n}{H}$. It only remains to show that the packets are delayed by conflicts for the same path for at most $O(\frac{n}{H}\min[H, \lceil \frac{\log n}{H} \rceil])$ steps. Consider a packet that is destined for rectangle R_j. It can only be delayed by packets on their way to rectangles R_k, $k \leq j$. By the convexity argument, the number of regions in R_j is at most $O(2^j \cdot \frac{2^j n}{H^2})$. This implies that the total number of regions in rectangles R_k, $k \leq j$ is at most $O(\frac{2^{2j} n}{H^2})$.

W.h.p. the number of packets bound for these rectangles (and hence the number of packets that pass into rectangle R_j) is also $O(\frac{2^{2j} n}{H^2})$. Since each of these packets passes along a randomly chosen path in R_j, the number of packets traveling along each path is w.h.p. $O(\frac{2^{2j} n}{H^2}\min[1, \frac{\log n}{2^{2j}}])$. This means that any packet destined for R_j can be delayed at most $O(\frac{2^{2j} n}{H^2}\min[1, \frac{\log n}{2^{2j}}])$ steps, and this is $O(\frac{n}{H}\min[H, \lceil \frac{\log n}{H} \rceil])$, for any value of j and H. ∎

3.2 Lower bounds

In this section we address the question of how much additional power is provided by the ability to be able to communicate directly with nodes that are far away. The wireless scenario might be faster than an array, since a wireless transmission can send any single message from source to destination in a single step. The cost of this, however, is that such a transmission can block many other transmissions from occurring. In fact, we demonstrate that in the Euclidean case, the use of such long transmissions does not help for most permutations, and in fact the results of the previous section are within a constant factor of optimal. We assume that packets to be transmitted are indivisible (i.e., that the algorithm does not encode or duplicate the content of the packets), and each transmission by a node can forward at most one packet.

Theorem 3.11 *For n nodes placed uniformly and independently at random in a square domain space, the time to route a random permutation is w.h.p. $\Omega(\sqrt{n})$.*

Proof. In the following, let (xy) denote the distance from node x to node y, where the unit of distance is defined to be the side length of a square of the n-partition. In this proof, in come cases, instead of using the distance between two points, we use the distance between the centers of the respective regions of the n-partition. Thus, we start with a claim that bounds the difference in distance between these two measures of distance.

Claim 3.12 *For $d > \sqrt{2}$, let $\beta_d = 1 - \frac{\sqrt{2}}{d}$. Let x and y be any two points that are at least distance d apart, and let $\bar{x}$ be the exact center of the region containing x.*

$$(xy) \geq \beta_d(\bar{x}y)$$

and

$$(\bar{x}y) \geq \beta_d(xy)$$

Let k be the minimum integer such that $\frac{1}{\beta_k} \leq \sqrt[3]{\alpha}$. Note that since α is a constant > 1, k is also a constant. For this k, we see that if node a transmits to node a' on a step where node b' also receives a message, where $(aa') \geq k$ and $(ab') \geq k$, then $(\bar{a}b') > \sqrt[3]{\alpha}(\bar{a}a')$.

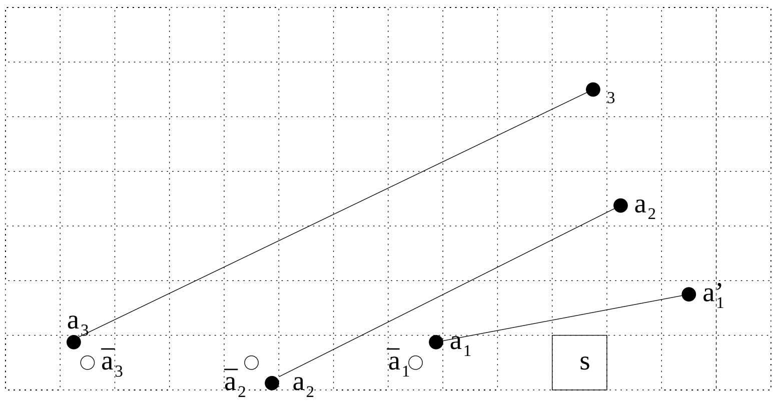

Figure 1: Region s is occupied by three messages.

Definition 3.13 *We say that region r_{ij} is* k-occupied *at time t by message m if message m is at a node in region r_{il}, $l + k < j$ at time $t - 1$, and message m is at a node in region r_{xy}, $y \geq j$ at time t.*

Thus, every time a message moves r regions to the right, $\max[r - k, 0]$ regions are k-occupied. This holds regardless of how many columns a packet moves during any step, and also regardless of how many rows the packet moves up or down during any step. Let o be the total number of regions that are k-occupied by messages over the course of a routing algorithm. Let p be the total number of transmissions that move a packet somewhere between 1 and k regions to the right. In a random permutation, the total, over all packets, of the number of columns moved from left to right is w.h.p. $\Omega(n\sqrt{n})$. However, when a message moves ℓ columns to the right, if $\ell \leq k$, the contribution of this step to p is 1, and if $\ell > k$, the contribution of this step to o is at least $\ell - k$. Thus, each column moved to the right by any message contributes at least $\frac{1}{k+1}$ to the sum $o + p$, Thus, in a random permutation, $o + p = \Omega(n^{3/2})$. This means that $\max[p, o] = \Omega(n^{3/2})$. If $p = \Omega(n^{3/2})$, the theorem follows directly from the fact that at most n messages can be sent at any step. Otherwise, the following lemma suffices.

Lemma 3.14 *For any time step t and for any region r, at most $O(1)$ messages can k-occupy r at step t.*

Proof. We assume that at some time step, nodes $a_1, a_2, \ldots a_j$ each successfully transmit to nodes $a'_1, a'_2, \ldots a'_j$ respectively, and they all k-occupy some region s. We shall see that j is bounded by a constant. The nodes are ordered so that a_1 is the closest to s and a_j is the furthest. Let $\bar{a_i}$ be the midpoint of the region containing a_i. Let $\bar{b}$ be the midpoint of the region containing b. This is depicted in Figure 3.2.

By the definition of k-occupy, $(a_h a'_i) \geq k$ for all h, i and so for all $h \neq i$, $(\bar{a_h} a'_i) \geq \sqrt[3]{\alpha}(\bar{a_h} a'_h)$. Thus,

$$(\bar{a_1} a'_2) > \sqrt[3]{\alpha}(\bar{a_1} a'_1) \tag{4}$$

$$(\bar{a_2} a'_1) > \sqrt[3]{\alpha}(\bar{a_2} a'_2) \tag{5}$$

Equation (5) implies

$$(\bar{a_2}\bar{a_1}) + (\bar{a_1} a'_1) > \sqrt[3]{\alpha}(\bar{a_2} a'_2) \tag{6}$$

Multiplying (6) by $\sqrt[3]{\alpha}$, and using (4), we obtain

$$\alpha^{1/3}(\bar{a}_2\bar{a}_1) + (\bar{a}_1 a_2') > \alpha^{2/3}(\bar{a}_2 a_2') \tag{7}$$

However, since the angle $\bar{a}_2\bar{a}_1 a_2'$ must be obtuse, we see that $(\bar{a}_2 a_2') > (\bar{a}_2\bar{a}_1)$, and $(\bar{a}_2 a_2') > (\bar{a}_1 a_2')$. Thus, when $\alpha^{2/3} > \alpha^{1/3} + 1$, we see that

$$\alpha^{1/3}(\bar{a}_2\bar{a}_1) + (\bar{a}_1 a_2') < \alpha^{2/3}(\bar{a}_2 a_2').$$

Since this contradicts (7), we see that when $\alpha^{2/3} > \alpha^{1/3} + 1$, j can be at most 1.

