

\tilde{O} (Congestion + Dilation) Hot-Potato Routing on Leveled Networks

Costas Busch*
Rensselaer Polytechnic Institute
buschc@cs.rpi.edu

July 23, 2003

Abstract

We study *packet routing problems*, in which we route a set of N packets on preselected paths with congestion C and dilation D . For *store-and-forward* routing, in which nodes have buffers for packets in transit, there are routing algorithms with performance that matches the lower bound $\Omega(C + D)$. Motivated from optical networks, we study the extreme case of *hot-potato* routing in which the nodes are bufferless. In hot-potato routing, packets may be unable to follow the preselected paths towards the destination nodes; thus it may take more time for packets to be routed. An interesting question is how much is the performance of routing algorithms affected from the absence of buffers.

Here, we answer this question for the general class of *leveled* networks, in which the nodes are partitioned into $L + 1$ distinct levels. We present a randomized hot-potato routing algorithm for leveled networks, which routes the packets in $\tilde{O}(C + L)$ time with high probability. For routing problems with dilation $O(L)$, this bound is within polylogarithmic factors from the lower bound $\Omega(C + L)$. Our algorithm demonstrates that the benefit from using buffers is no more than polylogarithmic; thus, hot-potato routing is an efficient way to route packets in leveled networks.

Our algorithm is *online*, that is, routing decisions are taken at real time at each node, while packets are routed in the network. A novel characteristic of our algorithm is that during the course of routing, packets may deviate from their preselected paths. To our knowledge, this is the *first* hot-potato algorithm designed and analyzed, in terms of congestion and dilation, for arbitrary leveled networks.

1 Introduction

1.1 Background

In a network, nodes communicate by sending packets through the nodes and the links (edges) of the network. In a *packet routing problem*, we are given a set of packets, where each packet has a *source node* and a *destination node*, and we route these packets to their destination nodes. A *routing algorithm* (routing protocol) specifies the decisions that the nodes take while they route the packets. The goal of a routing algorithm is to deliver the packets to their destinations as fast as possible.

*Supported from research funds of Rensselaer Polytechnic Institute.

We consider *hot-potato* (or *deflection*) routing algorithms, which were introduced by Baran [3]. In hot-potato routing, the nodes in the network have no buffers to store packets in transit: any packet that arrives at a node, it is immediately forwarded to another node; in other words, a packet is treated like a “hot potato”. Hot-potato routing algorithms have been observed to work well in practice [4], and have been used in parallel machines such as the HEP multiprocessor [25], the Connection machine [13], and the Caltech Mosaic C [24], as well as in high speed communication networks [19]. Hot-potato routing algorithms are well-suited for optical networks since it is difficult to buffer optical messages [1, 26].

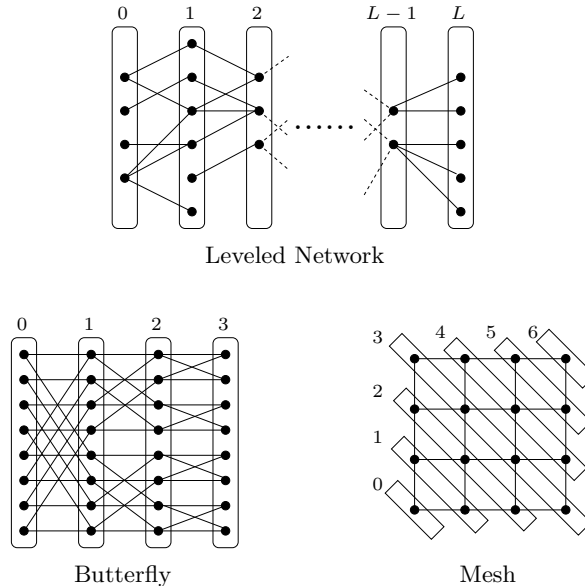


Figure 1: Leveled networks

We study hot-potato routing algorithms for the wide class of *leveled networks*. A leveled network with *depth* L consists from $L + 1$ *levels* of nodes, numbered 0 to L , such that each node belongs to exactly one level, and edges connect nodes that belong to consecutive levels, as shown in Figure 1. Many multiprocessor network architectures can be represented as leveled networks. Typical examples are the *butterfly network* and the *mesh network*, for which specific cases are shown in Figure 1. In particular, the mesh network can be viewed in four different ways as a leveled network, according to which corner node is level 0. Other examples of networks that can be treated as leveled networks are the *shuffle-exchange network*, *multidimensional array*, *hypercube*, and *fat-tree* [16].

We consider leveled networks in which the nodes are *synchronous*; namely, time is discrete, and at each time step, a node receives packets, makes a routing decision, and then forwards the packets to the adjacent nodes. At each time step, a node is allowed to send at most one packet per link.* We study a special case of *many-to-one* packet routing problems, in which each node is the source of at most one packet, and each node is the destination of an arbitrary number of packets. Each packet has a *preselected path*, from the source node to the destination node, which the packet follows during routing.† In leveled networks, the paths are oriented from lower levels to

*Note that at any time step at most two packets can traverse a link, one packet at each direction of the link.

†The packet paths are selected before the routing begins. In this work we do not consider how these paths are

higher levels; that is, from left to right on the network on the top of Figure 1.

There are two parameters associated with the preselected paths that influence the performance of any routing algorithm: the *congestion* C , the maximum number of packets that traverse any edge in the network, and the *dilation* D , the maximum length of any path. Since at most one packet can traverse any edge at a time step, there is a trivial lower bound for any routing problem: $\Omega(C + D)$. In leveled networks the path length does not exceed L , and there are routing problems for which $D = O(L)$ for which the lower bound is $\Omega(C + L)$. It is desirable to design routing algorithms with routing time that matches this lower bound.

In hot-potato routing, an interesting situation occurs when two or more packets appear in the same node at the same time and all of them wish to follow the same link; note that there could be up to C such packets. This situation represents a conflict among the packets, since only one of them can follow the link. One of these packets will successfully follow the link, while the rest packets are sent to alternative links, since there is no buffer to store these packets in the current node; we say that such packets are *deflected*. Due to deflections, a packet may not be able to follow the links of its preselected path towards the destination node. This makes the analysis of hot-potato algorithms challenging.

1.2 Contribution

We present a new hot-potato routing algorithm for leveled networks. Our algorithm is randomized and routes N packets in $O((C+L) \ln^9(LN))$ time steps with probability at least $1 - 1/LN$. Note that this time is optimal within polylogarithmic factors. Our algorithm enjoys the following attributes:

- It is an *online* algorithm: all routing decisions are made in the nodes during the course of routing.
- It is a *local* (distributed) algorithm: the routing decisions of the nodes depend only on their local states, and the packets they receive.
- The algorithm works for any leveled network, and its performance doesn't depend on the edge degrees of the nodes.

