

Direct Routing: Algorithms and Complexity

Costas Busch¹, Malik Magdon-Ismail¹, Marios Mavronicolas², and Paul Spirakis³

¹ Department of Computer Science, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, USA. Email: {buschc,magdon}@cs.rpi.edu

² Department of Computer Science, University of Cyprus, P. O. Box 20537, Nicosia CY-1678, Cyprus. Email: mavronic@ucy.ac.cy

³ Department of Computer Engineering and Informatics, University of Patras, Rion, 265 00 Patras, Greece. Email: spirakis@cti.gr

Abstract. *Direct routing* is the special case of *bufferless routing* where N packets, once injected into the network, must be routed along specific paths to their destinations without conflicts. We give a general treatment of three facets of direct routing:

- (i) *Algorithms.* We present a polynomial time *greedy* algorithm for arbitrary direct routing problems which is worst-case optimal, i.e., there exist instances for which no direct routing algorithm is better than the greedy. We apply variants of this algorithm to commonly used network topologies. In particular, we obtain *near-optimal* routing time using for the *tree* and *d-dimensional mesh*, given arbitrary sources and destinations; for the *butterfly* and the *hypercube*, the same result holds for random destinations.
- (ii) *Complexity.* By a reduction from Vertex Coloring, we show that Direct Routing is inapproximable, unless $P=NP$.
- (iii) *Lower Bounds for Buffering.* We show the existence of routing problems which cannot be solved efficiently with direct routing. To solve these problems, *any* routing algorithm needs buffers. We give non-trivial lower bounds on such buffering requirements for general routing algorithms.

1 Introduction

Direct routing is the special case of *bufferless routing* where N packets are routed along specific paths from *source* to *destination* so that they do not conflict with each other, i.e., once injected, the packets proceed “directly” to their destination without being delayed (buffered) at intermediate nodes. Since direct routing is bufferless, packets spend the minimum possible time in the network, given the paths they must follow – this is appealing in power/resource constrained environments (for example optical networks or sensor networks). From the point of view of quality of service, it is often desirable to provide a guarantee on the delivery time after injection, for example in streaming applications like audio and video; direct routing can provide such guarantees, since packets are not delayed after injection.

The task of a direct routing algorithm is to compute the injection times of the packets so as to minimize the *routing time*, which is the time at which the last packet is delivered to its destination. Algorithms for direct routing are inherently offline in order to guarantee no conflicts. We give a general analysis of three aspects of direct routing, namely efficient algorithms for direct routing; the computational complexity of direct routing; and, the connection between direct routing and buffering.

Algorithms for direct routing. We present efficient polynomial time algorithms for general as well as specific routing problems on commonly used network topologies. Thus, we show that in many cases, efficient routing can be achieved without the use of buffers.

Arbitrary: We give a simple *greedy* algorithm which considers packets in some order and assigns the first available injection time to each packet. This algorithm is worst-case optimal, in the sense that there exist instances of direct routing problems for which no direct routing algorithm achieves better routing time than our greedy algorithm.

Tree: For arbitrary sources and destinations on arbitrary trees, we give a direct routing algorithm with routing time $rt = O(rt^*)$ where rt^* is the minimum possible routing time achievable by *any* routing algorithm (direct or not).

d-Dimensional Mesh: For arbitrary sources and destinations on a d -dimensional mesh with n nodes, we give a direct routing algorithm with routing time $rt = O(d^2 \cdot \log^2 n \cdot rt^*)$ with high probability (w.h.p.).

Butterfly and Hypercube: For permutation and random destination problems with one packet per node, we obtain routing time $rt = O(rt^*)$ w.h.p. for the butterfly with n inputs and the n -node hypercube.

Computational complexity of direct routing. By a reduction from vertex coloring, we show that direct routing is NP-complete. The reduction is *gap-preserving*, so direct routing is as hard to approximate as coloring.

Lower bounds for buffering. There exist routing problems whose efficient solution cannot be accomplished with direct routing. For such problems, efficient solutions require buffering. We show that for *any buffered algorithm* there exist routing instances, for which packets are buffered $\Omega(N^{4/3})$ times in order to achieve near-optimal routing time.

Next, we discuss related work, followed by preliminaries and main results. Most of the proofs can be found in the appendix.

Related Work

The only previous known work on direct routing is for trees. In particular, Symvonis [1] and Alstrup et al. [2] study permutations on trees and give routing algorithms with routing time $O(n)$ for any tree with n nodes, which is worst-case optimal. Our algorithm for trees is every-case optimal (up to a factor of 2) for arbitrary routing problems.

A dual to direct routing is time constrained routing where the task is to schedule as many packets as possible within a given time frame [3, 4]. In these papers, the authors show that the time constrained version of the problem is NP-complete, and study approximation algorithms on linear networks, trees and meshes. They also discuss how much buffering could help in this setting.

Other models of bufferless routing, in which packets do not follow specific paths, are *matching routing* [5] and *hot-potato routing* [6–8]. In [6], the authors also study lower bounds for near-greedy hot-potato algorithms on the mesh. Optimal routing for given paths on arbitrary networks has been studied extensively in the context of store-and-forward algorithms, [9–11].

2 Preliminaries

Problem Definition. We are given a graph $G = (V, E)$ with $n \geq 1$ nodes, and a set of packets $\Pi = \{\pi_i\}_{i=1}^N$. Each packet π_i is to be routed from its source, $s(\pi_i) \in V$, to its destination, $\delta(\pi_i) \in V$, along a pre-specified path p_i . The nodes in the graph are synchronous: time is discrete and all nodes take steps simultaneously. At each time step, at most one packet can follow a link in each direction (thus, at most two packets can follow a link at the same time, one packet at each direction).

A path p is a sequence of vertices $p = (v_1, v_2, \dots, v_k)$. Two paths p_1 and p_2 *collide* if they share an edge in the same direction. We also say that their respective packets π_1 and π_2 collide. Two packets *conflict* if they are routed in such a way that they appear in the same node at the same time, *and* the next edge in their paths is the same. The length of a path p , denoted $|p|$, is the number of edges in the path. For any edge $e = (v_i, v_j) \in p$, let $d_p(e)$ denote the length of path (v_1, \dots, v_i, v_j) . The *distance* between two nodes, is the length of a shortest path that connects the two nodes.