When α is smaller, we boost the effective value of α in the above equations as follows. Since $(\bar{a}_3 a_2') \geq \sqrt[3]{\alpha}(\bar{a}_3 a_3')$, we have that $(\bar{a}_3\bar{a}_2) + (\bar{a}_2 a_2') \geq \sqrt[3]{\alpha}(\bar{a}_3 a_3')$. If we multiply this by $\alpha^{2/3}$ and use (7) to substitute for $\alpha^{2/3}(\bar{a}_2 a_2')$, we obtain

$$\alpha^{2/3}(\bar{a}_3\bar{a}_2) + \alpha^{1/3}(\bar{a}_2\bar{a}_1) + (\bar{a}_1 a_2') > \alpha^{2/3}(\bar{a}_3 a_3').$$

A simple geometric argument gives us that $(\bar{a}_1 a_3') > (\bar{a}_1 a_2')$, and thus we have

$$\alpha^{2/3}(\bar{a}_3\bar{a}_2) + \alpha^{1/3}(\bar{a}_2\bar{a}_1) + (\bar{a}_1 a_3') > \alpha(\bar{a}_3 a_3').$$

Iterating this same argument i times, we can show that for any $i \leq j$,

$$\sum_{\ell=1}^{i-1} \alpha^{\ell/3}(a_{\overline{\ell+1}}\bar{a}_\ell) + (\bar{a}_1 a_i') > \alpha^{i/3}(\bar{a}_i a_i').$$

This in turn implies that

$$\alpha^{(i-1)/3}(\bar{a}_i\bar{a}_1) + (\bar{a}_1 a_i') > \alpha^{i/3}(\bar{a}_i a_i'). \tag{8}$$

However, we again have that angle $\bar{a}_i\bar{a}_1 a_i'$ is obtuse, and thus that $(\bar{a}_i a_i') > (\bar{a}_i\bar{a}_1)$, and $(\bar{a}_i a_i') > (\bar{a}_1 a_i')$. For any constant $\alpha > 1$, there is a constant I such that $\alpha^{I/3} > 1 + \alpha^{(I-1)/3}$. For $i \geq I$, it must be the case that

$$\alpha^{(i-1)/3}(\bar{a}_i\bar{a}_1) + (\bar{a}_1 a_i') < \alpha^{i/3}(\bar{a}_i a_i').$$

Since this contradicts (8), it must be the case that $j < I$, which completes the proof of the lemma. We also point out that a more complicated argument using the law of cosines can be used to obtain much better constants. ■

■

We next turn to the case of nodes distributed independently and uniformly at random into any 2-dimensional convex region. As a first step, we consider the problem of nodes distributed randomly into any rectangular region.

Theorem 3.15 *If n nodes are distributed randomly in a rectangular domain space with height H, then the time required to route a random permutation is $\Omega(\frac{n}{H}\min[H, \lceil\frac{\log n}{H}\rceil])$, w.h.p.*

Proof. By using the technique of occupied regions used for the proof of the previous theorem, it is straightforward to show that the time required is $\Omega(\frac{n}{H})$, and thus we only describe the case where $H \leq \log n$. The bound for this case follows from the following:

Claim 3.16 *There exists a column c between columns $\frac{n}{4H}$ and $\frac{3n}{4H}$ of the n-partition such that w.h.p. at most $\lceil\frac{H^2}{\log n}\rceil$ packets can cross column c at any time step.*

Proof. When $H < \log n$, w.h.p. there is between columns $\frac{n}{4H}$ and $\frac{3n}{4H}$ a set of $\frac{\log n}{4H}$ consecutive columns that do not contain any nodes. To see this, we partition the columns into sets of $\frac{\log n}{4H}$ consecutive columns. Next, we ask the question of how many of these sets between columns $\frac{n}{4H}$ and $\frac{3n}{4H}$ contain at least one node. We then place the nodes, one at a time, uniformly at random into the sets. By using the solution to the coupon collectors problem (see for example [29]), After n nodes have been placed, w.h.p., there is still some set that has not received any nodes.

When $H < \sqrt{\log n}$, at most a constant number of transmissions can be successful across this gap of $\frac{\log n}{4H}$ consecutive columns. When $\sqrt{\log n} < H < \log n$, no more than a constant number of transmissions can be successful across this gap in any set of $\frac{\log n}{H}$ consecutive rows, and thus the total number of transmissions that can be successful is at most $\frac{H^2}{\log n}$. ∎

Thus, when $H \leq \sqrt{\log n}$, the total time required for transmission is $\Omega(n)$, and when $\sqrt{\log n} < H < \log n$, the total time required is at least $\Omega(\frac{n \log n}{H^2})$. ∎

We next turn our attention to the case of an arbitrarily shaped convex domain space. Note that the convexity assumption gives us that the width W of the domain space is $\Omega(\frac{n}{H})$. We can then use essentially the same lower bound technique as for the case of a rectangular domain space to obtain the following.

Theorem 3.17 *If n nodes are distributed uniformly and independently at random in a convex domain space with height H, then the time required to route a random permutation is $\Omega(\frac{n}{H}\min[H, \lceil\frac{\log n}{H}\rceil])$, w.h.p.*

References

[1] N. Alon, A. Bar-Noy, N. Linial, D. Peleg. A lower bound for radio broadcast. *J. Comput. Syst. Sci.* **43**(2), pp. 290-298, 1991.

[2] N. Alon, F.R.K. Chung, R.L. Graham. Routing permutations on graphs via matchings. *SIAM J. Discr. Math.* **7**(3), pp. 513-530, 1994.

[3] R. Bar-Yehuda, O. Goldreich, A. Itai. On the time complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *J. Comput. and Syst. Sci.* **45**, pp. 104-126, 1992.

[4] R. Bar-Yehuda, A. Israeli, A. Itai. Multiple communication in multihop radio networks. *SIAM J. Comput.* **22**(4), pp. 875-887, 1993.

[5] S. Boztas. A robust multi-priority topology-independent transmission schedule for packet radio networks. *Inf. Proc. Letters* **55**, pp. 291-295, 1995.

[6] D. Brushi, M. Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distrib. Comput.* **10**, pp. 129-135, 1997.

[7] K.-C. Chen. Medium access control of wireless LANs for mobile computing. *IEEE Network* **8**(5), pp. 50-63, 1994.

[8] I. Chlamtac, A. Farago. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Trans. Networking* **2**, pp. 23-29, 1994.

[9] I. Chlamtac, S. Kutten. On broadcasting in radio networks – problem analysis and protocol design. *IEEE Trans. Commun.* **33**(12), 1985.

[10] I. Chlamtac, S. Pinter. A distributed nodes organization algorithm for channel access in a multihop dynamic radio network. *IEEE Trans. Comput.* **36**, pp. 728-737, 1987.

[11] I. Chlamtac, O. Weinstein. The wave expansion approach to broadcasting in multihop radio networks. *IEEE Trans. Commun.* **39**(3), pp. 426-433, 1991.

[12] I. Cidon, M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Trans. Comput.* **38**(10), pp. 1353-1361, 1989.

[13] R. Cole, B. Maggs, R. Sitaraman. Reconfiguring arrays with random faults. *Manuscript*, 1997.

[14] R. Cypher, F. Meyer auf der Heide, C. Scheideler, B. Vöcking. Universal algorithms for store-and-forward and wormhole routing. In *28th Ann. ACM Symp. on Theory of Computing*, pp. 356-365, 1996.

[15] S. Dolev, E. Kranakis, D. Krizanc, D. Peleg. Bubbles: Adaptive routing scheme for high-speed dynamic networks. In *Proc. of the 27th Ann. ACM Symp. on Theory of Computing*, pp. 528-537, 1995.

[16] B. Das, R. Sivakumar, V. Bharghavan. Routing in ad-hoc networks using a virtual backbone. To appear in *IEEE IC3N 97.*

[17] I. Gaber, Y. Mansour. Broadcast in radio networks. In *SODA '95*, pp. 577-582, 1995.

[18] R. Gallager. A perspective on multiaccess channels. *IEEE Trans. Inform. Theory*, Special Issue on Random Access Communications, **31**, pp. 124-142, 1985.

[19] *The Mobile Communications Handbook.* J.D. Gibson (Ed.), CRC Press, 1996.

[20] F.K. Hwang. the time complexity of deterministic broadcast radio networks. *Discr. Appl. Math.* **60**, pp. 219-222, 1995.

[21] *Mobile Computing*, T. Imielinski, H. Korth (Eds.), Kluwer Academic Publishers, 1996.

[22] J.M. Jacobsmeyer. Congestion relief on power-controlled CDMA networks. *IEEE J. Sel. Areas in Commun.* **18**(9), pp. 1758-1761, 1996.

[23] D.B. Johnson, D.A. Maltz. Protocols for adaptive wireless and mobile networking. *IEEE Personal Comm.* **3**(1), 1996.

[24] C. Kaklamanis, A.R. Karlin, F.T. Leighton, V. Milenkovic, P. Raghavan, S.B. Rao, C. Thomborson, A. Trantilas. Asymptotically Tight Bounds for Computing with Faulty Arrays of Processors. In *Proc. of the 1990 IEEE 31st Ann. Symp. on Foundations of Computer Science*, pp. 285-296, 1990.

[25] L.M. Kirousis, E. Kranakis, D. Krizanc, A. Pelc. Power consumption in packet radio networks. In *STACS '97.*

[26] E. Kushilevitz, Y. Mansour. An $\Omega(D\log(N/D))$ lower bound for broadcast in radio networks. In *PODC '93.*

[27] F.T. Leighton, B.M. Maggs, S.B. Rao. Universal packet routing algorithms. In *Proc. of the 29th Ann. Symp. on Foundations of Computer Science*, pp. 256-271, 1988.