A high level description of our algorithm is as follows. We separate the packets randomly and uniformly in $O(C)$ disjoint sets, which we call *frontier-sets*, such that the congestion among packets of the same frontier-set is at most $\ln(LN)$. We route the packets of each frontier-set separately inside areas of the network that we call *frontier-frames* (see figure 2); there is a frontier-frame for each frontier-set. A frontier-frame contains a polylogarithmic number, in terms of L and N , of consecutive levels. The algorithm consists of a sequence of phases, the duration of each phase is also polylogarithmic. The position of a frontier-frame changes at each phase. Initially, a frontier-frame appears at the lowest level of the network, and then, at each subsequent phase, the frontier-frame shifts to one level higher. Different frontier-frames are “pipelined” one after the other, without overlapping, and all frontier-frames shift simultaneously.

A packet of a frontier-set is injected into the network at the right moment, when the respective frontier-frame passes through the source node. While the packet is routed to its destination node, the packet remains inside the frontier-frame; that is, while the frontier-frame shifts to higher levels, the packet shifts together with it. The packet reaches its destination node, when its frontier-frame

selected, but how to design fast routing algorithms given the paths.

passes through that node. Thus, by the time when the frontier-frame reaches the highest-level in the network, all the packets of the respective frontier-set must have been delivered to their destinations. The algorithm terminates when the last frontier-frame reaches the highest level in the network, at which point all the packets in the network have been delivered to their destinations. We show that this event occurs no later than time $O(C + L)$ multiplied by polylogarithmic factors.

In the analysis, we establish that due to the logarithmic congestion between packets of the same frontier-set, these packets will remain within a polylogarithmic number of levels away from each other. This property enables the packets to remain inside the frontier-frame during the course of routing. An interesting characteristic of our algorithm is that due to deflections, packets may deviate from their preselected paths. We show in the analysis of our algorithm that this doesn't affect significantly the performance of the algorithm, since when packets are inside their frontier-frames, they stay very close to their preselected paths (within polylogarithmic distance).

We would like to note that due to the big polylogarithmic factor in the time expression, our algorithm is not really practical, in the sense of direct applicability. Nevertheless, the analysis shows that it is at least possible to come close to the optimal $O(C + L)$ bound, which might motivate searching for more practical algorithms that come to the same or even better performance guarantee.

1.3 Related Work

Hot-potato routing algorithms have been studied for specific network multiprocessor architectures such as the 2-dimensional mesh and torus [5, 9, 10, 12, 14], the d -dimensional mesh [5, 7], the hypercube [8, 12], and trees [2]. Meyer auf der Heide and Scheideler [20] study the more general class of *vertex symmetric networks*. For more about multiprocessor architectures you can look at [15]. Bhatt *et al.* [6] study hot-potato routing on leveled networks, but for different routing problems than the ones we consider here.

Leveled networks have been studied in the context of store-and-forward routing, by Leighton *et al.* [16], where they present an $O(C + L + \log N)$ randomized algorithm with constant size buffers. For store-and-forward routing, there has been a lot of research for obtaining optimal $O(C + D)$ algorithms for arbitrary networks [17, 18, 21, 22, 23].

The technique of decreasing congestion, by assigning packets to different sets, was first used in [17]. In our algorithm (Section 3) packets have states with different priorities; this particular technique was introduced in [11]. To our knowledge, there is no previous work on hot-potato routing on arbitrary networks, not even on leveled networks, for the packet routing problems we consider here (routing problems analyzed in terms of congestion and dilation).

Paper Outline

We proceed in our paper as follows. In Section 2, we give some necessary preliminaries. In Section 3, we present our algorithm; the time analysis of our algorithm appears in Section 4. In section 5, we conclude with a discussion.

2 Preliminaries

2.1 Basic Formulas and Definitions

We use the following parameters.[‡]

$$a = \frac{2e^3}{\ln(LN)} \quad m = \ln^2(LN) + 5 \quad q = \frac{1}{m^2 \ln(LN)}$$

$$w = 4em^2 \ln(LN) \ln(p_1^{-1}) + 3m + 1$$

$$p_0 = 1 - \frac{1}{2LN} \quad p_1 = \frac{1}{(amC + L)2amCLN^2}$$

We define the function $p(k)$ for any $k \geq 0$, as follows.

$$p(k) = \begin{cases} p_0 & (k = 0) \\ p(k-1)(1 - amCNp_1) & (k > 0) \end{cases}$$

We use the following inequalities. For all real numbers n and k , such that $n \geq 1$ and $|k| \leq n$,

$$e^k \left(1 - \frac{k^2}{n}\right) \leq \left(1 + \frac{k}{n}\right)^n \leq e^k. \quad (1)$$

For all real numbers n and k , such that $0 < n < 1$ and $k \geq 1$,

$$1 - kn \leq (1 - n)^k. \quad (2)$$

Denote by $Pr(X)$ the probability that event X is true.

2.2 Paths

In a leveled network with depth L , we denote an edge as $e = (v_1, v_2)$ such that nodes v_1 and v_2 are in respective levels ℓ and $\ell + 1$, where $0 \leq \ell < L$; that is, edges are oriented from lower levels to higher levels.[§] A *path* is a sequence of nodes connected by edges e_1, e_2, \dots, e_n (we denote a path with the edge sequence). A *valid path* is any path e_1, e_2, \dots, e_n in which for any two consecutive edges $e_i = (v_i, v_j)$ and $e_{i+1} = (v_k, v_l)$, it holds $v_j = v_k$; that is, the nodes in a valid path are in consecutive levels of the network starting from a lower level and ending at a higher level. A *subpath* of a path is any subsequence of consecutive edges of the path. Note that any subpath of a valid path is also a valid path. The *length* of a path is the number of its edges.

For any packet π , the *preselected path* of the packet is a valid path $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)$, where v_1 is the *source node* of π and v_n is its *destination node*. For convenience, we assume that the path of a packet is stored in the header of a packet in the form of a list of edges which we refer to as the *path list*.

[‡]Some of the parameters should be integers, but we don't use the ceiling operation on them to avoid notational clutter; this doesn't affect our results significantly.

[§]We avoid to use the term *directed edge*, since during routing, the edges are used in both directions.

2.3 Routing

Initially, a packet waits in the source node until it is *injected* into the network. After the packet is injected we say that the packet is *active*. An active packet follows the links of its preselected path until the packet reaches its destination node (where the packet is *absorbed*). When a packet follows a link then we remove that link from the packet's path list. Thus, at any time step, the path list of a packet consists from the subpath (of its preselected path), starting from the current node, at which the packet resides, and ending at the destination node. We will refer to this subpath as the *current path* of the packet. Clearly, at time 0 the current path of a packet is its preselected path.

If a packet π moves from a lower level to a higher level then we say that the packet moves *forward*, otherwise that the packet moves *backward*. We say that two or more packets *meet* if they appear in the same node at the same time step. We say that a packet is *injected in isolation* if at the time step it is injected into the network it doesn't meet with other packets at the source node. A packet injected in isolation is guaranteed to follow the first link of its preselected path. As we will prove in Section 4, in our algorithm packets are always injected in isolation.