A *direct routing problem* has the following components. **Input:** (G, Π, \mathcal{P}) , where G is a graph, and the packets $\Pi = \{\pi_i\}_{i=1}^N$ have respective paths $\mathcal{P} = \{p_i\}_{i=1}^N$. **Output:** The injection times $\mathcal{T} = \{\tau_i\}_{i=1}^N$, denoted a *routing schedule* for the routing problem. **Validity:** If each packet π_i is injected at its corresponding time τ_i into its source s_i , then it will follow a conflict-free path to its destination where it will be absorbed at time $t_i = \tau_i + |p_i|$.

For a routing problem (G, Π, \mathcal{P}) , the *routing time* $rt(G, \Pi, \mathcal{P})$ is the maximum time at which a packet gets absorbed at its destination, $rt(G, \Pi, \mathcal{P}) = \max_i\{\tau_i + |p_i|\}$. The *offline time*, $ol(G, \Pi, \mathcal{P})$ is the number of operations used to compute the routing schedule \mathcal{T} . We measure the efficiency of a direct routing algorithm with respect to the *congestion* C (the maximum number of packets that use an edge) and the *dilation* D (the maximum length of any path). A well known lower bound on the routing time of any routing algorithm (direct or not) is given by $\Omega(C + D)$.

Dependency Graphs. Consider a routing problem (G, Π, \mathcal{P}) . The *dependency graph* \mathcal{D} of the routing problem is a graph in which each packet $\pi_i \in \Pi$ is a

- 1: // **Greedy direct routing algorithm:**
- 2: // **Input:** routing problem (G, II, \mathcal{P}) with N packets $II = \{\pi_i\}_{i=1}^N$.
- 3: // **Output:** Set of injection times $\mathcal{T} = \{\tau_i\}_{i=1}^N$.
- 4: Let π_1, \dots, π_N be any specific, arbitrarily chosen ordering of the packets.
- 5: **for** $i = 1$ **to** N **do**
- 6: Greedily assign the first available injection time τ_i to packet $\pi_i \in II$, so that it does not conflict with any packet already assigned an injection time.
- 7: **end for**

The greedy direct routing algorithm is really a family of algorithms, one for each specific ordering of the packets. It is easy to show by induction, that no packet π_j conflicts with any packet π_i with $i < j$, and thus the greedy algorithm produces a valid routing schedule. The routing time for the greedy algorithm will be denoted $rt_{Gr}(G, II, \mathcal{P})$. Consider the dependency graph \mathcal{D} for the routing problem (G, II, \mathcal{P}) . We can show that $\tau_i \leq W(\pi_i)$, where $W(\pi_i)$ is the weight degree of packet π_i , which implies:

Lemma 2. $rt_{Gr}(G, II, \mathcal{P}) \leq \max_i \{W(\pi_i) + |p_i|\}$.

We now give an upper bound on the routing time of the greedy algorithm. Since the congestion is C and $|p_i| \leq D \forall i$, a packet collides with other packets at most $(C - 1) \cdot D$ times. Thus, $W(\pi_i) \leq (C - 1) \cdot D, \forall i$. Therefore, using Lemma 2 we obtain:

Theorem 1 (Greedy Routing Time Bound). $rt_{Gr}(G, II, \mathcal{P}) \leq C \cdot D$.

The general $O(C \cdot D)$ bound on the routing time of the greedy algorithm is worst-case optimal, within constant factors, since from Theorem 9, there exist worst-case routing problems with $\Omega(C \cdot D)$ routing time. In the next sections, we will show how the greedy algorithm can do better for particular routing problems by using a more careful choice of the order in which packets are considered.

Now we discuss the offline time of the greedy algorithm. Each time an edge on a packet's path is used by some other packet, the greedy algorithm will need to desynchronize these packets if necessary. This will occur at most $C \cdot D$ times for a packet, hence, the offline computation time of the greedy algorithm is $ol_{Gr}(G, II, \mathcal{P}) = O(N \cdot C \cdot D)$, which is polynomial. This bound is tight since, in the worst case, each packet may have $C \cdot D$ collisions with other packets.

3.1 Trees

Consider the routing problem (T, II, \mathcal{P}) , in which T is a tree with n nodes, and all the paths in \mathcal{P} are shortest paths. Shortest paths are optimal on trees given sources and destinations because any paths must contain the shortest path. Thus, $rt^* = \Omega(C + D)$, where rt^* is the minimum routing time for the given sources and destinations using *any* routing algorithm. The greedy algorithm with a particular order in which the packets are considered gives an asymptotically optimal schedule.

Let r be an arbitrary node of T . For a packet π_i , let d_i denote its *path depth*, which is the minimum depth of any node in the path p_i , with respect to the tree rooted at r . The direct routing algorithm can now be simply stated as the greedy algorithm with the packets considered in sorted order, according to the path depth d_i , with $d_{i_1} \leq d_{i_2} \leq \dots \leq d_{i_N}$.

Theorem 2. *Let (T, Π, \mathcal{P}) be any routing problem on the tree T . Then the routing time of the greedy algorithm using the path depth ordered packets is $rt(T, \Pi, \mathcal{P}) \leq 2C + D - 2$.*

Proof. We show that every injection time satisfies $\tau_i \leq 2C - 2$. When packet π_i with path depth d_i is considered, let v_i be the closest node to r on its path. All packets that are already assigned times that could possibly conflict with π_i are those that use the two edges in π_i 's path incident with v_i , hence there are at most $2C - 2$ such packets. Since π_i is assigned the smallest available injection time, it must therefore be assigned a time in $[0, 2C - 2]$.

3.2 d -Dimensional Mesh

A d -dimensional mesh network $M = M(m_1, m_2, \dots, m_d)$ is a multi-dimensional grid of nodes with side length m_i in dimension i . The number of nodes is $n = \prod_{i=1}^d m_i$, and define $m = \sum_{i=1}^d m_i$. Every node is connected to up to $2d$ of its neighbors on the grid. Theorem 1 implies that the greedy routing algorithm achieves asymptotically optimal worst case routing time in the mesh. We discuss some important special cases where the situation is considerably better. In particular, we give a variation of the greedy direct routing algorithm which is analyzed in terms of the number of times that the packet paths “bend” on the mesh. We then apply this algorithm to the 2-dimensional mesh in order to obtain optimal permutation routing, and the d -dimensional mesh, in order to obtain near-optimal routing, given arbitrary sources and destinations.