[28] A. Myles, D.B. Johnson, C.E. Perkins. A mobile host protocol supporting route optimization and authentication. *IEEE J. Sel. Areas in Commun.*, special issue on Mobile and Wireless Computing Networks, **13**(5), pp. 839-349, 1995.

[29] *Randomized Algorithms*, R. Motwani and P. Rahgavan, Cambridge University Press, 1995.

[30] F. Meyer auf der Heide, C. Scheideler. Deterministic routing with bounded buffers: Turning offline into online protocols. In *Proc. of the 37th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 370-379, 1996.

[31] P. Piret. On the connectivity of radio networks. *IEEE Trans. on Inform. Theory* **37**(5), pp. 1490-1492, 1991.

[32] M.J. Post, P.E. Sarachik, A.S. Kershenbaum. A distributed evolutionary algorithm for reorganizing network communications. In *Proc. IEEE MILCOM*, 1985.

[33] Y. Rabani, É. Tardos. Distributed packet switching in arbitrary networks. In *28th Ann. ACM Symp. on Theory of Computing*, pp. 366-375, 1996.

[34] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. of Computer and System Sciences* **37**, pp. 130-143, 1988.

[35] R. Raghavan. Robust Algorithms for Packet Routing in a Mesh. In *Proc. of the 1st Ann. Symp. on Parallel Algorithms and Architectures*, pp. 344-350, 1989.

[36] K. Ravishankar, S. Singh. Asymptotically optimal gossiping in radio networks. *Discr. Appl. Math.* **61**, pp. 61-82, 1995.

[37] L.G. Roberts. *Aloha Packet System with and without Slots and Capture*, ASS Notes 8, Advanced Research Projects Agency, Network Information Center, Stanford Research Institute, Stanford, CA, 1972.

[38] A. Sen, M. Huson. A new model for scheduling packet radio networks. In *Proc. of the 15th Ann. Joint Conf. of the IEEE Computer and Communication Societies INFOCOM*, pp. 1116-1124, 1996.

[39] S. Ulukus, R.D. Yates. Stochastic power control for cellular radio systems. Dept. of Electrical and Computer Engineering, Rutgers University, NJ. Submitted to *IEEE Trans. Commun.*, September 1996.

[40] L.G. Valiant. A scheme for fast parallel communication. *SIAM J. on Comput.* **11**(2), pp. 350-361, 1982.

[41] O. Weinstein. *The wave expansion approach to broadcasting in multihop networks*, M.Sc. Thesis, Computer Science Dept., Technion, Haifa, Israel, 1987.

A Appendix

A.1 Proof of Theorem 2.12

A.1.1 The uniform case

We first consider the uniform case, that is, all edges have the same latency. Let $\mathcal{S}$ be the offline protocol for $\bar{G}$. We construct from $\mathcal{S}$ a protocol $\mathcal{S}'$ for G that works as follows.

At any time step, a packet waiting to traverse an edge e is given as *rank* the difference between the actual time step and the time step it is supposed to traverse e according to $\mathcal{S}$. If more than one packet want to use the same edge at the same time, then the one with highest rank is preferred for trials to cross it, and the others are *blocked*. (Note that no two packets can have the same rank at the same time at some node.)

In the following we bound the time $\mathcal{S}'$ needs, w.h.p., to simulate $\mathcal{S}$ in G. Let us assume that, for all edges e, $\varphi(e) = \frac{1}{\ell}$ for some fixed $2 \leq \ell \leq L$. Our aim will be to bound the worst case distribution over all ranks of the expected number of packets with certain rank at any time step during the simulation.

Let us first introduce some notation. For any two functions $f_1, f_2 : \mathbb{Z} \to \mathbb{R}_{\geq 0}$, let us say that f_1 *dominates* f_2 (or $f_1 \triangleright f_2$) if, for every $i \in \mathbb{Z}$, $\sum_{j \geq i} f_1(j) \geq \sum_{j \geq i} f_2(j)$. We define the *rank distribution* of the N packets at some time step as $r : \mathbb{Z} \to \{0, \ldots, N\}$, where $r(i)$ denotes the current number of packets with rank i, that is, $\sum_i r(i) = N$. (Note that ranks of packets can be negative if packets are ahead of their schedule.) Define the *delay* of a packet at node v to be the current time step minus the dilation of the path it traversed so far. Clearly, during the whole simulation the delay of a packet is an upper bound on its rank. That is, if r denotes the current rank distribution and d denotes the current delay distribution of the packets then, for every $i \in \mathbb{Z}$, $\sum_{j \geq i} d(j) \geq \sum_{j \geq i} r(j)$. Hence, d dominates r. In order to make the proof simpler in the following, assume that any delay change is by ±1 (in reality, the delays are a multiple of ℓ apart), and any rank change is compressed accordingly.

Let r_0 denote the initial rank distribution and d_0 denote the initial delay distribution of the packets for the simulation of $\mathcal{S}$ by G. Then we define $E_t[r_0]$ to be the expected rank distribution after t time steps of the simulation. Let $\hat{E}_t[d_0]$ denote the expected delay distribution after t time steps, assuming MAXDELAY at each time step, where MAXDELAY is defined as follows:

MAXDELAY: Given a delay distribution d consider, for every i, $\min[\sum_{j>i} d(j), d(i)]$ of the $d(i)$ packets with delay i to be blocked from transmissions.

For every $t \in \mathbb{N}$, $\hat{E}_1^t[d]$ is defined as $\hat{E}_1[\hat{E}_1^{t-1}[d]]$ and $\hat{E}_1^0[d] = d$. Our aim is to show that $\hat{E}_1^t[d_0] \triangleright E_t[r_0]$. This would enable us to compute an upper bound for the number of packets with ranks greater than some given rank by applying $\hat{E}_1$ t times to d_0, which is significantly easier to calculate than $E_t[r_0]$. For this, we show the following lemma.

Lemma A.1 *For every time step $t \in \mathbb{N}_0$, $\hat{E}_1^t[d_0] \triangleright E_t[r_0]$.*

Proof. We will show by complete induction that, for all $t \in \mathbb{N}_0$, $\hat{E}_1^t[d_0] \triangleright E_t[r_0]$. Clearly, for $t = 0$ the relation is true. Suppose now that the relation holds for some $t \in \mathbb{N}_0$. Let $d = \hat{E}_1^t[d_0]$ and $d' = \hat{E}_1[d]$. Since in case of MAXDELAY an expected amount of $\frac{1}{\ell}(d(i) - \min[d(i), \sum_{j>i} d(j)])$ packets is moved from $d(i)$ to $d(i-1)$ for every $i \in \mathbb{Z}$, we have

$$d'(i) = d(i) - \tfrac{1}{\ell}(d(i) - \min[d(i), \sum_{j>i} d(j)]) + \tfrac{1}{\ell}(d(i+1) - \min[d(i+1), \sum_{j>i+1} d(j)]) \ .$$

Hence, we get for all $i \in \mathbb{Z}$

$$\begin{aligned}
\sum_{j \geq i} d'(j) &= \sum_{j \geq i} \left(d(j) - \tfrac{1}{\ell}(d(j) - \min[d(j), \sum_{k>j} d(k)]) + \right. \\
&\qquad \left. \tfrac{1}{\ell}(d(j+1) - \min[d(j+1), \sum_{k>j+1} d(k)]) \right) \\
&= \sum_{j \geq i} d(j) - \tfrac{1}{\ell} d(i) + \tfrac{1}{\ell} \min[d(i), \sum_{j>i} d(j)] \ . \qquad (9)
\end{aligned}$$

Let $P = \{h : [N] \to \mathbb{N}_0\}$ denote the set of all possible assignments of positions to packets in a way that, given an $h \in P$, $h(i)$ denotes the number of edges packet i has already passed. Let the random variable X_t be $h \in P$ if and only if the outcome after t steps of the simulation is h. Given a $h \in P$, let r_h be the corresponding rank distribution. Furthermore, let $r_h^-(i)$ for every $i \in \mathbb{Z}$ denote the number of packets with rank i that are allowed to attempt transmissions, and let $r_h^+(i)$ for every $i \in \mathbb{Z}$ denote the number of packets with rank larger than i that are allowed to attempt transmissions and, in case of success, get rank i. Then it holds for every $i \in \mathbb{Z}$:

$$\begin{aligned}
E_{t+1}[r_0](i) &= \sum_{h \in P} \Pr[X_t = h] \cdot (r_h(i) + \tfrac{1}{\ell} r_h^+(i) - \tfrac{1}{\ell} r_h^-(i)) \\
&= E_t[r_0](i) + \tfrac{1}{\ell} \sum_{h \in P} \Pr[X_t = h] \cdot (r_h^+(i) - r_h^-(i)) \ .
\end{aligned}$$