Let's assume that at time t packet π is in node v at level $\ell > 0$ and π wishes to follow link e towards level $\ell + 1$. A *conflict* occurs when other packets appear at node v and time t that wish to follow link e . Only one of these packets will follow link e . If packet π loses in the conflict then it is sent to an alternative link $e' \neq e$ connected to v , and we say that packet π is *deflected*. If link e' connects node v to level $\ell - 1$ ($\ell + 1$) then we say that the deflection is backward (deflection is forward), and at time $t + 1$ packet π appears in level $\ell - 1$ ($\ell + 1$). Note that when π is deflected it may not be directed on the link it followed the previous time step. Thus, packet π may not be able to stay on its preselected path during the course of routing.

On a deflection, we update the current path of a packet as follows. Let g denote the current path of packet π at time t . When packet π is deflected at time t and sent on edge e' , we update the current path of packet π so that at time $t + 1$ the first link is e' and the rest is g . In the case where edge e' was traversed in the opposite direction by another packet σ at the previous time step $t - 1$, and σ didn't follow edge e' due to a deflection, then we call the deflection of packet π a *safe deflection*. Essentially, in a safe deflection the edge e is transferred from the path list of packet σ to the path list of packet π . In other words, with safe deflections, edges are "recycled", that is, transferred from the path list of one packet to the path list of another packet. This property is important in our algorithm for preserving the congestion during routing, as we will show in the analysis.

We show that deflections can always be backwards and safe; this guarantees that current paths of packets are valid.

Lemma 2.1 *If packets are injected in isolation, then packets are deflected backwards, deflections are safe, and the current paths of packets are valid.*

Proof: We prove the claim by induction on time t . For the basis case, we consider time step 0. All the packets that appear in the network at time step 0 must have just been injected in isolation; thus, there are no deflections and the current paths of packets are trivially valid. Let's assume now that the claim holds for any time step t' , where $0 < t' < t$. We will prove that the claim holds for time step t .

First, we show that the current paths of packets at time step t are valid. Consider a packet π which is in node v at time step t . Let g denote the current path of π at time step $t - 1$. From the

induction hypothesis, g is valid. Furthermore, there are only three possible ways that π arrived at v : (i) packet π is injected at time t , and thus its current path is trivially valid, or (ii) packet π moved forward at time $t-1$ to node v (without being deflected at time $t-1$), and thus the current path of π in v is a subpath of g , which is valid, or (iii) packet π got deflected at time $t-1$ and moved backwards to node v by following some edge e , and thus the current path of π in v is edge e followed by g , which is a valid path. Therefore, in all cases the current path of π is still valid.

Next, we prove that all deflections at time t are backwards and safe. Notice that at time t all current paths are valid and thus packets wish to move forward. Consider a conflict situation in a node v of level ℓ and at time t , in which $x > 1$ packets wish to follow the same edge e towards level $\ell + 1$. At time step $t-1$, it must be that from the x packets at least $x-1$ moved forward, without being deflected, from level $\ell-1$ to node v (since current paths are valid); let E denote the set of edges that these packets followed (clearly, $|E| \geq x-1$). At time step t , exactly one packet moves forward on edge e . The rest $x-1$ packets have to be deflected; they can be deflected backwards to level $\ell-1$ by following any of the edges of set E . Thus, at time t deflections are backwards and safe. ■

2.4 Congestion

For any edge e and any time step t , the *edge congestion* c_e^t of e is defined as the number of packets that have on their current paths the edge e ; we take into account active and non-active packets. The congestion C^t at time t is defined as the maximum c_e^t taken over all the edges e of the network. Denote $C = C^0$, namely, C is the congestion of the preselected paths. At time t , the dilation D^t is defined as the maximum length of any current path. Denote $D = D^0$, namely, D is the dilation of the preselected paths. From Lemma 2.1, we know that if packets are injected in isolation then the current paths of packets are valid. Since the length of a valid path cannot exceed L , we have for any time step $t \geq 0$, $D^t \leq L$.

The congestion C determines the maximum number of packets that could meet at the same node at the same time, and wish to follow the same edge. To reduce the congestion we separate the packets into aC sets $S_0, S_1, \dots, S_{aC-1}$, which we call *frontier-sets*. Each packet belongs to exactly one frontier-set and this set is chosen uniformly and at random among the aC frontier-sets, before routing begins. Clearly, $N = \sum_{i=0}^{aC-1} |S_i|$.

For any frontier-set S_i , edge e , and time t , the *frontier-set edge congestion* $c_{e,i}^t$ is defined as the number of packets of S_i that include e on their current paths; we take into account active and non-active packets. The *frontier-set congestion* C_i^t at time t is defined as the maximum frontier-set edge congestion $c_{e,i}^t$, taken over all the edges e of the network. Let $c_{e,i} = c_{e,i}^0$ and $C_i = C_i^0$, namely, $c_{e,i}$ and C_i correspond to the frontier-set congestion of the preselected paths of packets in S_i . Using a Chernoff-type bound, we can show, with high probability, that the congestion of the preselected paths in all the frontier-sets is no more than $\ln(LN)$.

Lemma 2.2 *For all i , where $0 \leq i \leq aC-1$, $C_i \leq \ln(LN)$ with probability at least p_0 .*

2.5 Phases, Frontiers, and Target Nodes

In our algorithm, time is divided into consecutive time periods called *phases* (starting from phase 0). Each phase consists of m consecutive time periods called *rounds* (starting from round 0). Each round is w time steps long; each phase is mw time steps long. The *beginning* (*end*) of a time period is its first (last) time step.

A *frontier* f is a pointer that points to a level $\ell = f$, as shown in Figure 2. In our algorithm, there are aC frontiers $f_0, f_1, \dots, f_{aC-1}$, as many as the frontier-sets. At phase 0, $f_i = -im$, for all $0 \leq i \leq aC - 1$. At the beginning of each subsequent phase, the frontier f_i increases by one (f_i moves forward). Thus, at the beginning of phase im , frontier f_i points to level 0 in the network, and during the next L phases, the frontier f_i points to some subsequent level in the network. Note that all frontiers move forward simultaneously, and any frontier f_i points to some level in the network only for $L + 1$ phases.

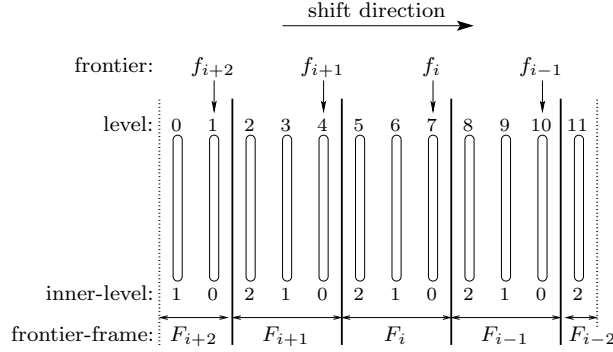


Figure 2: The frontier-frames of a leveled network with $L = 11$ and $m = 3$

The *frontier-frame* F_i , of frontier f_i , consists of levels $f_i, f_i - 1, \dots, f_i - m + 1$, for all $0 \leq i \leq aC - 1$, as shown in Figure 2. Actually, at any time step, the frontier-frame consists only of those levels that exist in the network, which depends on where f_i points to; for example, in Figure 2, the frontier-frames F_{i+2} , and F_{i-2} are not complete. We call the levels of a frontier-frame F_i the *inner-levels* of F_i , and we number them from 0 to $m - 1$; thus, inner-level k corresponds to level $f_i - k$, for all k , where $0 \leq k \leq m - 1$. A frontier-frame F_i shifts forward (one level higher) whenever the respective frontier f_i moves forward. Note that all frontier-frames shift forward simultaneously, and that different frontier-frames do not overlap.