Multi-bend Paths. Here, we give a variation of the greedy direct routing algorithm which we analyze in terms of the number of times a packet bends in the network. Consider a routing problem (G, Π, \mathcal{P}) . We first give an upper bound on the weight degree $W(\mathcal{D})$ of dependency graph \mathcal{D} in terms of bends of the paths. We then use the weight degree bound in order to obtain an upper bound on the routing time of the algorithm.

For any subset of packets $\Pi' \subseteq \Pi$, let $\mathcal{D}_{\Pi'}$ denote the subgraph of \mathcal{D} induced by the set of packets Π' . (Note that $\mathcal{D} = \mathcal{D}_{\Pi}$.) Consider the path p of a packet π . Let's assume that $p = (\dots, v_i, v, v_j, \dots)$, such that the edges (v_i, v) and (v, v_j) are in different dimensions. We say that the path of packet π *bends* at node v , and that v is an *internal bending node*. We define the source and destination nodes of a packet π to be *external bending nodes*. The *segment* $p' = (v_i, \dots, v_j)$ of a path p , is a subpath of p in which only v_i and v_j are bending nodes. Consider two packets π_1 and π_2 whose respective paths p_1 and p_2 collide at some edge e . Let p'_1 and p'_2 be the two respective segments of p_1 and p_2 which contain e . Let

p' be the longest subpath of p'_1 and p'_2 which is common to p'_1 and p'_2 ; clearly e is an edge in p' . Let's assume that $p' = (v_i, \dots, v_j)$. It must be that v_i is a bending node of one of the two packets, and the same is true of v_j . Further, none of the other nodes in p' are bending nodes of either of the two packets. We refer to such a path p' as a *common subpath*. Note there could be many common subpaths for the packets π_1 and π_2 , if they meet multiple times on their paths.

Since p_1 and p_2 collide on e , the edge $h = (\pi_1, \pi_2)$ will be present in the dependency graph \mathcal{D} with some weight $w \in \mathcal{W}_{1,2}$ representing this collision. Weight w suffices to represent the collision of the two packets on the entire subpath p' . Therefore, a common subpath contributes at most one to the weight-number of \mathcal{D} . Let $A_{\mathcal{P}}$ denote the number of common subpaths. We have that $W(\mathcal{D}) \leq A_{\mathcal{P}}$. Therefore, in order to find an upper bound on $W(\mathcal{D})$, we only need to find an upper bound on the number of common subpaths.

For each common subpath, one of the packets must bend at the beginning and one at end nodes of the subpath. Thus, a packet contributes to the number of total subpaths only when it bends. Consider a packet π which bends at a node v . Let e_1 and e_2 be the two edges of the path of π adjacent to v . On e_1 the packet may meet with at most $C - 1$ other packets. Thus, e_1 contributes at most $C - 1$ to the number of common subpaths. Similarly, e_2 contributes at most $C - 1$ to the number of common subpaths. Thus, each internal bend contributes at most $2C - 2$ to the number of common subpaths, and each external bend $C - 1$. Therefore, for the set of packets Π' , where the maximum number of internal bends is b , $A_{\mathcal{P}} \leq 2(b + 1)|\Pi'| (C - 1)$, giving the following lemma:

Lemma 3. *For any subset $\Pi' \subseteq \Pi$, $W(\mathcal{D}_{\Pi'}) \leq 2(b + 1)|\Pi'| (C - 1)$, where b is the maximum number of internal bending nodes of any path in Π' .*

Since the sum of the node weight degrees is $2W(\mathcal{D})$, we have that the average node weight degree of the dependency graph for *any* subset of the packets is upper bounded by $4(b + 1)(C - 1)$. We say that a graph \mathcal{D} is *K-amortized* if the average weight degree for every subgraph is at most K . *K-amortized* graphs are similar to balanced graphs [12]. Thus \mathcal{D} is $4(b + 1)C$ -amortized. A *generalized coloring* of a graph with weights on each edge is a coloring in which the difference between the colors of adjacent nodes cannot equal a weight. *K-amortized* graphs admit generalized colorings with $K + 1$ colors. This is the content of the next lemma.

Lemma 4 (Efficient Coloring of Amortized Graphs). *Let \mathcal{D} be a K -amortized graph. Then \mathcal{D} has a valid $K + 1$ generalized coloring.*

A generalized coloring of the dependency graph gives a valid injection schedule with maximum injection time one less than the largest color, since with such an injection schedule no pair of packets is synchronized. Lemma 4 implies that the dependency graph \mathcal{D} has a valid $4(b + 1)(C - 1) + 1$ generalized coloring. Lemma 4 essentially determines the order in which the greedy algorithm considers the packets so as to ensure the desired routing time. Hence, we get the following result:

Theorem 3 (Multi-bend Direct Routing Time). *Let (M, II, \mathcal{P}) be a direct routing problem on a mesh M with congestion C and dilation D . Suppose that each packet has at most b internal bends. Then there is a direct routing schedule with routing time $rt \leq 4(b+1)(C-1) + D$.*

Permutation Routing on the 2-Dimensional Mesh. Consider a $\sqrt{n} \times \sqrt{n}$ mesh. In a permutation routing problem every node is the source and destination of exactly one packet. We solve permutation routing problems by using paths with one internal bend. Let e be a column edge in the up direction. Since at most \sqrt{n} packet originate and have destination at each row, the congestion at each edge in the row is at most $O(\sqrt{n})$. Similarly for edges in rows. Applying Theorem 3, and the fact that $D = O(\sqrt{n})$, we then get that $rt = O(\sqrt{n})$, which is worst case optimal for permutation routing on the mesh.

Near Optimal Direct Routing on the Mesh. Maggs *et al.* [13, Section 3] give a strategy to select paths in the mesh M for a routing problem with arbitrary sources and destinations. The congestion achieved by the paths is within a logarithmic factor from the optimal, i.e., $C = O(dC^* \log n)$ w.h.p., where C^* is the minimum congestion possible for the given sources and destinations. Following the construction in [13], it can be shown that the packet paths are constructed from $O(\log n)$ shortest paths between random nodes in the mesh. Hence, the number of bends b that a packet makes is $b = O(d \log n)$, and $D = O(m \log n)$, where m is the sum of the side lengths. We can thus use Theorem 3 to obtain a direct routing schedule with the following properties:

Theorem 4. *For any routing problem (M, II) with given sources and destinations, there exists a direct routing schedule with routing time $rt = O(d^2 C^* \log^2 n + m \log n)$, w.h.p..*

Let D^* denote the maximum length of the shortest paths between sources and destinations for the packets in II . D^* is the minimum possible dilation. Let rt^* denote the optimal routing time (direct or not). For any set of paths, $C + D = \Omega(C^* + D^*)$, and so the optimal routing time rt^* is also $\Omega(C^* + D^*)$. If $D^* = \Omega(m/(d^2 \log n))$, then $rt^* = \Omega(C^* + m/(d^2 \log n))$, so Theorem 4 implies:

Corollary 1. *If $D^* = \Omega(m/(d^2 \log n))$, then there exists a direct routing schedule with $rt = O(rt^* d^2 \log^2 n)$, w.h.p..*

3.3 Butterfly and Hypercube

First, consider the n -input butterfly network B , where $n = 2^k$, [14]. There is a unique path from an input node to an output node of length $\lg n + 1$. Assume that every input node is the source of one packet and the destinations are randomly chosen.

