

OPTIMAL BOUNDS FOR MATCHING ROUTING ON TREES*

LOUXIN ZHANG†

Abstract. The permutation routing problem is studied for trees under the matching model. By introducing a novel and useful (so-called) caterpillar tree partition, we prove that any permutation on an n -node tree (and thus graph) can be routed in $\frac{3}{2}n + O(\log n)$ steps. This answers an open problem of Alon, Chung, and Graham [*SIAM J. Discrete Math.*, 7 (1994), pp. 516–530].

Key words. matching routing, off-line algorithms, trees

AMS subject classifications. 05C, 68M, 68R

PII. S0895480197323159

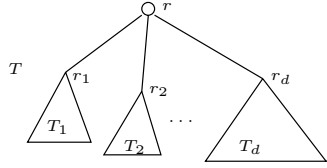
1. Introduction. Routing problems on networks arise in different fields such as communications, parallel architectures, and very large scale integration (VLSI) theory and have been extensively studied in recent years (see [9, 10] for a comprehensive survey). In this paper, we study permutation routing under the matching model, which was proposed by Alon, Chung, and Graham [2]. The routing of this type is described as follows. Given a graph $G = (V, E)$ with vertex set V and edge set E . Initially, each vertex v of G is occupied by a “packet” p . To each packet p is associated a destination $\pi(v) \in V$ so that distinct packets have distinct destinations, where the permutation π is usually called a routing assignment. Packets can be moved to different vertices of G under the protocol, which at each step a set S of edges sharing no endpoints of G is selected, and packets at the endpoints of each edge in S are interchanged. The goal of routing is to route all the packets to their destinations in a minimum number of parallel steps. The routing model considered here is interesting as its basic building blocks are very simple. Like hot-potato routing or deflection routing [1, 4], the striking feature of the matching model is that it involves no message queues at each node. In addition, best-known off-line routing algorithms for the hypercube, the linear array, and the mesh can be implemented in this routing model [6, 10, 12]. Such a kind of off-line routing algorithms has also been generalized in various Cayley networks [3].

In their paper, Alon, Chung, and Graham [2] investigated this routing problem for a variety of popular networks including trees, complete (bipartite) graphs, hypercubes, expander graphs, and Cayley graphs. One of their interesting results is that any permutation on a tree with n nodes can be routed in at most $3n$ steps, which was improved to $\frac{13}{5}n$ in [13] and then $2n$ in [5, 7]. They also conjecture that the optimal upper bound for trees is $\frac{3}{2}n$. In [13], Roberts, Symvonis, and Zhang proved that any permutation can be routed in at most $n + o(n)$ steps on an n -node d -ary complete trees in which the root has degree d and any other internal node has degree $d + 1$ for some fixed $d > 0$. Furthermore, relation routing under a variant of the matching model

*Received by the editors June 17, 1997; accepted for publication (in revised form) March 5, 1998; published electronically January 29, 1999. This work was done at the University of Waterloo and was partially supported by a CGAT grant. A preliminary version of this paper was presented at the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 1997.

<http://www.siam.org/journals/sidma/12-1/32315.html>

†BioInformatics Center, Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Kent Ridge, Singapore 119613 (lxzhang@krdl.org.sg).

FIG. 1. Partition a tree T .

are also studied by Krizanc and Zhang [8] and Pantziou, Roberts, and Symvonis [11] independently.

This paper favors their conjecture. After a number of refinements are incorporated in Alon, Chung, and Graham's algorithm and their analysis, we obtain a simple $2n$ upper bound for routing on trees in section 2. In section 3, by introducing a novel tree partition and extending our approach, we prove that any permutation can be routed in $\frac{3}{2}n + O(\log n)$ steps on any n -node tree. (As a consequence, any permutation on a n -node graph G can be routed in at most $\frac{3}{2}n + O(\log n)$ steps since G has an n -node spanning subtree.) This bound is quite sharp since the lower bound for the problem is $\frac{3}{2}n$ (see [2]).