Each frontier-frame F_i has a *target level*. The target level changes at each round of phase. At rounds 0 and 1, the target level of F_i is its inner-level 0. At each subsequent round j of the current phase, where $2 \leq j \leq m - 1$, the target level shifts one level backwards, such that it points to inner-level $j - 1$ of F_i . Note that at each round, the target level shifts backwards, while at each phase, the frontier-frame shifts forward. The packets of frontier-set S_i are associated with frontier-frame F_i . At any time step, any packet $\pi \in S_i$ has a *target node* which is defined as follows. Let g denote the current path of π . If g is valid and g contains a node v in the target level of F_i (that is, g goes through the target level) then v is the target node of π ; otherwise, the target node of π is its destination node.

3 The Algorithm

We describe the algorithm in terms of the actions of some arbitrary packet π of frontier-set S_i , during some particular phase. The actions of the rest packets of frontier-set S_i are similar to the actions of packet π . First, we give a high level description of the algorithm. Let's assume that packet π is in its frontier-frame F_i , in a round $j > 0$ of the current phase. Let's assume that during

round j the current path of π is valid and the target node v of π is in the target level of F_i . The goal of packet π is to reach the target node v by following the current path towards v .

When packet π reaches the target node v , then packet π enters a *wait* state. In the wait state, packet π waits at the target node by *oscillating* (moving back and forth) between the target node and an adjacent node in the immediate higher inner-level. If packet π enters the wait state on target node v , and remains in the wait state, uninterrupted from other packets, until the end of the current round j , then packet π will wait on node v for every subsequent round until the end of the current phase.

Since we consider hot-potato routing, there is a possibility that during round j packet π may fail to reach target node v , or packet π may fail to remain in the wait state on node v , due to conflicts with other packets. In this case, packet π attempts again in the next round $j + 1$ to reach the new target node and enter the wait state on that node; note that in round $j + 1$, the target node appears in one inner-level higher, since the target level of F_i moves backwards one level. This procedure is repeated in the subsequent rounds. In the analysis we show that at some round of the current phase, packet π will eventually successfully reach some target node in F_i and remain in the wait state on that node.

The effect of the algorithm is that the packets of frontier-set S_i first attempt to wait at the lower inner-levels of frontier-frame F_i , and if they fail then they attempt to wait at higher inner-levels. Since the target level moves backwards, packets that already wait at lower inner-levels will not be affected by packets that attempt to enter the wait state at higher inner-levels at subsequent rounds. Consequently, at the end of the phase, packets accumulate at the lower inner-levels of F_i (near frontier f_i), where they wait on their target nodes, and the higher (at least the last three) inner-levels of F_i are free from packets. Thus, when at the next phase frontier-frame F_i shifts one level forward, all the packets of S_i appear inside the new position of F_i ; essentially, the packets of set S_i have shifted with F_i (even though the packets haven't really moved). Every time that frontier-frame F_i shifts, the packets of frontier-set S_i shift together with F_i , and the packets are getting closer to their destinations.

In our algorithm, in order to guarantee that packet π reaches its target level, there are two more packet states: **normal** and **excited**. Originally, packet π is in the **normal** state and follows its current path to the target node. Frequently though, packet π changes its state and becomes an **excited** packet, which has the highest priority. The **excited** packet π has very good chances to reach its target node, uninterrupted from other packets. We now give the detailed description of the algorithm for packet π .

Packet Injection. Packet π is injected into the network at the beginning of the phase in which its source node is in inner-level $m - 1$ of frontier-frame F_i ; that is, π is injected at the last inner-level of F_i . In the extreme case, where packet π is unable to be injected at that particular time, because all links connected to the source node are occupied by other packets, then packet π will be injected at any subsequent time step in which there is an available link from the source node.[¶]

Packet State. The active packet π has a state which is one of the following: **normal**, **excited**, **wait**. States have priorities and the order of priorities from higher to lower is: **excited**, **normal**, **wait**. In conflicts, packets with higher priority win over packets with lower priority. Conflicts among packets of the same priority are resolved arbitrarily. If at any time step a packet reaches its destination node then the packet is absorbed. The state of packet π changes as follows.

- **Normal state.** When packet π is injected into the network, its priority is set to **normal**. In

[¶]In the analysis we show that this extreme case doesn't occur (with high probability).

the **normal** state, packet π follows the current path towards its target node.

- **Excited state.** At any time step t , the **normal** packet π attempts to change its priority, and becomes an **excited** packet with probability q . In the **excited** state, packet π follows the current path towards its target node. If at any time step, the **excited** packet is deflected, then it becomes a **normal** packet. At the end of each round, the **excited** packet π becomes a **normal** packet.
- **Wait state.** At any time step t that the **excited** or **normal** packet π reaches its target node v , packet π enters the **wait** state. Let $e = (v', v)$ be the last link that π traversed when it reached the target node v . In the subsequent time steps, packet π remains in the **wait** state by oscillating on edge e , that is, packet π moves back and forth between nodes v and v' .^{||} If packet π is not deflected until the end of the current round, while it is in the **wait** state on node v , then π remains in the **wait** state on node v for all subsequent rounds, until the end of the current phase. If however, at any time step the **wait** packet π is deflected from other packets, then it becomes **normal**. At the end of each phase, the **wait** packet π becomes a **normal** packet.

4 Analysis of the Algorithm

In the analysis of our algorithm, we determine how much time it is needed until all packets are absorbed. Since our algorithm works in phases, we will prove that during each phase our algorithm satisfies some specific correctness conditions which guarantee the fast delivery of the packets to their destinations. These correctness conditions are stated in the form of invariants which are preserved from one phase to the next, with high probability. In particular, given N packets, we will prove that the following invariants hold at the end of each phase $k \geq 0$, with probability at least $p(k)$.

I_a^k : Up to the end of phase k , packets are injected in isolation.

I_b^k : Up to the end of phase k , packets are deflected backwards, deflections are safe, and the current paths of packets are valid.

I_c^k : Up to the end of phase k , active packets with frontier-frame F_i are always inside F_i .

I_d^k : Up to the end of phase k , packets of different frontier-sets do not meet.

I_e^k : At any time step t up to the end of phase k , $C_i^t \leq \ln(LN)$.

I_f^k : At the end of phase k , all active packets with frontier-frame F_i appear in an inner-level $\ell \leq m - 4$.