For packet π_i , we consider the Bernoulli random variables x_j which equal one if packet π_j collides with π_i . Then the degree of π_i in the dependency graph is $X_i = \sum_j x_j$. We show that $E[X_i] = \frac{1}{4}(\lg n - 1)$, and since the x_j are independent,

we use the Chernoff bound to get a concentration result on X_i . Thus, we show that w.h.p, $\max_i X_i = O(\lg n)$. Since the injection time assigned by the greedy algorithm to any packet is at most its degree in the dependency graph, we find that the routing time of the greedy algorithm for a random destination problem on the butterfly satisfies $\mathbf{P} [rt_{Gr}(B, II, \mathcal{P}) \leq \frac{5}{2} \lg n] > 1 - 2\sqrt{2}n^{-\frac{1}{2}}$ (the details are given in an appendix).

Valiant [15, 16] proposed permutation routing on butterfly-like networks by connecting two butterflies, with the outputs of one as the inputs to the other. The permutation is routed by first sending the packets to random destinations on the outputs of the first butterfly. This approach avoids hot-spots and converts the permutation problem to two random destinations problems. Thus, we can apply the result for random destinations twice to obtain the following theorem:

Theorem 5. *For permutation routing on the double-butterfly with random intermediate destinations, the routing time of the greedy algorithm satisfies $\mathbf{P} [rt_{Gr} \leq 5 \lg n] > 1 - 4\sqrt{2}n^{-\frac{1}{2}}$.*

A similar analysis holds for the hypercube network (see appendix).

Theorem 6. *For permutation routing using random intermediate destinations and bit-fixing paths on a hypercube with n nodes, the routing time of the greedy algorithm satisfies $\mathbf{P} [rt_{Gr} < 14 \lg n] > 1 - 1/(16n)$.*

4 Computational Complexity of Direct Routing

In this section, we show that direct routing and approximate versions of it are NP-complete. First, we introduce the formal definition of the direct routing decision problem. In our reductions, we will use the well known NP-complete problem VERTEX COLOR, the vertex coloring problem [17], which asks whether a given graph G is κ -colorable. The chromatic number, $\chi(G)$ is the smallest κ for which G is κ -colorable. An algorithm *approximates* $\chi(G)$ with *approximation ratio* $q(G)$ if on any input G , the algorithm outputs $u(G)$ such that $\chi(G) \leq u(G)$ and $u(G)/\chi(G) \leq q(G)$. Typically, $q(G)$ is expressed only as a function of the number of vertices in G . It is known [18] that unless $\mathbf{P}=\mathbf{NP}^\dagger$, there does not exist a polynomial time algorithm to approximate $\chi(G)$ with approximation ratio $N^{1/2-\epsilon}$ for any constant $\epsilon > 0$.

By polynomially reducing coloring to direct routing, we will obtain hardness and inapproximability results for direct routing. We now formally define a generalization of the direct routing decision problem which allows for collisions. We say that an injection schedule is a valid K -collision schedule if at most K collisions occur during the course of the routing (a collision is counted for every collision of every pair of packets on every edge).

Problem: APPROXIMATE DIRECT ROUTE

[†] It is also known that if $\mathbf{NP} \not\subseteq \mathbf{ZPP}$ then χ is inapproximable to within $N^{1-\epsilon}$, however we cannot use this result as it requires both upper and lower bounds.

Input: A direct routing problem (G, Π, \mathcal{P}) and integers $T, K \geq 0$,

Question: Does there exist a valid k -collision direct routing schedule \mathcal{T} for some $k \leq K$ and with maximum injection time $\tau_{max} \leq T$?

The problem DIRECT ROUTE is the restriction of APPROXIMATE DIRECT ROUTE to instances where $K = 0$. Denoting the maximum injection time of a valid K -collision injection schedule by T , we define the K -collision injection number $\tau_K(G, \Pi, \mathcal{P})$ for a direct routing problem as the minimum value of T for which a valid K -collision schedule exists. We say that a schedule approximates $\tau_K(G, \Pi, \mathcal{P})$ with ratio q if it is a schedule with at most K collisions and the maximum injection time for this schedule approximates $\tau_K(G, \Pi, \mathcal{P})$ with approximation ratio q . We now show that direct routing is NP-hard.

Theorem 7 (DIRECT ROUTE is NP-Hard). *There exists a polynomial time reduction from any instance (G, κ) of VERTEX COLOR to an instance $(G', \Pi, \mathcal{P}, T = \kappa - 1)$ of DIRECT ROUTE.*

Sketch of Proof. We will use the direct routing problem illustrated to the right, for which the dependency graph is a synchronized clique. Each path is associated to a level, which denotes the x -coordinate at which the path moves vertically up after making its final left turn. There is a path for every level in $[0, L]$, and the total number of packets is $N = L + 1$. The level- i path for $i > 0$ begins at $(1 - i, i - 1)$ and ends at $(i, L + i)$, and is constructed as follows. Beginning at $(1 - i, i - 1)$, the path moves right till $(0, i - 1)$, then alternating between up and right moves till it reaches level i at node $(i, 2i - 1)$ (i alternating up and right moves), at which point the path moves up to $(i, L + i)$. Every packet is synchronized with every other packet, and meets every other packet exactly once.

Given an instance $I = (G, K)$ of VERTEX COLOR, we reduce it in polynomial time to an instance $I' = (G', \Pi, \mathcal{P}, T = K - 1)$ of DIRECT ROUTE. Each node in G corresponds to a packet in Π . The paths are initially as illustrated in the routing problem above with $L = N - 1$. The transformed problem will have dependency graph \mathcal{D} that is isomorphic to G , thus a coloring of G will imply a schedule for \mathcal{D} and vice versa.