2. A simple $2n$ -step algorithm. Let T be a tree. For a permutation assignment π on T , we define $rt(\pi, T)$ to be the minimum number of steps required to route π . Further, we define $rt(T)$, the *routing number* of T [2], by $rt(T) = \max_{\pi} rt(\pi, T)$, where π ranges over all possible permutations on T . We first prove a weak result to demonstrate our approach as a warmup.

THEOREM 2.1. *For any n -node subtree T , $rt(T) \leq 2n$.*

For proving Theorem 2.1, we first introduce some necessary definitions and facts. Given a tree $T = (V, E)$ with vertex set V and edge set E , a class of disjoint subtrees $\{T_i\}_1^s$ is called a *partition* of T if $V(T) = \cup_{i=1}^s V(T_i)$. Let T' be any one of these subtrees. For a routing assignment π , call a packet p that is initially located in T' *improper* if its destination $\pi(p)$ does not belong to T' ; call it *proper*, otherwise. T' is *pure* if it does not contain any improper packets, and is *mixed*, otherwise. Since π is a permutation and each node holds one packet, the number of improper packets located in a subtree T' is always equal to the number of improper packets with destinations in T' during routing. This simple observation plays a crucial role in designing off-line algorithms for matching routing.

Proof. We use the standard divide-and-conquer technique to design a $2n$ -step off-line algorithm for our purpose, which is a refinement of Alon, Chung, and Graham's algorithm [2, Theorem 1].

For any n -node tree T , there is always a node $r \in V(T)$ that minimizes maximum remaining components. Such a node is called the "centroid." Obviously, each subtree formed by removing the centroid r (and all incident edges) has at most $n/2$ nodes (see Figure 1). Let d subtrees be formed after the removal of r , and let $|T_i| \geq |T_{i+1}|$ for all $i \leq d-1$. Then there exists a nonnegative integer s such that

$$(2.1) \quad 1 + \sum_{j=s+1}^d |T_j| \leq |T_1| < 1 + \sum_{j=s}^d |T_j|,$$

where we assume $\sum_{j=d+1}^d |T_j| = 0$. Let T_r be the subtree (of T) consisting of the node r and subtrees T_{s+1}, \dots, T_d . Obviously, T_1, T_2, \dots, T_s and T_r form a partition of T .

The first objective of our algorithm is to move all improper packets into their destination subtrees in the partition $\{T_r, T_1, T_2, \dots, T_s\}$. For any subtree T' in the partition, let $I(T')$ and $P(T')$ denote the sets of improper and proper packets in T' , respectively.

LEMMA 2.2. *All improper packets, with respect to the partition, can be moved into their destination subtrees in at most*

$$(2.2) \quad \max\{|P(T_r)|, |P(T_j)| \mid j \leq s\} + |I(T_r)| + \sum_{j=1}^s |I(T_j)| + s - 1 \leq n + s - 1$$

steps.

Proof. A similar result was proved in [13]. We use a greedy algorithm for moving improper packets into their destination subtrees. In a subtree T' , whenever we can interchange an improper and proper packet so as to bring the improper packet close to its destination subtree, we do it; between the vertex r and the roots r_i of subtrees T_i , whenever we can interchange two improper packets so as to bring them into their destination subtrees, we do it.

Our algorithm consists of two phases. In the first phase, we only move improper packets in each subtree T' towards r' , the root of T' , as many as $m = \max\{|P(T_r)|, |P(T_i)| \mid 1 \leq i \leq s\}$ steps. More specifically, this phase is executed as follows. A node v is said to be in level i of the tree if the distance between it and the root r is i . Let p_v denote the packet at node v before step t . Then packets are moved according to the following rules at step t :

- When $t = 0 \pmod{2}$, the packet p_v in even levels is interchanged with an improper packet in a child of v if p_v is proper and the child of v holds an improper packet. When several children hold improper packets, one child is chosen randomly. Otherwise, p_v is not moved at this step.
- When $t = 1 \pmod{2}$, the packet p_v in odd levels is interchanged with an improper packet in a child of v if p_v is proper and the child of v holds an improper packet. Otherwise, p_v is not moved at this step.

Since there are at most m proper packets in each subtree T' , after this phase is executed, the first improper packet has reached the root r' of T' . Further, in every two steps, a new improper packet will be moved onto r' if there are improper packets. All these facts allow the second phase described below to be executed without delay.