Summing up over all $j \geq i$ yields

$$\sum_{j \geq i} E_{t+1}[r_0](j) = \sum_{j \geq i} E_t[r_0](j) + \tfrac{1}{\ell} \sum_{j \geq i} \sum_{h \in P} \Pr[X_t = h] \cdot (r_h^+(j) - r_h^-(j))$$

As it is easy to see, $\sum_{j \geq i} \sum_{h \in P} \Pr[X_t = h] \cdot \frac{1}{\ell}(r_h^+(j) - r_h^-(j))$ represents the expected number of all packets that change from a rank at least i to a rank lower than i after step $t+1$. This number is minimal if we assume that all ranks can only be reduced by one and, for every $h \in P$, $\min[r_h(i), \sum_{j>i} r_h(j)]$ of the packets with rank i are blocked. (Note that the offline schedule $\mathcal{S}$ ensures that each node can only have one packet with rank i at any time, so packets with the same rank cannot block each other.) Hence, together with the fact that for any two sequences $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ of real numbers it holds that $\sum_i \min[a_i, b_i] \leq \min[\sum_i a_i, \sum_i b_i]$, we obtain

$$\begin{aligned}
\sum_{j \geq i} E_{t+1}[r_0](j) &\leq \sum_{j \geq i} E_t[r_0](j) - \tfrac{1}{\ell} \sum_{h \in R} \Pr[X_t = h] \cdot (r_h(i) - \min[r_h(i), \sum_{j>i} r_h(j)]) \\
&= \sum_{j \geq i} E_t[r_0](j) - \tfrac{1}{\ell} E_t[r_0](i) + \\
&\qquad \tfrac{1}{\ell} \sum_{h \in R} \Pr[X_t = h] \cdot \min[r_h(i), \sum_{j>i} r_h(j)] \\
&\leq \sum_{j \geq i} E_t[r_0](j) - \tfrac{1}{\ell} E_t[r_0](i) + \\
&\qquad \tfrac{1}{\ell} \min\left[\sum_{h \in R} \Pr[X_t = h] \cdot r_h(i), \sum_{h \in R} \Pr[X_t = h] \cdot \sum_{j>i} r_h(j) \right] \\
&= \sum_{j \geq i} E_t[r_0](j) - \tfrac{1}{\ell} E_t[r_0](i) + \tfrac{1}{\ell} \min[E_t[r_0](i), \sum_{j>i} E_t[r_0](i)] \\
&= \sum_{j \geq i} \hat{E}_1[E_t[r_0]](j) \ . \qquad (10)
\end{aligned}$$

It remains to be shown that (9) $\geq$ (10) for all i. Let $r = E_t[r_0]$. According to the induction hypothesis, $d \rhd r$. Clearly, from any delay distribution d any rank distribution r that is dominated by d can be constructed by moving units from position i to position $i-1$ for some $i \in \mathbb{Z}$ again and again. (Note that units cannot be added or deleted, since every distribution function f has to fulfill $\sum_i f(i) = N$.) For the case that d is equal to r the claim immediately follows, since ranks of successful packets are reduced by at least as much as their delays. In all other cases we can restrict ourselves to considering two delay distributions d_1 and d_2, where d_2 results from d_1 by moving an ϵ-unit from position i to $i-1$ for some $i \in \mathbb{Z}$. Let $d_1' = \hat{E}_1[d_1]$ and $d_2' = \hat{E}_1[d_2]$. We claim that $\sum_{k \geq j} d_2'(k) \leq \sum_{k \geq j} d_1'(k)$ for all $j \in \mathbb{Z}$. For this, we have to consider some cases.

- For all $j < i-1$ and $j > i$ it clearly holds that $d_1(j) = d_2(j)$ and $\sum_{k>j} d_1(k) = \sum_{k>j} d_2(k)$. Hence, $\sum_{k \geq j} d_1'(k) = \sum_{k \geq j} d_2'(k)$, which fulfills the claim for these j.
- For $j = i$ it holds that $d_1(j) = d_2(j) + \epsilon$ and $\sum_{k>j} d_1(k) = \sum_{k>j} d_2(k)$. Hence, $\sum_{k \geq j} d_1'(k) \geq \sum_{k \geq j} d_2'(k)$, which also fulfills the claim for this j.
- For $j = i-1$, we have $d_2(j) = d_1(j) + \epsilon$ and $\sum_{k>j} d_2(k) = \sum_{k>j} d_1(k) - \epsilon$. If $d_1(j) \leq \sum_{k>j} d_1(k)$ then, clearly, $\sum_{k \geq j} d_1'(k) \geq \sum_{k \geq j} d_2'(k)$. It therefore remains to consider the case $d_1(j) > \sum_{k>j} d_1(k)$. Let $\delta = d_1(j) - \sum_{k>j} d_1(k)$. Then $\sum_{k \geq j} d_1'(k) = \sum_{k \geq j} d_1(k) - \delta/\ell$. On the other hand,
$$d_2(j) - \sum_{k>j} d_2(k) = (d_1(j) + \epsilon) - (\sum_{k>j} d_1(k) - \epsilon) = \delta + 2\epsilon \ .$$
Hence,
$$\sum_{k \geq j} d_2'(k) = \sum_{k \geq j} d_2(k) - (\delta + 2\epsilon)/\ell = \sum_{k \geq j} d_1(k) - (\delta + 2\epsilon)/\ell \leq \sum_{k \geq j} d_1'(k) \ .$$
Thus, the claim is also true for $j = i-1$.

This completes the induction step, which proves the lemma. ■

Lemma A.1 implies that we can use the same assumption for the delays of the packets as for their ranks: Packets can only be blocked by packets with higher delays. (Recall that packets can only be blocked by packets with higher ranks, since the offline protocol $\mathcal{S}$ guarantees that, for any node v and any rank r, there can be at most one packet at v at any time step with rank r.) Furthermore, the lemma implies that in order to have an upper bound for $\sum_{j \geq i} E_t[r_0](j)$ at some time step t it remains to find an upper bound for $\sum_{j \geq i} \hat{E}_1^t[d_0](j)$.

Since we assume all edges e to have a success probability of $1/\ell$ for some fixed ℓ, every time a packet is successful, its delay is decreased by $\ell - 1$ and every time it is not successful, its delay is increased by one. Hence, at any time step, the difference between the delays of any two packets is an integer multiple of ℓ. In order to simplify the analysis for the distribution of the packets, let us consider instead of the delay the number of edges a packet already passed. Clearly, if at time step t a packet already passed s edges then its delay is $t - \ell \cdot s$.

For every time step t, let the distribution function $p_t : \mathbb{N}_0 \to [0,1]$ be defined such that, for all s, $p_t(s) \cdot N$ denotes the expected number of packets that have already passed s edges. Initially, $p_0(0) = 1$ and $p_0(s) = 0$ for all $s > 0$. In the next lemma, we show that p_t can be bounded as follows.

Lemma A.2 *Using MAXDELAY at every time step $t \geq 1$ it holds that, for all $s \geq 0$,*
$$p_t(s) \leq \frac{\left(\frac{2t}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^t}{s!} \ .$$

Proof. We show this by complete induction over t. It is easy to check that for $t = 1$ the bound is correct. So suppose that for some $t \geq 1$ it has already been shown that the bound above is correct. Then we show that it is also correct for $t+1$. For this we use the fact that for an upper bound we are allowed to apply $\hat{E}_1$ for any time step t. Let us consider the following cases in order to bound $p_{t+1}(s)$ for all $s \in \mathbb{N}_0$.