The most important invariant is I_c^k , which states that the packets of frontier-set S_i remain inside the frontier-frame F_i for the duration of routing. Eventually all the packets of frontier-set S_i will be absorbed while frontier-frame F_i passes through the respective destination nodes. The rest of the invariants are important in order to prove the truth of invariant I_c^k . Invariant I_f^k is the most difficult to prove. It states that at the end of each phase, the last three inner-levels of

^{||}Note that the oscillation of packet π on edge e can be thought of as a safe deflection, in which the edge e remains in the path list of π , while π oscillates on e .

frontier-frame F_i are empty from packets; thus, at the next phase, when F_i shifts one level forward, all the packets of F_i appear again inside F_i ; essentially, this means that the packets of frontier-set S_i shift together with F_i . We prove the truth of all the invariants by induction on the phase k .

Basis case ($k = 0$). According to the algorithm, in phase 0, no packets are injected into the network, so invariants $I_a^0, I_b^0, I_c^0, I_d^0$ and I_f^0 , are trivially true. From Lemma 2.2, invariant I_e^0 is also true with probability at least p_0 .

Induction hypothesis ($k < n$). Assume invariants $I_a^{n-1}, \dots, I_f^{n-1}$ hold with probability at least $p(n-1)$.

Induction step ($k = n$). We will prove that the invariants I_a^n, \dots, I_f^n are true with probability at least $p(n)$. In Sections 4.1, and 4.2, we prove the truth of invariants I_a^n, \dots, I_f^n , given that the invariants $I_a^{n-1}, \dots, I_f^{n-1}$ hold deterministically. In section 4.3, we determine the probability of the induction step, by including the probability $p(n-1)$ of the induction hypothesis; this completes the proof of the induction step. In section 4.4, we determine the total time of our algorithm.

4.1 Proof of Invariants $I_a^n, I_b^n, I_c^n, I_d^n$, and I_e^n

Consider the packets of frontier-set S_i , for some particular i , where $0 \leq i \leq aC - 1$.

Lemma 4.1 *At the beginning of phase n , any active packet π with frontier-frame F_i , which was active at the previous phase, appears in an inner-level $\ell \leq m - 2$.*

Proof: Packet π was active at the end of phase $n-1$. Therefore, from the truth of invariant I_f^{n-1} , at the end of phase $n-1$, packet π is in an inner-level ℓ_{n-1} of the frame F_i such that $\ell_{n-1} \leq m-4$. Since at the beginning of phase n frontier-frame F_i shifts to one higher level, we have that ℓ_{n-1} corresponds to inner-level ℓ_n in phase n , such that $1 \leq \ell_n \leq m-3$. Since packet π could have been deflected, or move forward at the end of phase $n-1$, we have that packet π appears in the beginning of phase n in inner-level ℓ such that $\ell \leq m-2$. ■

Proposition 4.2 I_a^n is true.

Proof: From Lemma 4.1, at the beginning of phase n there are no active packets, which were active at the previous phase, in inner-level $m-1$ of frontier-frame F_i . From the description of the algorithm, we have that packets are injected only at the beginning of phase n , and at inner-level $m-1$ of frontier-frame F_i . Recall that we consider routing problems in which at most one packet is injected at any node. Therefore, packets are injected in isolation at the beginning of phase n . Thus, since I_a^{n-1} holds, I_a^n is true. ■

From Invariant I_b^{n-1} , Proposition 4.2, and Lemma 2.1, we obtain:

Proposition 4.3 I_b^n is true.

From Lemma 4.1, and the proof of Proposition 4.2, we obtain:

Lemma 4.4 *At the beginning of phase n , any active packet π with frontier-frame F_i appears in an inner-level $\ell \leq m - 1$.*

Lemma 4.5 *At any time step during phase n , any active packet π with frontier f_i appears in some level ℓ of the network with $\ell \leq f_i$.*

Proof: From Lemma 4.4, at the beginning of phase n , the packet π with frontier f_i appears in a level ℓ with $\ell \leq f_i$. During phase n , the rightmost target node (considering all rounds) of packet π is in level f_i . From the description of the algorithm, and from Proposition 4.3, packet π never moves to a level higher than f_i during phase n . Therefore, $l \leq f_i$. ■

Lemma 4.6 *During phase n , any active packet with frontier f_i , appears in a level higher than f_{i+1} .*

Proof: Assume for contradiction that there is a packet π with frontier f_i that appears in level f_{i+1} or lower during phase n . From Lemma 4.4, at the beginning of phase n , packet π appears in a level higher than f_{i+1} . In order for packet π to appear in a subsequent time step in level f_{i+1} or lower, it must be that packet π got deflected at time t from some node v of level $f_{i+1} + 1$ to a node v' of level f_{i+1} , and thus, it followed the edge $e = (v', v)$. Let's assume, without loss of generality, that packet π is the first packet deflected from level $f_{i+1} + 1$ to level f_{i+1} ; thus, t is the earliest time that such a deflection occurs. Note that t cannot be the first time step of phase n , since from the proof of Proposition 4.2, at that time, all packets at level f_{i+1} are newly injected packets, which move forward.

It must be that packet π got deflected because of conflicts with other packets that appear in node v at time t . From Proposition 4.3, we have that deflections are safe, and thus there is a packet σ which appears in node v at time t , such that at the previous time step packet σ traversed edge e , and then the edge e was transferred from the path list of σ to the path list of π .** From Lemma 4.5, packet σ must be at the beginning of phase n in a level higher than f_{i+1} (packet σ must have frontier f_k , where $k \leq i$). Furthermore, packet σ must have been deflected from level $f_{i+1} + 1$ to level f_{i+1} at a time step earlier than t , which violates the assumption that π is the first packet deflected from level $f_{i+1} + 1$ to level f_{i+1} . A contradiction. ■

Combining Lemmas 4.4, 4.5 and 4.6 we obtain:

Lemma 4.7 *During phase n , all active packets with frontier-frame F_i remain inside F_i .*

From Lemma 4.7 and invariant I_c^{n-1} , we obtain:

Proposition 4.8 I_c^n is true.

Since any two frontier-frames do not overlap, Proposition 4.8 and invariant I_d^{n-1} , imply together that packets with different frontiers do not meet during phase n .

Proposition 4.9 I_d^n is true.

We continue with the proof of invariant I_e^n . First, we show that during phase n the frontier-set congestion doesn't increase.

Lemma 4.10 *At any time step t during phase n , $C_i^t \leq \ln(LN)$, for all $0 \leq i \leq aC - 1$.*

**Note that packet π cannot possibly be in the `wait` state and oscillate on e , since v cannot be a target node for π .

Proof: Consider any edge $e = (v_1, v_2)$ in the network and a frontier-set S_i . Let $c'_{e,i}$ denote the frontier-set edge congestion of e at the end of phase $n - 1$. Write $c^t_{e,i} = c'_{e,i} + k$. From Proposition 4.9, during phase n , packets of frontier-set S_i meet only with packets of the same frontier-set. Furthermore, from Proposition 4.3, deflections are safe. Thus, every time $t' \leq t$, a packet π of frontier-set S_i traverses edge $e = (v_1, v_2)$ from node v_2 to v_1 (deflection), another packet σ of frontier-set S_i must have traversed edge e at the previous time step from v_1 to v_2 , and edge e is transferred from the path list of packet σ to the path list of packet π .^{††} Therefore, packet π increases k by one, while packet σ decreases k by one. Thus, k does not increase more than 0, which implies that $c^t_{e,i}$ doesn't increase more than $c'_{e,i}$. Subsequently, C_i^t doesn't increase more than C'_i , where C'_i denotes the frontier-set congestion of S_i at the end of phase $n - 1$. From Invariant I_e^{n-1} we have that $C'_i \leq \ln(LN)$. Therefore, $C_i^t \leq \ln(LN)$, as needed. ■

From invariant I_e^{n-1} , and Lemma 4.10, we obtain:

Proposition 4.11 I_e^n is true.