In the second phase, we move improper packets across r into their destination components, as well as keeping on moving up improper packets in each subtree. Before T_r becomes pure, we can move two improper packets into their destination subtrees in every two steps. However, we need extra steps after T_r is pure. After T_r has become pure, the packet p_r at r is proper. If other subtrees still contain improper packets, we have to move p_r into a subtree before moving improper packets cross r and then move it back to r . In order to use the minimum number of steps to move the remaining improper packets, we move the packet p_r to a mixed subtree T' and move p_r back to r when the last improper packet is moved into T' . It can easily be seen that we need one extra step for moving improper packets out of T' for each remaining mixed subtree T' . Since there are at most as many as s such subtrees, the second phase can be finished in at most

$$|I(T_r)| + \sum_{j=1}^s |I(T_j)| + s - 1$$

steps, where the subtractive term -1 is derived from the fact that the last two mixed subtrees become pure during the last step (since either at least two mixed subtrees exist or none).

Overall, the whole procedure takes at most

$$\max\{|P(T_r)|, |P(T_j)| \mid j \leq s\} + |I(T_r)| + \sum_{j=1}^s |I(T_j)| + s - 1 \leq n + s - 1$$

steps. This proves the lemma. \square

Now, we continue the proof of Theorem 2.1. After all improper packets have been moved into their destination subtrees, we route each subtree (in parallel).

Obviously, our algorithm is correct. Let $rt(T)$ denote the number of steps needed for routing a permutation on tree T . Then we have

$$(2.3) \quad rt(T) \leq \max\{rt(T_j), rt(T_r) \mid j \leq s\} + n + s - 1 = rt(T_1) + n + s - 1.$$

Let $k = |T_s|$. Then, $k \geq 1$. By inequality (2.1), $|T_1| \leq |T_r| + k - 1$. Since, by the assumption, $|T_j| \geq |T_s|$ for $j \leq s$,

$$s \leq 1 + \frac{n - |T_1| - |T_r|}{k} \leq 1 + \frac{n - 2|T_1| + k - 1}{k}.$$

Therefore, from (2.3) follows that

$$(2.4) \quad rt(T) \leq \left\lceil \left(1 + \frac{1}{k}\right)n - \frac{2|T_1|}{k} + rt(T_1) + \frac{k-1}{k} \right\rceil.$$

Further, for a 2-node tree T , $rt(T) = 1 \leq 2 \times 2$. Since $rt(T) \geq |T|$, $k \geq 1$, and $|T_1| \leq \frac{n}{2}$, (2.4) implies that

$$rt(T) \leq 2n.$$

This proves the theorem. \square

3. Tight bound for the routing number of trees. In this section, we prove that the routing number of n -node trees is $\frac{3n}{2} + O(\log n)$ by refining our approach used in last section. We first introduce the (so-called) caterpillar partition for trees.

Given a tree T and after the removal of a node v (and all incident edges), T is partitioned into some disjoint subtrees. Such a partition is called a *star partition* of T . Given a positive real number $\gamma < 1$, a star partition is γ -type if each subtree in the partition contains at most $\gamma|T|$ vertices. Recall that every tree has a $\frac{1}{2}$ -star partition. However, for any $\gamma < \frac{1}{2}$, a γ -star partition does not always exist for an arbitrary tree.

A partition of T into subtrees is called a *caterpillar partition* if there is a path P in T such that each subtree is rooted at either a nonendpoint v' in P or a child of endpoints of P (see Figure 2). A subtree is *direct* if its root is on P and *indirect* otherwise. The path P is called the *backbone* of the partition. Given two positive real numbers $\gamma, \beta < 1$, a caterpillar partition is called (γ, β) -type if each indirect subtree has at most $\gamma|T|$ nodes and the number of nodes in all direct subtrees is at most $\beta|T|$. Note that by definition, the backbone of a caterpillar partition contains at most $\beta|T| + 2$ nodes.

The proof of our main theorem will use the following result about tree partition. The theorem is of independent interest.