Case $p_t(s) \geq \sum_{s'<s} p_t(s')$ and $p_t(s-1) \geq \sum_{s'<s-1} p_t(s')$:
In this case, using MAXDELAY, an expected fraction of $\frac{1}{\ell}(p_t(s) - \sum_{s'<s} p_t(s'))$ is moved from $p_t(s)$ to $p_t(s+1)$, while an expected fraction of $\frac{1}{\ell}(p_t(s-1) - \sum_{s'<s-1} p_t(s'))$ is moved from $p_t(s-1)$ to $p_t(s)$. Hence, we get

$$\begin{aligned} p_{t+1}(s) &= p_t(s) - \tfrac{1}{\ell}(p_t(s) - \sum_{s'<s} p_t(s')) + \tfrac{1}{\ell}(p_t(s-1) - \sum_{s'<s-1} p_t(s')) \\ &= \Big(1 - \tfrac{1}{\ell}\Big)p_t(s) + \tfrac{2}{\ell}p_t(s-1) \ . \end{aligned}$$

Using the induction hypothesis, it follows that

$$\begin{aligned} p_{t+1}(s) &\leq \left(1 - \frac{1}{\ell}\right) \cdot \frac{\left(\frac{2t}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^t}{s!} + \frac{2}{\ell} \cdot \frac{\left(\frac{2t}{\ell-1}\right)^{s-1} \left(1 - \frac{1}{\ell}\right)^t}{(s-1)!} \\ &\leq \frac{\left(\frac{2t}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^{t+1}}{s!} \left(1 + \frac{s}{t}\right) \ . \end{aligned}$$

It holds that

$$\begin{aligned} \frac{\left(\frac{2t}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^{t+1}}{s!} \left(1 + \frac{s}{t}\right) &\leq \frac{\left(\frac{2(t+1)}{\ell-1}\right)^s \left(1 - \frac{1}{\ell}\right)^{t+1}}{s!} \\ \Leftrightarrow \quad 1 + \frac{s}{t} &\leq \left(1 + \frac{1}{t}\right)^s \ , \end{aligned}$$

which is clearly true for any $s \geq 0$ and $t \geq 1$. This completes the first case.

Case $p_t(s) < \sum_{s'<s} p_t(s')$ and $p_t(s-1) \geq \sum_{s'<s-1} p_t(s')$:
In this case, using MAXDELAY, an expected fraction of $\frac{1}{\ell}(p_t(s-1) - \sum_{s'<s-1} p_t(s'))$ is moved from $p_t(s-1)$ to $p_t(s)$. Hence, we get

$$p_{t+1}(s) = p_t(s) + \tfrac{1}{\ell}(p_t(s-1) - \sum_{s'<s-1} p_t(s')) \ .$$

This term is less than $(1 - \frac{1}{\ell})p_t(s) + \frac{2}{\ell}p_t(s-1)$ if and only if

$$\begin{aligned} \tfrac{1}{\ell}p_t(s) &< \tfrac{1}{\ell}p_t(s-1) + \tfrac{1}{\ell} \sum_{s'<s-1} p_t(s') \\ \Leftrightarrow \quad p_t(s) &< p_t(s-1) + \sum_{s'<s-1} p_t(s') \ , \end{aligned}$$

which is true since $p_t(s) < \sum_{s'<s} p_t(s')$. Thus, we can use the same calculation as above to prove the bound for $p_{t+1}(s)$ also in this case.

Case $p_t(s-1) < \sum_{s'<s-1} p_t(s')$:
Then it is easy to see that nothing can have reached $p_t(s)$ yet, that is, $p_t(s) = 0$. This completes the proof of Lemma A.2. ∎

Since each successful traversal of an edge by a packet corresponds to ℓ steps in $\mathcal{S}$, $d_t(s) = p_t(s/\ell) \cdot N$ corresponds to the expected number of packets that are at time step t of the simulation at step s in $\mathcal{S}$. Clearly, the simulation finishes if all packets are beyond step T in $\mathcal{S}$. It therefore remains to be shown that for $t = O(T + L \log N)$ this is the case w.h.p. For this, let us set $s = T$ and $t = (1+\epsilon)T$. If $\epsilon \geq 4$ then

$$\begin{aligned} p_t(s/\ell) = \frac{\left(\frac{2(1+\epsilon)T}{\ell-1}\right)^{T/\ell}\left(1-\frac{1}{\ell}\right)^{(1+\epsilon)T}}{(T/\ell)!} &\leq \frac{\left(\frac{T}{\ell-1}\right)^{T/\ell}}{\mathrm{e}^{T/\ell}(T/\ell)!}\left(\frac{2(1+\epsilon)}{\mathrm{e}^{\epsilon}}\right)^{T/\ell} \\ &\leq \left(\frac{\ell}{\ell-1}\right)^{T/\ell}\left(\frac{2(1+\epsilon)}{\mathrm{e}^{\epsilon}}\right)^{T/\ell} \\ &\leq \mathrm{e}^{\frac{T}{\ell}\left(\frac{1}{\ell-1}-\frac{\epsilon}{3}\right)} . \end{aligned}$$

Let the random variable X be defined as the number of packets at time step t with a rank of at least $t - T$. Using Lemma A.1, $\mathrm{E}[X]$ can be upper bounded by $\sum_{\delta \geq t-T} d_t(t-\delta)$. It is easy to check that, for all $0 \leq s \leq T/\ell$ and $t \geq T$, $p_t(s) \leq p_t(s+1)$. Hence, since during the simulation the delays of the packets are always a multiple of ℓ apart, for any constant $\alpha > 1$ there exists a constant ϵ such that

$$\mathrm{E}[X] = \sum_{\substack{t-T \leq \delta \leq t, \\ \ell \mid (t-\delta)}} d_t(t-\delta) = N \sum_{\substack{t-T \leq \delta \leq t, \\ \ell \mid (t-\delta)}} p_t\left(\frac{t-\delta}{\ell}\right) \leq \left(\frac{1}{N}\right)^{\alpha} .$$

Using the Markov inequality, it follows

$$\Pr[X \geq 1] \leq \left(\frac{1}{N}\right)^{\alpha} .$$

This concludes the proof of Theorem 2.12 for the uniform case.

A.1.2 The nonuniform case

Now, we show how to prove Theorem 2.12 also for the non-uniform case. Although we only show it for PCGs, it also holds for any transmission graph using a zone access scheme if we assume that at every time step all zones at all nodes participate in decisions to attempt hops, whether a packet has to be sent out for some zone or not. This ensures that the probability distribution for certain node–zone pairs to have or not to have success is the same throughout the whole simulation. We will note at specific places in our proof where we need this assumption.

Let $\mathcal{S}_1$ be an offline protocol for some DCG $\bar{G}$. First, we modify $\mathcal{S}_1$ so that it is easier to bound the time of simulating it. We start with transforming $\mathcal{S}_1$ into a schedule $\mathcal{S}_2$ in which all edge latencies are powers of 2 and every edge with latency ℓ only starts to send a packet at an integer multiple of ℓ. This can be done as follows:

Expand $\mathcal{S}_1$ by a factor of two in a way that an edge that previously forwarded a packet at time t now starts to forward it at time $2t$. Change the latency of every edge to the next highest power of two. Thus, the highest and the lowest edge latency, L and ℓ, are now powers of two. Clearly, the new schedule is still valid (in a sense that traversals of edges by a packet do not overlap in time), and the order of transmitting packets at the edges is preserved. Expand $\mathcal{S}_1$ again by a factor of two as before. Change the starting time of every traversal to the next highest time that is an integer multiple of the corresponding edge latency. Again it is easy to see that the resulting schedule is valid and the order of transmissions is preserved.

Next, transform $\mathcal{S}_2$ into a schedule $\mathcal{S}_3$ where all packets use paths of the same type p. This path p consists of a so-called (L, ℓ)-*sequence* s of $2L/\ell - 1$ edges that repeats itself again and again. The edge latency of edge $i \in \{1, \ldots, 2L/\ell - 1\}$ of the (L, ℓ)-sequence is $\ell \cdot j$, where j is the largest integer for which i is an integer multiple of 2^{j-1}. To give an example, for $\ell = 1$ and $L = 8$, s represents a sequence of edges with latencies

$$1, 2, 1, 4, 1, 2, 1, 8, 1, 2, 1, 4, 1, 2, 1.$$

It can be easily checked that the time to traverse all edges of the (L, ℓ)-sequence one after the other is equal to $L \log(2L/\ell)$. For each such sequence, $\mathcal{S}_3$ reserves a time interval of length $L \log(2L/\ell)$ so that disjoint time intervals can be assigned to the edges.

In order to transform $\mathcal{S}_2$ into $\mathcal{S}_3$, consider any time interval I of length L in $\mathcal{S}_2$ starting at time step t, where t is an integer multiple of L. Let us map $\mathcal{S}_2$ restricted to I to $\mathcal{S}_3$ restricted to the $(t/L + 1)$th occurance of s in p, denoted by s', in the following way: For every packet P and every edge e with latency ℓ' scheduled to be used by P with starting time t in I, schedule P in $\mathcal{S}_3$ to use e represented by the $(\frac{t}{\ell'} + 1)$th edge in s' with latency ℓ'. (This is possible, since t is at most $L - \ell'$ and an integer multiple of ℓ'.) As it is easy to check, the transformation ensures that

- traversals of different edges by the same packet and
- the traversal of the same edge by different packets

remain in order and do not overlap. In addition to this transformation, we fill all time intervals a packet is not scheduled to traverse an edge by traversals of the corresponding edges in p. For any packet P, these edges are called *imaginary edges*, whereas the edges already used in $\mathcal{S}_2$ are called *real edges*. So we end up with a schedule $\mathcal{S}_3$ in which all packets traverse the same type of path p for T/L occurences of s without waiting, where T is the runtime of $\mathcal{S}_2$.