4.2 Proof of Invariant I_f^n

We consider time steps only in phase n . We also consider the packets of a particular frontier-set S_i for some $0 \leq i \leq aC - 1$. From Proposition 4.8, we know that during phase n , the active packets of frontier-set S_i are inside frontier-frame F_i . From Proposition 4.3 the current paths of packets are valid, and thus the target nodes of these packets are inside frontier-frame F_i .

First we consider round 0 of phase n . Let A denote the set of active packets with destinations inside frontier-frame F_i . From the description of the algorithm, we have that during round 0 the target nodes for these packets is also their destination nodes. We prove that all packets in A will be absorbed in round 0 with high probability. We consider the following events.

X_π : Packet $\pi \in A$ is absorbed in round 0.

X : All packets of A are absorbed in round 0.

We will find a lower bound on probability $Pr(X_\pi)$ and then we will use this bound to determine a lower bound on probability $Pr(X)$. We need to show first that excited packets have a good chance to reach their target nodes (destination nodes). This will help us to estimate the probability that packets reach their target nodes (no matter their state), since packets become excited with probability q .

Lemma 4.12 For any particular time step t , and any particular edge $e = (v_0, v_1)$ inside F_i , the event that node v_0 does not contain any excited packets that wish to traverse e , occurs with probability at least $(1 - mq)^{\ln(LN)}$.

Proof: Let ℓ be the inner-level of node v_0 . Let π be an excited packet that appears at node v_0 and at time t and wishes to follow e . Packet π must have become excited with probability p , at some node v' in a inner-level $\ell' \geq \ell$ of F_i , and at time $t' = t - (\ell' - \ell)$. Packet π has exactly one chance to become excited at node v' so that it appears at node v_0 at time t ; according to

^{††}Note that this fact holds even when packet π is in the wait state and oscillates on e , since in that case e remains to π .

the algorithm, this chance is given to π at time t' with probability q . Since there are at most m inner-levels like ℓ' , the probability that packet π gets **excited** in any of these levels and then appears at node v_0 and time t is at most mq . Therefore, the probability that π doesn't appear at node v_0 at time t in the **excited** state, is at least $1 - mq$. From Proposition 4.11, there are at most $\ln(LN)$ packets similar to π that wish to traverse edge e . Therefore, none of these $\ln(LN)$ packets appear in the **excited** state in node v_0 at time t with probability at least $(1 - mq)^{\ln(LN)}$. ■

Let t_0 denote the first time step of round 0. Notice that if a packet π becomes **excited** before time step $t_0 + w - m - 1$, then π might be able to reach its destination node before the end of round 0.

Lemma 4.13 *If a packet π becomes **excited** at any particular time t , where $t_0 \leq t \leq t_0 + w - m - 1$, then π will reach its destination node with probability at least $1/2e$.*

Proof: Let's assume that packet π is in node v at time t and wishes to follow the edge $e = (v_0, v_1)$ of F_i . Packet π will not follow the edge e only if it conflicts with other **excited** packets on edge e . From Lemma 4.12, with probability at least $(1 - mq)^{\ln(LN)}$, there is no other **excited** packet on node v and time t , that wishes to follow e ; thus packet π will successfully follow edge e with at least that probability. Since there are less than m edges on the path of π towards the destination node, starting from node v_0 , the **excited** packet π will successfully follow all these edges and reach the destination node with probability at least $(1 - mq)^{m \ln(LN)}$. From the definition of q , and Equation 1 we obtain,

$$(1 - mq)^{m \ln(LN)} \geq \left(1 - \frac{1}{m \ln(LN)}\right)^{m \ln(LN)} \geq \frac{1}{2e}.$$

■

Since a packet becomes excited with probability q , we obtain from Lemma 4.13:

Lemma 4.14 *At any particular time step t , where $t_0 \leq t \leq t_0 + w - m - 1$, the event that a **normal** packet changes its state and becomes an **excited** packet, which then reaches its destination node, occurs with probability at least $q/2e$.*

We are now ready to estimate the probability $Pr(X_\pi)$.

Lemma 4.15 *For any packet $\pi \in A$, $Pr(X_\pi) \geq 1 - p_1$.*

Proof: Let \bar{X}_π denote the event in which packet π doesn't reach its destination node at the end of round 0; clearly, $Pr(\bar{X}_\pi) = 1 - Pr(X_\pi)$. During the time period $[t_0, t_0 + w - m - 1]$, packet π traverses $w - m - 1$ links. From these link traversals, at most $(w - m - 1)/2 + m$ are towards the destination node, and at least $(w - m - 1)/2 - m$ are taking π further from the destination node, due to deflections from other packets. Immediately after each deflection, packet π is always in the **normal** state. Therefore, from Lemma 4.14, packet π has a chance after each deflection to become **excited** and then reach its destination at a subsequent time step with probability at least $q/2e$, and fails to do so with probability at most $1 - q/2e$. Since there are at least $(w - m - 1)/2 - m$ deflections, packet π , will fail to take its chance after each deflection to become **excited** and reach

its destination with probability at most $Pr(\bar{X}_\pi) \leq (1 - q/2e)^{(w-m-1)/2-m}$. From the definition of q and w , and Equation 1, we obtain

$$\begin{aligned} Pr(\bar{X}_\pi) &\leq \left(1 - \frac{1}{2em^2 \ln(LN)}\right)^{2em^2 \ln(LN) \ln(p_1^{-1})} \\ &\leq \frac{1}{e^{\ln(p_1^{-1})}} = p_1. \end{aligned}$$

Subsequently, $Pr(X_\pi) \geq 1 - p_1$. ■

We continue to prove:

Lemma 4.16 $Pr(X) \geq 1 - |A|p_1$.

Proof: Let $\pi \in A$. From Lemma 4.15, we have that packet π is not absorbed in round 0 with probability $Pr(\bar{X}_\pi) \leq p_1$. Let \bar{X} denote the event that some packet $\pi \in A$ is not absorbed in round 0; clearly, $Pr(X) = 1 - Pr(\bar{X})$. We have, $Pr(\bar{X}) = \sum_{\pi \in A} Pr(\bar{X}_\pi) \leq |A|p_1$. Subsequently, $Pr(X) \geq 1 - |A|p_1$. ■

Now we consider round j of phase n , where $m - 1 \geq j \geq 1$. Denote B_j the set of active packets in round j , which are not in the `wait` state at the beginning of round j . We consider the following events.