If (u, v) is not an edge in G , then we remove that edge in the dependency graph by altering the path of u and v without affecting their relationship to any other paths; we do so via altering the edges of G' by making one of them pass above the other, thus avoiding the conflict. After this construction, the resulting dependency graph is isomorphic to G . ■

DIRECT ROUTE is in NP, since, given a direct routing schedule, by traversing every pair of packets and checking for collision we can determine if it is valid

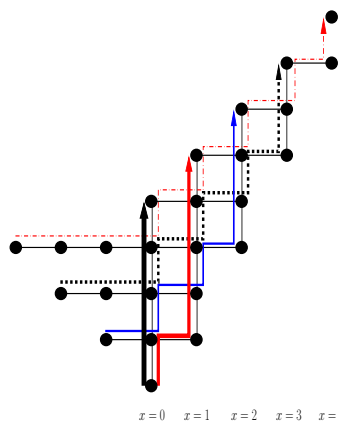


Fig. 2. A mesh routing problem.

and if the minimum injection time is $\leq T$, and so DIRECT ROUTE is NP complete. Further, we see that the reduction is gap preserving with gap preserving parameter $\rho = 1$ [19].

Theorem 8 (Inapproximability of Collision Injection Number). *A polynomial time algorithm that approximates $\tau_K(G, \Pi, \mathcal{P})$ with ratio r for an arbitrary direct routing problem yields a polynomial time algorithm that approximates the chromatic number of an arbitrary graph with ratio $r + K + 1$. In particular, choosing $K = O(r)$ preserves the approximation ratio.*

Letting $K = O(N^{1/2-\epsilon})$, since χ is hard to approximate with ratio $O(N^{1/2-\epsilon})$, we have

Corollary 2 (Inapproximability of Scheduling). *Unless $P=NP$, for $K = O(N^{1/2-\epsilon})$, there is no polynomial time algorithm to determine a valid K -collision direct routing schedule that approximates $\tau_K(G, \Pi, \mathcal{P})$ with ratio $O(N^{1/2-\epsilon})$ for any $\epsilon > 0$.*

5 Lower Bounds for Buffering

Here we consider the buffering requirements of any routing algorithm. We construct a “hard” routing problem for which *any* direct routing algorithm has routing time $rt = \Omega(C \cdot D) = \Omega((C + D)^2)$, which is asymptotically worse than optimal. We then analyze the amount of buffering that would be required to attain near optimal routing time, which results in a lower bound on the amount of buffering needed by *any* store-and-forward algorithm.

Theorem 9 (Hard Routing Problem). *For every direct routing algorithm, there exist routing problems for which the routing time is $\Omega(C \cdot D) = \Omega((C + D)^2)$.*

Proof. We construct a routing problem with $N = \Theta(C \cdot D)$ and for which the dependency graph \mathcal{D} is a synchronized clique. The paths are as in Figure 2, and the description of the routing problem is in the proof of Theorem 7. The only difference is that c packets use each path. The congestion is $C = 2c$ and the dilation is $D = 3L$. Since every pair of packets is synchronized, Lemma 1 implies that $rt(G, \Pi, \mathcal{P}) \geq N$. Since $N = c(L + 1) = \frac{C}{2}(\frac{D}{3} + 1)$, $rt(G, \Pi, \mathcal{P}) = \Omega(C \cdot D)$. Choosing $c = \Theta(\sqrt{N})$ and $L = \Theta(\sqrt{N})$, we have that $C + D = \Theta(\sqrt{N})$ so $C + D = \Theta(\sqrt{C \cdot D})$.

Let problem A denote the routing problem in the proof of Theorem 9. Suppose that we have the option to buffer some of the packets. We would like to determine how much buffering is necessary in order to decrease the routing time for routing problem A . Let T be the maximum injection time (so the routing time is bounded by $T + D$). We give a lower bound on the number of packets that need to be buffered at least once:

Lemma 5. *In routing problem A , if $T \leq \alpha$, then at least $N - \alpha$ packets are buffered at least once.*

Proof. If β packets are not buffered at all, then they form a synchronized clique, hence $T \geq \beta$. Therefore $\alpha \geq \beta$, and since $N - \beta$ packets are buffered at least once, the proof is complete.

If the routing time is $O(C + D)$, then $\alpha = O(C + D)$. Choosing c and L to be $\Theta(N^{1/2})$, we have that $\alpha = O(N^{1/2})$, and so from Lemma 5, the number of packets buffered is $\Omega(N)$:

Corollary 3. *There exists a routing problem for which any algorithm will buffer $\Omega(N)$ packets at least once to achieve asymptotically optimal routing time.*

By repeating problem A appropriately, we can strengthen this corollary to obtain the following theorem (details are given in the appendix).

Theorem 10 (Buffering-Routing Time Tradeoff). *There exists a routing problem which requires $\Omega(N^{(4-2\epsilon)/3})$ buffering to obtain a routing time that is a factor $O(N^\epsilon)$ from optimal.*