$\mathcal{S}_3$ will be simulated by the PCG G of $\bar{G}$ in the following way: The traversal of any real edge e by some packet, where e corresponds to the original edge e' in $\bar{G}$, is simulated in $\bar{G}$ by attempts to cross the counterpart of e', e'', in G. If e has a larger latency than e', then we add an extra random experiment that decides whether to attempt to traverse e'' so that, together with the probability that the attempt fails for e'', we arrive at a probability of traversing e'' that is the inverse of the latency of e. (Recall that for transmission graphs using zone access schemes, all probabilites are already powers of 2. Hence, an offline protocol for its corresponding DCG does not require any change in the latency of the edges.) The traversal of any imaginary edge by some packet is simply simulated *internally* in the PCG, that is, the corresponding packet does not leave its current node. Obviously, once the PCG finishes $\mathcal{S}_3$ it has also finished $\mathcal{S}_1$. Hence, it remains to provide an upper bound for the time the PCG requires to simulate $\mathcal{S}_3$. For this we show the following lemma.

Lemma A.3 *For every time step $t \in \mathbb{N}_0$, $\hat{E}_t[d_0] \triangleright E_t[r_0]$.*

Proof. We present here a different proof than for Lemma A.1. The reason for this is that, in the uniform case, ranks are decreased always by at least as much as delays, no matter where some packet might be for two different simulations. This does not hold for the nonuniform case.

Let the rank and the delay of a packet during the course of the simulation be defined as for the uniform case. Since in $\mathcal{S}_3$ no packet is ever waiting at an edge, the following fact holds.

Fact A.4 *Schedule $\mathcal{S}_3$ has the property that, at any time during its simulation, the delay of a packet is always equal to its rank.*

To continue, we need some notation. For any two multisets D_1, D_2 of functions of the form $f : [N] \to \mathbb{Z}$ with the property that $|D_1| = |D_2|$ we say that D_1 *strongly dominates* D_2 (or $D_1 \triangleright_s D_2$) if there is a permutation $\pi : D_1 \to D_2$ such that for all $d \in D_1$ it holds that $d(i) \geq \pi(d)(i)$ (or $d \triangleright_s \pi(d)$).

Let $P_t[d_0]$ be defined as the probability distribution over all assignments of delays to the packets after t steps of the simulation. Similarly, let $\hat{P}_t[d_0]$ be defined as the probability distribution over all assignments of delays to the packets after t steps, assuming MAXDELAY at every step. However, instead of allowing to choose an arbitrary set of packets as in the uniform case, we will select a specific set of packets, depending on the situation. Since the success probability of any packet at any node is a rational number, we can transform $P_t[d_0]$ into a multiset $D_t[d_0]$ with the property that, for all assignments $d : [N] \to \mathbb{Z}$ of delays to the N packets, $P_t[d_0](d) =$ (number of times d appears in $D_t[d_0])/|D_t[d_0]|$. $\hat{D}_t[d_0]$ is defined accordingly, using $\hat{P}_t[d_0]$. Let $D_t[d_0]$ and $\hat{D}_t[d_0]$ be chosen such that they are of the same cardinality.

Our aim is to show by complete induction that, for all $t \in \mathbb{N}_0$, $\hat{D}_t[d_0] \rhd_s D_t[d_0]$. Clearly, if $\hat{D}_t[d_0] \rhd_s D_t[d_0]$ is true, then this would immediately imply that $\hat{E}_t[d_0] \rhd E_t[d_0]$. Since, by Fact A.4, during the whole simulation the delay of a packet is equal to its rank, $E_t[d_0] = E_t[r_0]$ and therefore the lemma would follow.

For $t = 0$ our claim is obviously true. Suppose now that $\hat{D}_t[d_0]$ strongly dominates $D_t[d_0]$ for some $t \in \mathbb{N}_0$. Let the permutation $\pi : \hat{D}_t[d_0] \to D_t[d_0]$ be defined such that, for all $d \in \hat{D}_t[d_0]$, $d \rhd_s \pi(d)$. Fix any $d \in \hat{D}_t[d_0]$, and let $d' = \pi(d)$. We will show that then $\hat{D}_1[d] \rhd_s D_1[d']$. From this the lemma would follow.

According to the definition of $\rhd_s$, for every packet P it holds that P w.r.t. d is not ahead of P w.r.t. d'. (Notice that once the delay of a packet is known at a specific time step, then also its position is known.) Clearly, all packets whose positions w.r.t. d are behind their positions w.r.t. d' imply $\hat{D}_1[d] \rhd_s D_1[d']$ restricted to these packets. So we only have to consider those packets whose positions are the same w.r.t. d and d'. As $\mathcal{S}_3$ ensures that there can be no two packets with the same delay (resp. rank) that intend to cross some common edge in G at the same time, MAXDELAY is the worst case that can happen for the packets placed according to d'. Since, due to $d \rhd_s d'$, $|\{j \mid d(j) \geq i\}| \geq |\{j \mid d'(j) \geq i\}|$ for all $i \in \mathbb{Z}$, all packets with equal positions w.r.t. d and d' that are blocked w.r.t. d' can also be blocked via MAXDELAY w.r.t. d. Hence, $\hat{D}_1[d] \rhd_s D_1[d']$ restricted to all packets that are blocked w.r.t. d'. For the remaining packets, we have the situation that w.r.t. d and d' they are allowed to perform access trials, that is, the probability distribution for any combination of these packets to be successfully transmitted is the same for both situations. (Note that we need here the assumption formulated at the beginning of the section that in a transmission graph all zones in all nodes participate in decisions for hop trials. Otherwise our results for PCGs cannot be transferred to the transmission graphs, since the probability distributions are not the same.) This implies that $\hat{D}_1[d] \rhd_s D_1[d']$ for all packets, which proves the lemma. ■

This lemma allows us to argue in the following only with $\hat{E}_t[d_0]$. For a final transformation from $\mathcal{S}_3$ to a schedule $\mathcal{S}_4$, we need the following lemma.

Lemma A.5 *Consider any schedule $\mathcal{S}$ in which all packets follow a path of the same type p. Let $\mathcal{S}'$ be a schedule in which the same number of packets follow a path of the same type p'. p' results from p by exchanging the positions of two consecutive edges in p, where the first edge has a lower latency than the second edge. Then $\hat{E}_1^t[d_0]$ w.r.t. $\mathcal{S}'$ dominates $\hat{E}_1^t[d_0]$ w.r.t. $\mathcal{S}$.*

Proof. Let us denote the two edges to be exchanged as e_1 and e_2, where e_1 be the ith edge of p. (W.l.o.g. we assume that $i \geq 2$. The case $i = 1$ can be prevented by inserting an edge with latency L in front of p.) Let ℓ_1 be the latency of e_1 and ℓ_2 be the latency of e_2, $\ell_1 < \ell_2$. Given a time step t, let $a^{(t)}, x^{(t)} \in \mathbb{R}^4$ be two vectors, where

- $a_0^{(t)}$ (resp. $x_0^{(t)}$) denotes the expected number of packets that have not traversed the $(i-1)$st edge in p (resp. p') yet,

- $a_1^{(t)}$ (resp. $x_1^{(t)}$) represents the expected number of packets that is currently at the ith edge of p (resp. p'),
- $a_2^{(t)}$ (resp. $x_2^{(t)}$) is the expected number of packets that is currently at the $(i+1)$st edge of p (resp. p'), and
- $a_3^{(t)}$ (resp. $x_3^{(t)}$) denotes the expected number of packets that has already passed the $(i+1)$st edge of p (resp. p').

See also the following Figure 2.

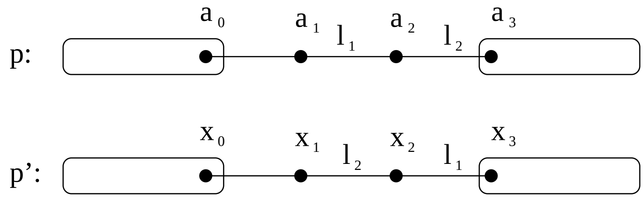

Figure 2: The positions of the a_i and x_i in the two paths p and p'

Clearly, it holds $a_0^{(0)} = x_0^{(0)} = 1$ and $a_j^{(0)} = x_j^{(0)} = 0$ for all $j \in \{1,2,3\}$. So for $t = 0$, $x^{(t)} \triangleright a^{(t)}$. For the remaining steps it remains to prove the following proposition.