Y_π^j : Packet $\pi \in B_j$, reaches its target node in round j .

Y^j : All packets of B_j reach their target nodes in round j .

Assume for the moment that event X is given, and that for each event Y^{j+1} , event Y^j is given, where $m - 1 \geq j \geq 2$. With proofs very similar to the proofs of Lemmas 4.15 and 4.16 we obtain the following two lemmas.

Lemma 4.17 For any packet $\pi \in B_j$, $Pr(Y_\pi^j) \geq 1 - p_1$, for any j , $m - 1 \geq j \geq 1$.

Lemma 4.18 $Pr(Y^j) \geq 1 - |B_j|p_1$, for any j , $m - 1 \geq j \geq 1$.

We now prove that the number of packets at each round which are not in the `wait` state, is decreasing. The reason is that at each round a big fraction of the packets become `wait` packets.

Lemma 4.19 At the end of any particular round j , where $m - 1 \geq j \geq 1$, the event that at least $|B_j|/\ln(LN)$ packets of set B_j are in the `wait` state, occurs with probability at least $1 - |B_j|p_1$.

Proof: In round j , the target nodes of the packets in B_j are in inner-level $j - 1$ of frontier-frame F_i . Let E denote the set of edges between inner-levels $j - 2$ and $j - 1$, from which the packets of B_j approach their target nodes. From Lemma 4.18, during round j every packet of B_j reaches its target node with probability at least $1 - |B_j|p_1$. According to the algorithm, when a packet reaches its target node then it becomes a `wait` packet on the edge of E which is connected to its target node. Notice that a `wait` packet π can be deflected only by another packet σ (`excited` or `normal`) which becomes a `wait` packet on the same edge with π . Therefore, with probability at least $1 - |B_j|p_1$, at least one packet will become a `wait` packet on each edge of E . From Proposition 4.11, the maximum capacity of any edge in frontier-frame F_i is $\ln(LN)$. Therefore, $|E| \geq |B_j|/\ln(LN)$. Thus, at least $|B_j|/\ln(LN)$ packets enter the `wait` state, with probability at least $1 - |B_j|p_1$. ■

Now, we consider all rounds j with $m - 1 \geq j \geq 2$. We only assume now that the event X is given.

Lemma 4.20 *For all j , where $m - 1 \geq j \geq 2$, $|B_j| \leq |B_{j-1}|(1 - 1/\ln(LN))$ with probability at least $(1 - |S_i|p_1)^{m-2}$.*

Proof: Consider some round j , where $m - 1 \geq j \geq 2$. From Lemma 4.19, we have that at least $|B_{j-1}|/\ln(LN)$ packets of B_{j-1} are **wait** packets at the end of round j , with probability at least $1 - |B_{j-1}|p_1$. Therefore, the packets which are not in a **wait** state, and continue in a non-**wait** state to the next round, are at most $|B_{j-1}| - |B_{j-1}|/\ln(LN)$. Therefore, $|B_j| \leq |B_{j-1}|(1 - 1/\ln(LN))$ with probability at least $1 - |B_{j-1}|p_1$. Considering now all the rounds j , where $m - 1 \geq j \geq 2$, we have that $B_j \leq B_{j-1}(1 - 1/\ln(LN))$ with probability at least $\prod_{j=2}^{m-1} (1 - |B_{j-1}|p_1)$. Since $|B_j| \leq |S_i|$, for all j where $m - 1 \geq j \geq 2$, this probability is at least $(1 - |S_i|p_1)^{m-2}$. ■

Next, we show that higher inner-levels do not contain any active packets.

Lemma 4.21 *For all j such that $m - 1 \geq j \geq \ln^2(LN) + 2$, $|B_j| = 0$ with probability at least $(1 - |S_i|p_1)^{m-2}$.*

Proof: From Lemma 4.20, for all j where $m - 1 \geq j \geq 2$, $|B_j| \leq |B_{j-1}|(1 - 1/\ln(LN))$, with probability at least $(1 - |S_i|p_1)^{m-2}$. Therefore, with at least that probability, $|B_j| \leq |B_1|(1 - 1/\ln(LN))^{j-1}$. Since $|B_1| \leq N$, $|B_j| \leq N(1 - 1/\ln(LN))^{j-1}$. From Equation 1, when $j \geq \ln^2(LN) + 2$, then $(1 - 1/\ln(LN))^{j-1} < 1/N$, and thus $|B_j| < 1$, which implies that $|B_j| = 0$. Therefore, $|B_j| = 0$ for all j such that $m - 1 \geq j \geq \ln^2(LN) + 2$, with probability at least $(1 - |S_i|p_1)^{m-2}$. ■

Now we consider all rounds of phase n , including round 0, without making any assumptions (without assuming that X is given).

Lemma 4.22 *At the end of phase n , there are no active packets in any inner-level j of frontier-frame F_i , where $m - 1 \geq j \geq \ln^2(LN) + 2$, with probability at least $(1 - |S_i|p_1)^m$.*

Proof: From Lemma 4.16, $Pr(X) \geq 1 - |A|p_1$. Multiplying this probability with the probability of Lemma 4.21, and since $|A| \leq |S_i|$, we have that there are no active packets in any of these inner-levels, with probability at least $(1 - |S_i|p_1)^{m-2} \cdot (1 - |A|p_1) \geq (1 - |S_i|p_1)^{m-1} \geq (1 - |S_i|p_1)^m$. ■

Now, we consider all frontier-frames F_i , where $0 \leq i \leq aC - 1$.

Lemma 4.23 *At the end of phase n , there are no active packets in all inner-levels j of all frontier-frames F_i , where $m - 1 \geq j \geq \ln^2(LN) + 2$, and $0 \leq i \leq aC - 1$, with probability at least $1 - amCNp_1$.*

Proof: From Lemma 4.22, the claim holds for any particular frontier-frame F_i , with probability at least $(1 - |S_i|p_1)^m$. Considering all frontier-frames we have that the claim holds with probability at least $\prod_{i=0}^{aC-1} (1 - |S_i|p_1)^m$. Since $|S_i| \leq N$, for all i where $0 \leq i \leq aC - 1$, we have that this probability is at least $(1 - Np_1)^{amC}$. From Equation 2, this probability is at least $1 - amCNp_1$. ■

From Lemma 4.23, and invariant I_f^{d-1} , we obtain:

Proposition 4.24 *Invariant I_f^n is true with probability at least $1 - amCNp_1$.*

4.3 All Invariants

We have shown in Propositions 4.2, 4.3, 4.3, 4.9, 4.11, and 4.24, that when the invariants $I_a^{n-1}, \dots, I_f^{n-1}$, hold deterministically, then invariants I_a^n, \dots, I_e^n are deterministically true and, invariant I_f^n is true with probability at least p_1^{mN} . Since from the induction hypothesis, invariants $I_a^{n-1}, \dots, I_f^{n-1}$ hold with probability at least $p(n-1)$, we have that the invariants I_a^n, \dots, I_e^n hold with probability at least $p(n-1)$ and invariant I_f^n holds with probability at least $p(n-1)(1 - amCNp_1) = p(n)$. Thus, all invariants I_a^n, \dots, I_f^n hold with probability at least $p(n)$. This completes the proof of the induction step.