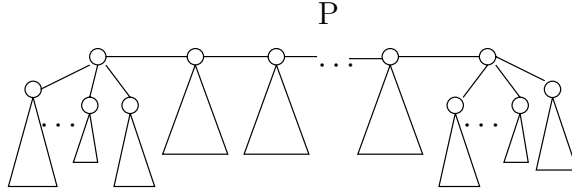


FIG. 2. A caterpillar partition of a tree T . Indirect subtrees are rooted at children of endpoints; direct ones are rooted at internal nodes on the path.

THEOREM 3.1. *Any tree T has either a $\frac{1}{3}$ -star or a $(\frac{1}{3}, \frac{1}{3})$ -caterpillar partition.*

Proof. Let v be a centroid of an n -node tree T . Let B be the set of all nodes u such that when u is removed there is a component not containing v with at least $n/3$ nodes. If B is empty, then the centroid v induces a $\frac{1}{3}$ -star partition. Otherwise, $v \in B$ and B consists of a path in T . This follows from the following facts. First, B induces a subtree. Let $u_1, u_2 \in B$. Then, for any node w on the path between u_1 and u_2 in T , there is a component C not containing v formed after w is removed satisfying the property that C contains u_1 or u_2 , and so $w \in B$. Second, if B does not induce a path, then there are three nodes $u_1, u_2, u_3 \in B$ that are adjacent to a common node $u \in B$. In this case, we have three different components not containing v each with at least $n/3$ after nodes u_1, u_2, u_3 are removed—a contradiction. Hence, B consists of a path in T .

It is easy to see that B forms the backbone of a desired $(\frac{1}{3}, \frac{1}{3})$ -caterpillar partition. \square

THEOREM 3.2. *For any n -node tree T ,*

$$rt(T) \leq \frac{3n}{2} + 9 \log_3 n.$$

Proof. We prove the result by induction. Obviously, the result is trivial for any tree with less than 4 nodes. Thus, we assume that $|T| > 4$ in the rest of our proof.

Let u and v be two leaves in the n -node tree T . They are *sibling* if they have a common parent. Consider a multiset of sibling leaves of T

$$S = \{v_1, v_2, \dots, v_l \mid l \geq 3\},$$

where v_i and v_j are different unless $i = 1$ and $j = l$. If the packet $p(v_i)$ at v_i has destination v_{i+1} for all $i \leq l - 1$, then we can use l steps to move $l - 1$ packets $p(v_i)$, $1 \leq i \leq l - 1$, to their destinations. By induction, we can route the rest of the packets to their destinations in $t \leq \frac{3}{2}(n - l + 1) + 9 \log_3(n - l + 1)$. Since $l \geq 3$, we can route π in at most $l + t \leq \frac{3}{2}n + 9 \log_3 n$ steps. Therefore, in what follows, we assume that T satisfies the following condition:

($\star\star\star$) There are no sibling leaves v_1, v_2 , and v_3 such that the packet $p(v_i)$ has destination v_{i+1} for $i = 1, 2$, where v_1 and v_3 could be same.

By Theorem 3.1, T has either a $\frac{1}{3}$ -star or a $(\frac{1}{3}, \frac{1}{3})$ -caterpillar partition.

Case 1. T has a $\frac{1}{3}$ -star partition.

With T_1, T_2, \dots, T_d denoting subtrees in such a partition, we assume that the “centroid” is r and $|T_i| \geq |T_{i+1}|$ for $i = 1, 2, \dots, d - 1$ (see Figure 1). We consider the following partition of T :

$$\{T_1, T_2, \dots, T_d, \{r\}\}.$$

We route a permutation by first moving all improper packets into their destination subtrees and then route each subtree recursively as in the proof of Theorem 2.1. In order to reach our desired bound, however, we have to overlap the two phases. For simplicity, we assume that the packet p_r located at node r is proper. (Otherwise, our algorithm takes less steps.) Recall that $P(T_i)$ and $I(T_i)$ denote the sets of proper and improper packets in the subtree T_i for $i \leq d$. After moving all improper packets towards r in $\max_{1 \leq i \leq d} |P(T_i)|$ steps, we start to move improper packets out of subtree T_1 by pushing p_r into T_1 . While we move improper packets out of T_1 , some improper packets may be moved out of other subtrees. Once T_1 has become pure (and p_r is back to r , of course), we start to route T_1 immediately, as well as move improper packets out of T_2 . In general, after moving all improper packets out of T_i , we start to route T_i , as well as move improper packets out of other trees T_j ($j \geq i + 1$), if necessary.