Proposition A.6 *For any strategy of moving packets from $a_0^{(t)}$ to $a_1^{(t)}$ that is also used for moving packets from $x_0^{(t)}$ to $x_1^{(t)}$ at every time step $t \in \mathbb{N}$ it holds under MAXDELAY that $x^{(t)} \triangleright a^{(t)}$ for all $t \in \mathbb{N}$.*

Proof. For any $t \in \mathbb{N}$, let ϵ_t denote the amount of load moved from $a_0^{(t)}$ to $a_1^{(t)}$ (resp. from $x_0^{(t)}$ to $x_1^{(t)}$). As long as $a_1^{(t)} \leq a_0^{(t)}$ it clearly holds $a_i^{(t)} = x_i^{(t)}$ for all $i \in \{1,2,3\}$, so $x^{(t)} \triangleright a^{(t)}$. Let t_1 be the first time step for which $a_1^{(t_1)} > a_0^{(t_1)}$, and let t_2 be the first time step for which $a_2^{(t_2)} > a_0^{(t_2)} + a_1^{(t_2)}$. Since $\ell_1 < \ell_2$, it holds $a_2^{(t)} > x_2^{(t)}$ and $a_3^{(t)} = x_3^{(t)} = 0$ for all $t_1 < t \leq t_2$. Hence, $x^{(t)} \triangleright a^{(t)}$ also for $t_1 \leq t \leq t_2$.

For $t = t_2 + 1$, we have $a_3^{(t)} > 0$ for the first time. Let ϵ be chosen such that $a_3^{(t_2+1)} = \frac{4}{\ell_1 \ell_2}\epsilon$. This means that $a_2^{(t_2)} = a_0^{(t_2)} + a_1^{(t_2)} + \frac{4}{\ell_1}\epsilon$, and because of $a_2^{(t_2-1)} < a_0^{(t_2-1)} + a_1^{(t_2-1)}$ that $a_1^{(t_2-1)} = a_0^{(t_2-1)} + 4\epsilon + \ell_1\delta$ and $a_2^{(t_2-1)} = a_0^{(t_2-1)} + a_1^{(t_2-1)} - \delta$ for some $\delta > 0$. Let the vectors $b^{(t)}, c^{(t)} \in \mathbb{R}^4$ be defined as $b^{(t_2-1)} = a^{(t_2-1)} + (\epsilon, -\epsilon, 0, 0)$ and $c^{(t_2-1)} = (-\epsilon, \epsilon, 0, 0)$, that is, $a^{(t_2-1)} = b^{(t_2-1)} + c^{(t_2-1)}$. For all $t \geq t_2$, let the load moved from $a_0^{(t)}$ to $a_1^{(t)}$ be equal to the load moved from $b_0^{(t)}$ to $b_1^{(t)}$. Let MAXDELAY be applied to $b^{(t)}$ and $c^{(t)}$. Then the equations for $a^{(t_2-1)}$ above yield that

- $b^{(t_2)} = (a_0^{(t_2)} + \epsilon, a_1^{(t_2)} - (1 - \frac{2}{\ell_1})\epsilon, a_2^{(t_2)} - \frac{2}{\ell_1}\epsilon, 0)$ and
- $c^{(t_2)} = (-\epsilon, (1 - \frac{2}{\ell_1})\epsilon, \frac{2}{\ell_1}\epsilon, 0)$.

Hence, $a^{(t_2)} = b^{(t_2)} + c^{(t_2)}$. Furthermore, it is easy to check that $b_2^{(t_2)} = b_0^{(t_2)} + b_1^{(t_2)}$, using the equation for $a_2^{(t_2)}$ above. Thus Claim A.7 shows that $a^{(t)} = b^{(t)} + c^{(t)}$ also for all $t > t_2$.

Claim A.7 *Given some time step t with distributions $(a_0^{(t)}, a_1^{(t)}, a_2^{(t)}, a_3^{(t)})$, $(b_0^{(t)}, b_1^{(t)}, b_2^{(t)}, b_3^{(t)})$ and $(c_0^{(t)}, c_1^{(t)}, c_2^{(t)}, c_3^{(t)})$, where*

- *$a_i^{(t)} = b_i^{(t)} + c_i^{(t)}$ for all $i \in \{0, \dots, 3\}$,*
- *$a_1^{(t)} \geq a_0^{(t)}$, $a_2^{(t)} \geq a_1^{(t)} + a_0^{(t)}$,*
- *$b_1^{(t)} \geq b_0^{(t)}$, $b_2^{(t)} \geq b_1^{(t)} + b_0^{(t)}$,*
- *$c_1^{(t)} \geq c_0^{(t)}$, $c_2^{(t)} \geq c_1^{(t)} + c_0^{(t)}$, and*
- *a load of $\epsilon \geq 0$ is moved from $a_0^{(t)}$ to $a_1^{(t)}$ and from $b_0^{(t)}$ to $b_1^{(t)}$ in step $t+1$.*

Then it holds that, using MAXDELAY for $a^{(t)}$, $b^{(t)}$ and $c^{(t)}$,

- *$a_i^{(t+1)} = b_i^{(t+1)} + c_i^{(t+1)}$ for all $i \in \{0, \dots, 3\}$,*
- *$a_1^{(t+1)} \geq a_0^{(t+1)}$, $a_2^{(t+1)} \geq a_1^{(t+1)} + a_0^{(t+1)}$,*
- *$b_1^{(t+1)} \geq b_0^{(t+1)}$, $b_2^{(t+1)} \geq b_1^{(t+1)} + b_0^{(t+1)}$, and*
- *$c_1^{(t+1)} \geq c_0^{(t+1)}$, $c_2^{(t+1)} \geq c_1^{(t+1)} + c_0^{(t+1)}$*

Proof. Follows directly from the application of MAXDELAY. ∎

Similarly, let us decompose $x^{(t)}$ into $y^{(t)}$ and $z^{(t)}$ with $y^{(t_2-1)} = x^{(t_2-1)} + (\epsilon, -\epsilon, 0, 0)$ and $z^{(t_2-1)} = (-\epsilon, \epsilon, 0, 0)$. Then the following claim holds.

Claim A.8 *Under the assumption that MAXDELAY is applied to $x^{(t)}$, $y^{(t)}$ and $z^{(t)}$ it holds that $x^{(t)} \triangleright y^{(t)} + z^{(t)}$ for all $t \geq t_2 - 1$.*

Proof. Let t_3 be the first time step for which $y_2^{(t)} \geq y_1^{(t)} + y_0^{(t)}$. Clearly, in this case it already holds that $x_2^{(t)} \geq x_1^{(t)} + x_0^{(t)}$. Suppose that $x^{(t_3)} \triangleright y^{(t_3)} + z^{(t_3)}$. Define $u^{(t_3)} = y^{(t_3)} + z^{(t_3)}$. Then, according to Claim A.7, $u^{(t)} = y^{(t)} + z^{(t)}$ for all $t \geq t_3$. Since $x^{(t_3)} \triangleright u^{(t_3)}$, applying MAXDELAY ensures that $x^{(t)} \triangleright u^{(t)}$ for all $t \geq t_3$. Hence, the claim would be true for all $t \geq t_3$, given that $x^{(t_3)} \triangleright y^{(t_3)} + z^{(t_3)}$.