References

1. Symvonis, A.: Routing on trees. *Information Processing Letters* **57** (1996) 215–223
2. Alstrup, S., Holm, J., de Lichtenberg, K., Thorup, M.: Direct routing on trees. In: *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 98)*. (1998) 342–349
3. Adler, M., Khanna, S., Rajaraman, R., Rosen, A.: Time-constrained scheduling of weighted packets on trees and meshes. In: *Proceedings of 11th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. (1999)
4. Adler, M., Rosenberg, A.L., Sitaraman, R.K., Unger, W.: Scheduling time-constrained communication in linear networks. In: *Proceedings of 10th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. (1998)
5. Alon, N., Chung, F., R.L.Graham: Routing permutations on graphs via matching. *SIAM Journal on Discrete Mathematics* **7** (1994) 513–530
6. Ben-Aroya, I., Chinn, D.D., Schuster, A.: A lower bound for nearly minimal adaptive and hot potato algorithms. *Algorithmica* **21** (1998) 347–376
7. Busch, C., Herlihy, M., Wattenhofer, R.: Hard-potato routing. In: *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*. (2000) 278–285
8. Meyer auf der Heide, F., Scheideler, C.: Routing with bounded buffers and hot-potato routing in vertex-symmetric networks. In Spirakis, P.G., ed.: *Proceedings of the Third Annual European Symposium on Algorithms*. Volume 979 of LNCS., Corfu, Greece (1995) 341–354
9. Leighton, T., Maggs, B., Richa, A.W.: Fast algorithms for finding $O(\text{congestion} + \text{dilation})$ packet routing schedules. *Combinatorica* **19** (1999) 375–401
10. Meyer auf der Heide, F., Vöcking, B.: Shortest-path routing in arbitrary networks. *Journal of Algorithms* **31** (1999) 105–131
11. Ostrovsky, R., Rabani, Y.: Universal $O(\text{congestion} + \text{dilation} + \log^{1+\epsilon} N)$ local control packet switching algorithms. In: *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, New York (1997) 644–653
12. Bollobás, B.: *Random Graphs*, Second Edition. New York edn. Cambridge University Press (2001)
13. Maggs, B.M., Meyer auf der Heide, F., Vocking, B., Westermann, M.: Exploiting locality for data management in systems of limited bandwidth. In: *IEEE Symposium on Foundations of Computer Science*. (1997) 284–293
14. Leighton, F.T.: *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*. Morgan Kaufmann, San Mateo (1992)
15. Valiant, L.G.: A scheme for fast parallel communication. *SIAM Journal on Computing* **11** (1982) 350–361
16. Valiant, L.G., Brebner, G.J.: Universal schemes for parallel communication. In: *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*. (1981) 263–277
17. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York (1979)
18. Feige, U., Kilian, J.: Zero knowledge and the chromatic number. In: *IEEE Conference on Computational Complexity*. (1996) 278–287
19. Hochbaum, D.S.: *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, New York (1997)
20. Motwani, R., Raghavan, P.: *Randomized Algorithms*. Cambridge University Press, Cambridge, UK (2000)

A Algorithms for Direct Routing

A.1 Greedy Algorithm

Proof of Lemma 2: We show that the injection times assigned by the greedy algorithm satisfy $\tau_i \leq W(\pi_i)$, from which the claim follows immediately. For packet i , we consider the path p_i and the interval of times $[0, W(\pi_i)]$. Every time a packet σ , that has already been assigned an injection time, uses an edge on p_i , we remove the (at most one) injection time in this set that would cause π_i to conflict with σ at the time σ uses this edge. Since $W(\pi_i)$ is the number of times packets can conflict with π_i , we remove at most $W(\pi_i)$ injection times from this set. As there are $W(\pi_i) + 1$ injection times in this set, it cannot be empty, so the greedy algorithm must assign an injection time to π_i that is in this set, as it assigns the smallest available injection time. ■

A.2 d -Dimensional Mesh

Proof of Lemma 4. We use induction on n , the order of G . For $n = 1$, the claim is trivial. Assume it is true for $n < r$ for $r > 1$, and consider $n = r$. Since the average node weight degree is $\leq K$, there is a node v with weight degree $\leq K$. Consider the subgraph induced by $\mathcal{D} - v$. This subgraph is K -amortized, so suppose we have a valid $K + 1$ generalized coloring of $\mathcal{D} - v$, which exists (by the induction hypothesis). Since v has weight degree at most K , one of the $K + 1$ colors can now be assigned to it to obtain a valid $K + 1$ generalized coloring of G . ■

A.3 Butterfly and Hypercube

Butterfly Consider a random destinations routing problem (B, Π, P) . A trivial lower bound on the routing time is $\lg n$, the length of any path. We will show that the greedy algorithm of Section 3 gives routing time at most $\frac{5}{2} \lg n$ w.h.p., which is optimal up to a constant factor. In order to get this bound, we first show that any packet collides with at most $\frac{3}{2} \lg n$ other packets w.h.p. Thus, the maximum node weight degree in the dependency graph \mathcal{D} is at most $\frac{3}{2} \lg n$. Since the paths are shortcut-free and $D \leq \lg n$, using Lemma 2, we get the bound.

Consider a packet $\pi \in \Pi$ with path $(v_0, v_1, \dots, v_{\lg n})$. Let $m_i, i = 1, \dots, \lg n - 1$, be the number of other packets that could possibly collide with packet π , with the first collision edge being (v_i, v_{i+1}) (note that it is not possible to have collisions on edge (v_0, v_1)). Let q_i be the probability that one of those m_i packets actually uses the edge (v_i, v_{i+1}) . The following lemma follows from the properties of the butterfly network.

Lemma 6. $m_0 = 0$, $m_i = 2^{i-1}$, and $q_i = 2^{-(i+1)}$, for $i = 1, \dots, \lg n - 1$.

Proof. Clearly $m_0 = 0$. Let σ be some packet $\sigma \neq \pi$ that collides for the first time with π on edge (v_i, v_{i+1}) . Packet σ arrives at v_i using edge (w, v_i) with

$w \neq v_{i-1}$. The number of input nodes that can reach w is 2^{i-1} , and σ could have originated from any of these nodes. Thus, $m_i = 2^{i-1}$.

To obtain q_i , we observe that from v_{i+1} , packet σ can reach $M = 2^{\lg n - (i+1)}$ destination nodes. Since the only way to get to these nodes is using the edge (v_i, v_{i+1}) , and since the destination nodes are chosen randomly with uniform probability, the probability that packet σ uses this edge is $q_i = M/n = 2^{-(i+1)}$.

Let X_i be the number of different other packets that collide with packet i . Let $x_j^{(i)}$ be the Bernoulli random variable that equals 1 if packet j collides with packet i , and let $q_j^{(i)} = \mathbf{P}[x_j^{(i)} = 1]$. $X_i = \sum_j x_j^{(i)}$, and $x_j^{(i)}$ are independent for different j . Let $\mu = \mathbf{E}[X_i] = \sum_j q_j^{(i)}$. Using Lemma 6 we obtain:

$$\sum_j q_j^{(i)} = \sum_{k=1}^{\lg n - 1} m_k q_k = \frac{1}{4}(\lg n - 1).$$

Thus, the expected number of packets that use packet i 's path is $\frac{1}{4}(\lg n - 1)$, independent of i , or the specific path used by packet i . Note that in the dependency graph \mathcal{D} , X_i is equal to $W(\pi_i)$. We will use the following version of the Chernoff bound to get a concentration result for X_i :

Lemma 7 ([20]). *Let y_1, \dots, y_n be independent binomial random variables, with $\mathbf{P}[y_i = 1] = b_i$ for $i \in [1, n]$, where $0 < b_i < 1$. Let $Y = \sum_{i=1}^n y_i$, $\mu = \sum_{i=1}^n b_i$. Then, for any $\alpha > 2e$, $\mathbf{P}[Y > \alpha\mu] < 2^{-\alpha\mu}$.*