4.4 Total Time

Invariant I_c^k implies that any packet π remains inside its respective frontier-frame from the moment that the packet is injected into the network until it is absorbed in its destination node. A frontier-frame traverses the network from a lower level to a higher level, and at some point of time leaves the network completely. By that time, all the packets of the frontier-frame must have been absorbed. Therefore, by the time when the last frontier-frame F_{aC-1} leaves the network, all packets in the network have been absorbed. Frontier-frame F_{aC-1} leaves the network at the beginning of phase $(aC-1)m + L + m = amC + L$, which occurs at time step $(amC + L)mw = am^2wC + mwL$. Since invariant I_c^{amC+L} holds with probability at least $p(amC + L)$, we obtain:

Proposition 4.25 *By time $am^2wC + mwL$, all packets will be absorbed with probability at least $p(amC + L)$.*

We are now ready to prove our main theorem.

Theorem 4.26 *By time $O((C + L) \ln^9(LN))$, all packets are absorbed with probability at least $1 - 1/LN$.*

Proof: From Proposition 4.25 and from the definitions of a , m , and w , we have that all packets will be absorbed by time $O((C + L) \ln^9(LN))$ with probability at least $p(amC + L)$. By unfolding the function $p(amC + L)$, and from Equation 2, we have that this probability is at least

$$\begin{aligned} p(amC + L) &= p_0(1 - amCNp_1)^{amC+L} \\ &= p_0 \left(1 - \frac{amCN}{(amC + L)2amCLN^2} \right)^{amC+L} \\ &\geq p_0 \left(1 - \frac{1}{2LN} \right) = \left(1 - \frac{1}{2LN} \right)^2 \geq 1 - \frac{1}{LN}. \end{aligned}$$

■

5 Discussion

We presented an $\tilde{O}(C + L)$ hot-potato routing algorithm for leveled networks. An immediate application of our algorithm is on routing in multiprocessor networks which are represented as leveled networks. For example, in [16] the authors describe how to obtain optimal paths for the $n \times n$ mesh with congestion and dilation n , and our algorithm can be used to route these packets with time close to the optimal up to polylogarithmic factors. It is interesting to extend our work for arbitrary network topologies.

Acknowledgments

We thank the reviewers of this conference for their comments and corrections. We are indebted to Marios Mavronicolas for his thorough comments and suggestions. We also thank Malik Magdon-Ismail and Roger Wattenhofer for helpful discussions.

References

- [1] A. S. Acampora and S. I. A. Shah. Multihop lightwave networks: a comparison of store-and-forward and hot-potato routing. In *Proc. IEEE INFOCOM*, pages 10–19, 1991.
- [2] S. Alstrup, J. Holm, K. de Lichtenberg, and M. Thorup. Direct routing on trees. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 342–349, 1999.
- [3] P. Baran. On distributed communications networks. *IEEE Transactions on Communications*, pages 1–9, 1964.
- [4] C. Bartzis, I. Caragiannis, C. Kaklamanis, and I. Vergados. Experimental evaluation of hot-potato routing algorithms on 2-dimensional processor arrays. In *EUROPAR: Parallel Processing, 6th International EURO-PAR Conference*, pages 877–881. LNCS, 2000.
- [5] A. Ben-Dor, S. Halevi, and A. Schuster. Potential function analysis of greedy hot-potato routing. *Theory of Computing Systems*, 31(1):41–61, Jan./Feb. 1998.
- [6] S. N. Bhatt, G. Bilardi, G. Pucci, A. G. Ranade, A. L. Rosenberg, and E. J. Schwabe. On bufferless routing of variable-length message in leveled networks. In *Proc. European Symposium on Algorithms*, pages 49–60, 1993.
- [7] A. Borodin, Y. Rabani, and B. Schieber. Deterministic many-to-many hot potato routing. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):587–596, June 1997.
- [8] J. T. Brassil and R. L. Cruz. Bounds on maximum delay in networks with deflection routing. *IEEE Transactions on Parallel and Distributed Systems*, 6(7):724–732, July 1995.
- [9] A. Broder and E. Upfal. Dynamic deflection routing on arrays. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 348–358, May 1996.
- [10] C. Busch, M. Herlihy, and R. Wattenhofer. Hard-potato routing. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 278–285, May 2000.
- [11] C. Busch, M. Herlihy, and R. Wattenhofer. Randomized greedy hot-potato routing. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 458–466, Jan. 2000.
- [12] U. Feige and P. Raghavan. Exact analysis of hot-potato routing. In IEEE, editor, *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 553–562, Pittsburgh, PN, Oct. 1992.
- [13] W. D. Hillis. *The Connection Machine*. MIT press, 1985.

- [14] C. Kaklamani, D. Krizanc, and S. Rao. Hot-potato routing on processor arrays. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 273–282, Velen, Germany, June 30–July 2, 1993.
- [15] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*. Morgan Kaufmann, San Mateo, 1992.
- [16] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao. Randomized routing and sorting on fixed-connection networks. *J. Algorithms*, 17(1):157–205, 1994.
- [17] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-scheduling in $O(\text{congestion} + \text{dilation})$ steps. *Combinatorica*, 14:167–186, 1994.
- [18] T. Leighton, B. Maggs, and A. W. Richa. Fast algorithms for finding $O(\text{congestion} + \text{dilation})$ packet routing schedules. *Combinatorica*, 19:375–401, 1999.
- [19] N. F. Maxemchuk. Comparison of deflection and store and forward techniques in the Manhattan street and shuffle exchange networks. In *Proc. IEEE INFOCOM*, pages 800–809, 1989.
- [20] F. Meyer auf der Heide and C. Scheideler. Routing with bounded buffers and hot-potato routing in vertex-symmetric networks. In P. G. Spirakis, editor, *Proceedings of the Third Annual European Symposium on Algorithms*, volume 979 of *LNCS*, pages 341–354, Corfu, Greece, 25–27 Sept. 1995.
- [21] F. Meyer auf der Heide and B. Vöcking. Shortest-path routing in arbitrary networks. *Journal of Algorithms*, 31(1):105–131, Apr. 1999.
- [22] R. Ostrovsky and Y. Rabani. 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*, pages 644–653, New York, May 1997.
- [23] Y. Rabani and É. Tardos. Distributed packet switching in arbitrary networks. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 366–375, Philadelphia, Pennsylvania, 22–24 May 1996.
- [24] C. L. Seitz. The caltech mosaic C: An experimental, fine-grain multicomputer. In *Proceedings of the 4th Symp. on Parallel Algorithms and Architectures*, June 1992. Keynote Speech.
- [25] B. Smith. Architecture and applications of the HEP multiprocessor computer system. In *Proceedings of the 4th Symp. Real Time Signal Processing IV*, pages 241–248. SPIE, 1981.
- [26] T. Szymanski. An analysis of “hot potato” routing in a fiber optic packet switched hypercube. In *Proc. IEEE INFOCOM*, pages 918–925, 1990.