We prove the claim for $t_2 - 1 \leq t \leq t_3$ by complete induction over t. For $t = t_2 - 1$ the claim is obviously true. Suppose now that the claim is true for some $t_2 - 1 \leq t < t_3$. Let $u^{(t)} = (z_0^{(t)}, z_1^{(t)}, x_2^{(t)} - y_2^{(t)}, z_3^{(t)} + (z_2^{(t)} - (x_2^{(t)} - y_2^{(t)})))$. Since it follows from Claim A.7 that $x_0^{(t')} = y_0^{(t')} + z_0^{(t')}$ and $x_1^{(t')} = y_1^{(t')} + z_1^{(t')}$ for all $t' \geq t_2 - 1$, it holds that $x_2^{(t)} + x_3^{(t)} = y_2^{(t)} + z_2^{(t)} + z_3^{(t)}$. Hence,

$$x_2^{(t)} - y_2^{(t)} \geq z_2^{(t)} \quad \Leftrightarrow \quad z_2^{(t)} + z_3^{(t)} - x_3^{(t)} \geq z_2^{(t)} \quad \Leftrightarrow \quad z_3^{(t)} \geq x_3^{(t)} ,$$

which is true. Therefore, $u^{(t)} \triangleright z^{(t)}$, and also $u^{(t+1)} \triangleright z^{(t+1)}$. Furthermore, $x^{(t)} = y^{(t)} + u^{(t)}$. Since $y_2^{(t)} < y_1^{(t)} + y_0^{(t)}$, $u_3^{(t+1)} - u_3^{(t)}$ is equal to

$$\begin{aligned} \frac{1}{\ell_1}(u_2^{(t)} - (u_1^{(t)} + u_0^{(t)})) &= \frac{1}{\ell_1}(x_2^{(t)} - y_2^{(t)} - (x_1^{(t)} - y_1^{(t)} + x_0^{(t)} - y_0^{(t)})) \\ &= \frac{1}{\ell_1}(x_2^{(t)} - (x_1^{(t)} + x_0^{(t)})) - \frac{1}{\ell_1}(y_2^{(t)} - (y_1^{(t)} + y_0^{(t)})) \geq x_3^{(t+1)} - x_3^{(t)} \end{aligned}$$

Thus, $x^{(t+1)} \triangleright y^{(t+1)} + u^{(t+1)} \triangleright y^{(t+1)} + z^{(t+1)}$. This completes the proof of the claim. ∎

Furthermore, the next claim shows that $z^{(t)} \triangleright c^{(t)}$ for all $t \geq t_2 - 1$.

Claim A.9 *Consider the case that $(a_0^0, a_1^0, a_2^0, a_3^0) = (-\epsilon, \epsilon, 0, 0)$ and for the rest of the time steps nothing is moved from a_0 to a_1. Then we get with MAXDELAY that, for every $t \in \mathbb{N}$,*

$$\begin{aligned}
a_1^t &= \left(2\left(1-\frac{1}{\ell_1}\right)^t - 1\right)\epsilon\ , \\
a_2^t &= 2\left[\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{(t-1)-i} - \left(1-\left(1-\frac{1}{\ell_2}\right)^t\right)\right]\epsilon\ , \\
a_3^t &= 2\left[2-\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{(t-1)-i} - \left(1-\frac{1}{\ell_1}\right)^t - \left(1-\frac{1}{\ell_2}\right)^t\right]\epsilon\ .
\end{aligned}$$

Proof. We prove the formulas above by complete induction. Let us start with a_1^t. For $t = 0$ the formula is obviously correct. Suppose it is correct for some $t \geq 0$. Then it holds for step $t+1$:

$$\begin{aligned}
a_1^{t+1} &= a_1^t - \tfrac{1}{\ell_1}(a_1^t - a_0^t) \\
&= \left(2\left(1-\frac{1}{\ell_1}\right)^t - 1\right)\epsilon - \frac{1}{\ell_1}\left[\left(2\left(1-\frac{1}{\ell_1}\right)^t - 1\right)\epsilon + \epsilon\right] \\
&= \left(2\left(1-\frac{1}{\ell_1}\right)^t - 2\frac{1}{\ell_1}\left(1-\frac{1}{\ell_1}\right)^t - 1\right)\epsilon \\
&= \left(2\left(1-\frac{1}{\ell_1}\right)^{t+1} - 1\right)\epsilon\ .
\end{aligned}$$

Next we prove the formula for a_2^t. For $t = 0$, it is clearly correct. Suppose the formula is correct for some $t \geq 0$. Then it holds for step $t+1$:

$$\begin{aligned}
a_2^{t+1} &= a_2^t + \tfrac{1}{\ell_1}(a_1^t - a_0^t) - \tfrac{1}{\ell_2}(a_2^t - a_1^t - a_0^t) \\
&= \left(1-\tfrac{1}{\ell_2}\right)a_2^t + \left(\tfrac{1}{\ell_1}+\tfrac{1}{\ell_2}\right)a_1^t + \left(\tfrac{1}{\ell_1}-\tfrac{1}{\ell_2}\right)\epsilon \\
&= \left(1-\frac{1}{\ell_2}\right)a_2^t + \left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\left(2\left(1-\frac{1}{\ell_1}\right)^t - 1\right)\epsilon + \left(\frac{1}{\ell_1}-\frac{1}{\ell_2}\right)\epsilon \\
&= a_2^t - \frac{2}{\ell_2}\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\epsilon\sum_{i=0}^{t-1}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{(t-1)-i} - \frac{2}{\ell_2}\left(1-\frac{1}{\ell_2}\right)^t\epsilon + \\
&\quad 2\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\left(1-\frac{1}{\ell_1}\right)^t\epsilon \\
&= 2\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\left[\left(1-\frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{(t-1)-i} + \left(1-\frac{1}{\ell_1}\right)^t\right]\epsilon - \\
&\quad 2\left(1-\left(1-\frac{1}{\ell_2}\right)\left(1-\frac{1}{\ell_2}\right)^t\right)\epsilon \\
&= 2\left[\left(\frac{1}{\ell_1}+\frac{1}{\ell_2}\right)\sum_{i=0}^{t}\left(1-\frac{1}{\ell_1}\right)^i\left(1-\frac{1}{\ell_2}\right)^{t-i} - \left(1-\left(1-\frac{1}{\ell_2}\right)^{t+1}\right)\right]\epsilon\ .
\end{aligned}$$

It remains to prove the formula for a_3^t. Since $\sum_{i=0}^{3} a_i^t = 0$ for all $t \geq 0$ we get:

$$a_3^t \;=\; \epsilon - a_2^t - a_1^t$$

$$
\begin{aligned}
&= \left[1 - 2\left[\left(\frac{1}{\ell_1} + \frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1 - \frac{1}{\ell_1}\right)^i \left(1 - \frac{1}{\ell_2}\right)^{(t-1)-i} - \left(1 - \left(1 - \frac{1}{\ell_2}\right)^t\right)\right] - \left(2\left(1 - \frac{1}{\ell_1}\right)^t - 1\right)\right]\epsilon \\
&= 2\left[2 - \left(\frac{1}{\ell_1} + \frac{1}{\ell_2}\right)\sum_{i=0}^{t-1}\left(1 - \frac{1}{\ell_1}\right)^i \left(1 - \frac{1}{\ell_2}\right)^{(t-1)-i} - \left(1 - \frac{1}{\ell_2}\right)^t - \left(1 - \frac{1}{\ell_1}\right)^t\right]\epsilon \; .
\end{aligned}
$$

This completes the proof of the claim. ■

Since, for all $t \geq t_2$, $a^{(t)} = b^{(t)} + c^{(t)}$ according to Claim A.7, $x^{(t)} \triangleright y^{(t)} + z^{(t)}$ according to Claim A.8 and $z^{(t)} \triangleright c^{(t)}$ according to Claim A.9, it holds that:

In order to show that $x^{(t)} \triangleright a^{(t)}$ also for $t > t_2$ it remains to show that $y^{(t)} \triangleright b^{(t)}$ for all $t > t_2$. Because $y_3^{(t_2+1)} = b_3^{(t_2+1)} = 0$, this is certainly true for $t = t_2 + 1$. Applying the same reduction to $b^{(t)}$ and $y^{(t)}$ for time step $t_2 + 2$ as done to $a^{(t)}$ and $x^{(t)}$ for time step $t_2 + 1$, proves that also $x^{(t_2+2)} \triangleright a^{(t_2+2)}$. Continuing this for time steps $t_2 + 3, t_2 + 4, \ldots$ yields the proposition. ■

¿From the proposition it immediately follows that dominance is preserved. This proves the lemma. ■

Lemma A.5 implies that we can order the edges in p in such a way that first all edges with latency L are used, then all edges with latency $L/2$ are used, and so on, until we arrive at the edges with latency 2. Let the resulting schedule be called $\mathcal{S}_4$. Let $\mathcal{S}_{4,i}$ denote the part of $\mathcal{S}$ that contains all edges of latency 2^i. According to the uniform case, for each one of the $\mathcal{S}_{4,i}$, it takes at most $O(T + 2^i \log n)$ steps, w.h.p., to complete the simulation of $\mathcal{S}_{4,i}$. Hence, the total amount of time steps required, w.h.p., to simulate $\mathcal{S}_4$ is bounded by

$$
\sum_{i=\log \ell}^{\log L} O\left(T + 2^i \log n\right) = O(T \log(L/\ell) + L \log n) \; .
$$

Hence, after $O(T\log(L/\ell) + L\log N)$ time steps the simulation of $\mathcal{S}_1$ in G is completed, w.h.p., which concludes the proof of Theorem 2.12.