Define the event E_i by $E_i = \{X_i > \alpha \lg n\}$ for some $\alpha > 2e$. Applying the Chernoff bound in Lemma 7, we get $\mathbf{P}[X_i > \frac{\alpha}{4} \lg n] \leq \mathbf{P}[X_i > \alpha\mu] < \frac{2^{\alpha/4}}{n^{\alpha/4}}$. The identity $\mathbf{P}[\max_i X_i \leq \alpha \lg n] = 1 - \mathbf{P}[\cup_i E_i]$ and the union bound then give

$$\mathbf{P}\left[\max_i X_i \leq \frac{\alpha}{4} \lg n\right] > 1 - n \cdot \frac{2^{\alpha/4}}{n^{\alpha/4}} = 1 - \frac{2^{\alpha/4}}{n^{\alpha/4-1}}$$

Taking $\alpha = 6$, since $\max_i W(\pi_i) = \max_i X_i$, and $D = \lg n$, Lemma 2 then gives the following theorem:

Theorem 11. *For random destination routing problem (B, Π, P) on the n -input butterfly B , the routing time of the greedy algorithm satisfies $\mathbf{P}[rt_{Gr}(B, \Pi, P) \leq \frac{5}{2} \lg n] > 1 - 2\sqrt{2}n^{-\frac{1}{2}}$.*

It is known that there exist permutation routing problems with congestion at least $\Omega(\sqrt{n})$, i.e. some edges are hot-spots (see [20, Section 4.2]). In order to avoid hot-spots, Valiant [15, 16] proposed the following alternative scheme to route permutation routing problems in a butterfly-like network. Take two butterflies and connect them so that the outputs of the first butterfly are the inputs to the second butterfly. The permutation problem is for this ‘‘joint’’ butterfly network: each packet has source on the input of the first butterfly and destination on the output of the second butterfly. The routing idea is to allow each packet

to choose uniformly at random an intermediate node on the output of the first butterfly. The path is then given by source to random intermediate node followed by intermediate node to destination. Such a routing scheme avoids hot-spots – the permutation problem is now equivalent to two random destinations problems. Thus, we can apply Theorem 11 twice to obtain Theorem 5.

Hypercube We consider the n -hypercube network H with $n = 2^k$ nodes, [14]. The distance between any two nodes is $\leq \lg n$. Assume that each node is the source of one packet, and that the destinations are random. We use (left-to-right) *bit-fixing* to determine the paths given the sources and destinations: let π be a packet which has to be routed from source $s(\pi)$ to destination $\delta(\pi)$; flip the leftmost bit at which the labels of $s(\pi)$ and $\delta(\pi)$ differ and send packet π along the edge that leads to the resulting node v ; now repeat this process with v and $\delta(\pi)$, continuing until the path has reached $\delta(\pi)$. Note that bit-fixing paths are shortest paths, since the number of bits flipped is minimum. Further, $D \leq \lg n$, since no more than n bits are flipped.

Consider a random destinations routing problem (H, Π, P) with bit-fixing paths P . We will show that the greedy algorithm of Section 3, has routing time bounded by $7 \lg n$, w.h.p., which is optimal to within constant factors because it can be shown that $D \geq \frac{1}{4} \lg n$ w.h.p (using a simple Chernoff bounding argument). As with the Butterfly analysis, let X_i be the number of other different packets that packet i collides with. We will use the following result which is adapted from [20, Theorem 4.6]:

Lemma 8 ([20]). $\mathbf{P}[\max_i X_i \leq 6 \lg n] > 1 - 1/(32n)$.

Thus, the maximum node weight degree in the dependency graph \mathcal{D} is at most $6 \lg n$, with probability at least $1 - 1/(32n)$. Since $D \leq \lg n$, Lemma 2 implies that the routing time of the greedy algorithm is at most $7 \lg n$ w.h.p. We have the following theorem:

Theorem 12. *For a random destination routing problem (H, Π, P) on the n -hypercube H with bit-fixing paths. The routing time of the greedy algorithm satisfies $\mathbf{P}[rt_{Gr}(B, \Pi, P) \leq 7 \lg n] > 1 - 1/(32n)$.*

It is known that on the n -hypercube, there exist permutation routing problems with congestion at least $\Omega(\sqrt{n/\log n})$, i.e. some edges are hot-spots (see [20, Section 4.2]). In order to avoid hot-spots, we will use Valiant's scheme [15, 16]: for any permutation problem, we will construct paths P' by first taking bit-fixing paths from a source to a random uniformly picked intermediate node, followed by bit-fixing paths from the intermediate node to a destination. This routing problem is the combination of two random destinations problem. Thus, we can apply Theorem 12 twice to obtain Theorem 6.

B Computational Complexity of Direct Routing

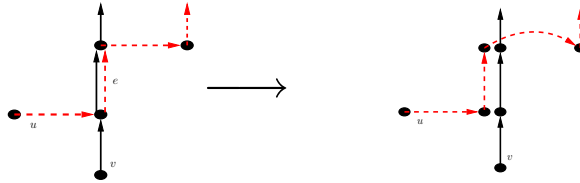
Proof of Theorem 7. We will explicitly demonstrate the reduction from coloring. Illustrated on Figure 2 is a routing problem for which there are N packets, and

all the packets form a synchronized clique of size N in the dependency graph \mathcal{D} . There are L levels in this routing problem. Each path in the figure (ending with an arrow) is the path for 1 packet. The anchor path is the vertical path of length L ($L = 4$ in the figure). Each path can be associated to a level, which denotes the x -coordinate at which the path moves vertically up after making its final left turn. Thus the anchor path is the level-0 path, which begins at coordinates $(0, 0)$ and ends at $(0, L)$. There is a path for every level in $[0, L]$, and so the total number of packets is $N = L + 1$. The level- i path for $i > 0$ begins at $(1 - i, i - 1)$ and ends at $(i, L + i)$, and is constructed as follows. Beginning at $(1 - i, i - 1)$, the path moves right till $(0, i - 1)$, then alternating between up and right moves till it reaches level i at node $(i, 2i - 1)$ (i alternating up and right moves), at which point the path moves up to $(i, L + i)$.

We list some properties of this set of paths. Let $j > i \geq 0$. (i) The level- j path meets the level i path exactly once at the edge from $(i, i + j - 1)$ to $(i, i + j)$. Further, an edge is shared by at most 2 paths. (ii) Every packet is synchronized with every other packet, i.e., if packets π_1, π_2 follow paths p_1, p_2 which share an edge e then $d_{p_1}(e) = d_{p_2}(e)$: this follows from (i) and the fact that the level- i path is injected at $(1 - i, i - 1)$. Thus, if two packets are injected at the same time into two paths p_1, p_2 , then they will conflict at e . (iii) The length of the level i path is $L + 2i$.

Since every pair of packets is synchronized, in the dependency graph \mathcal{D} , the packets form a synchronized clique of size N . Given an instance $I = (G, K)$ of VERTEX COLOR, we now show how to reduce it to the corresponding instance $I' = (G', \Pi, \mathcal{P}, T = K - 1)$ of DIRECT ROUTE. Each node in G corresponds to a packet in Π . The paths are initially as illustrated in the routing problem above with $L = N - 1$. We now show how to transform this routing problem so that the dependency graph \mathcal{D} for the transformed routing problem is isomorphic to G . This is the instance I' to which we reduce I .

Currently the dependency graph is K_N , an N -clique. We need to remove some of the edges to get G . If there is no edge between two nodes u, v in G , this means that the corresponding packets must not collide. We thus alter the two paths corresponding to u, v at their intersection edge e as follows,



Notice that the paths corresponding to u, v no longer collide. Further, the lengths of u and v and their relationships with any other paths have not been altered in any way. The resulting dependency graph is isomorphic to G .

Since every two packets that collide in this routing problem are synchronized, they cannot be assigned the same injection time in any valid schedule. Interpreting the injection time of a packet as the color of that packet, we see that any valid direct routing schedule induces a valid coloring of \mathcal{D} . Since \mathcal{D} is isomorphic to

G , this will also induce a valid coloring of G . Further, a valid coloring of G and hence of \mathcal{D} will induce a valid set of injection times since no two packets that collide (and hence are adjacent in \mathcal{D}) will have the same injection time. Thus the answer to instance I of VERTEX COLOR is true if and only if the answer to I' DIRECT ROUTE is true. The proof is concluded by noting that the construction of I' is clearly polynomial in N . ■

Proof of Theorem 8. Consider the instance of direct routing in the proof of Theorem 7. Let K be the number of collisions allowed and suppose we have a schedule that approximates $\tau_K(G', \Pi, \mathcal{P})$ with ratio r . Let T the maximum injection time. For collision i , pick one of the packets involved and assign it injection time $T+i$. Now none of the packets collide, as they are all synchronized, so we have a valid direct routing schedule with maximum injection time $T+K$. So, $\tau_0(G', \Pi, \mathcal{P}) \leq T+K$, and since $\chi(G) = \tau_0(G', \Pi, \mathcal{P}) + 1$, our approximation for $\chi(G)$ is $T+K+1$, it only remains to show that $(T+K+1)/\chi(G) \leq r+K+1$. Since $\chi(G) \geq 1$, it suffices to prove that $\frac{T}{\chi(G)} \leq r$. Clearly, $\chi(G) \leq T+K+1$. If $T/\tau_K(G', \Pi, \mathcal{P}) \leq r$ then $T/\tau_0(G', \Pi, \mathcal{P}) \leq r$ and so $(T+K+1)/\chi(G) \leq r+K+1$. The approximation is polynomial time since the reduction is polynomial time. ■

C Lower Bounds for Buffering

We construct routing problem B which forces packets to be buffered multiple times. In routing problem A normalize the packets so that all packet lengths are $3L$; in order to do so add edges at the end of the paths when necessary. We then construct a routing problem B by concatenating k identical copies of routing problem A as described next. Let A_i denote the i th copy of A , $1 \leq i \leq k$. In problem B a packet traverses each routing problem A one after the other in sequence. In problem B , the path of a packet π is the concatenation of the paths in each copy of A . When a packet exits A_i , it enters A_{i+1} at exactly the same level as in A_i . In routing problem B , $C = 2c$, $D = 3kL$ and $N = c(L+1) = \Theta(C \cdot D/k)$. The packets in B are synchronized in each A_i , which leads to the following result.

Lemma 9. *For routing problem B , if $T \leq \alpha$, then at least $N - j\alpha$ packets each need to be buffered, at least j times, in A_1, \dots, A_j , where $1 \leq j \leq k$.*

Proof. We prove the claim by induction on j . For $j = 1$ the claim follows immediately from Lemma 5. Let's assume that the claim holds for all $j < r$, where $r > 1$; we will prove that the claim holds for $j = r$.

Let x_i denote the packets which are buffered i times in A_1, \dots, A_{r-1} , where $i \geq 0$. Clearly, $N = \sum_{i \geq 0} x_i$. From those packets, the only packets that may be buffered less than r times in A_1, \dots, A_r , are packets from x_0, \dots, x_{r-1} . Let $X = \sum_{i=0}^{r-2} x_i$; from the induction hypothesis, we have that $X \leq (r-1)\alpha$. Let Y be the packets of x_{r-1} which will not be buffered in A_r . Then, the Y packets are all synchronized in A_r , and so must be injected at different times (Lemma 1). Thus, $T \geq Y$, and so $\alpha \geq Y$. Therefore the number of packets buffered less

than r times in A_1, \dots, A_r is at most $X + Y \leq (r-1)\alpha + \alpha = r\alpha$. Consequently, at least $N - r\alpha$ packets are buffered at least r times in A_1, \dots, A_r .

Proof of Theorem 10. From Lemma 9, at least $N - k\alpha$ packets are each buffered at least k times in routing problem B . Suppose we wish to obtain an $O(N^\epsilon)$ approximation to an optimal schedule, for $0 \leq \epsilon < \frac{1}{2}$. In this case, $\alpha \leq \lambda(C + D)N^\epsilon$ for some $\lambda > 0$. Now, we choose $c = N^{(2-\epsilon)/3}$, $L = N^{(1+\epsilon)/3} - 1$ and $k = AN^{(1-2\epsilon)/3}$ with A chosen so that $\lambda A(2 + 3A) < 1$. We then have that $N - k\alpha \geq (1 - \lambda A(2 + 3A))N + o(N)$, i.e., $N - k\alpha = \Omega(N)$. By Lemma 9, $\Omega(N)$ packets are buffered at least k times. ■