We now analyze the algorithm. Let n_i denote the number of steps taken by the algorithm for moving all improper packets out of T_i after all T_j 's ($j < i$) become pure. Since $|T_i| \leq \frac{1}{3}n$, by induction, all packets with destinations in T_i have been moved to their destinations after at most

$$(3.1) \quad t_i = \max_j |P_j| + \sum_{j=1}^i n_j + rt(T_i)$$

steps. Let I_{ij}^k ($i, j \geq k$) denote the number of improper packets moved from T_i to T_j during the phase of moving improper packets out of T_k . Then, it is not difficult to verify the following formulae:

$$(3.2) \quad |I(T_i)| = \sum_{l=1}^i \sum_{j=l}^d I_{ij}^l$$

and

$$(3.3) \quad n_i = 1 + \sum_{j,k \geq i} I_{jk}^i.$$

The additive term 1 in the last formula is derived from the fact that we use one extra step to move the packet p_r , located at r , into T_i at the beginning of moving improper packets out of T_i . Plugging formulae (3.2) and (3.3) into (3.1), we have that

$$(3.4) \quad t_i \leq i + \sum_{j=1}^d |T_j| + rt(T_i).$$

By the condition $(\star \star \star)$, all subtrees have become pure if all improper packets are moved out of all subtrees that have at least 2 nodes. Thus, we need only to consider those multinode subtrees T' . Since there are at most $\frac{n}{2}$ multinode subtrees, formula (3.4) can be refined into

$$(3.5) \quad t_i \leq \min \left\{ i, \frac{n}{2} \right\} + \sum_{j=1}^d |T_j| + rt(T_i).$$

For $i \leq 9$, since $|T_i| \leq \frac{n}{3}$,

$$(3.6) \quad \min \left\{ i, \frac{n}{2} \right\} + |T_i| \leq 9 + \frac{n - |T_i|}{2}$$

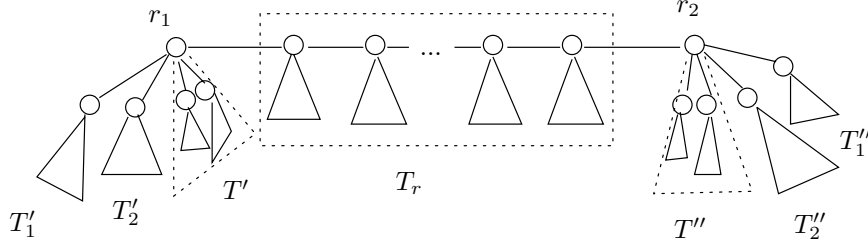


FIG. 3. A $(\frac{1}{3}, \frac{1}{3})$ -caterpillar partition of T . Combine all direct subtrees into a middle subtree and some small indirect subtrees at each endpoint into a larger subtree.

holds.

For $9 < i \leq \frac{n}{3}$, we have $|T_i| \leq \frac{n}{i} < \frac{n}{9}$. Thus inequality (3.6) is also true.

For $i > \frac{n}{3}$, we have $|T_i| \leq 2$. Thus inequality (3.6) is still true.

By the induction hypothesis,

$$rt(T_i) \leq \frac{3|T_i|}{2} + 9 \log_3 |T_i|.$$

Therefore, (3.5) becomes

$$t_i \leq \frac{3(n - |T_i|)}{2} + 9 + rt(T_i) \leq \frac{3n}{2} + 9 \log_3 |T|,$$

which implies that

$$rt(\pi, T) = \max_{1 \leq i \leq d} t_i \leq \frac{3n}{2} + 9 \log_3 |T|.$$

Case 2. T has a $(\frac{1}{3}, \frac{1}{3})$ -caterpillar partition.

Let such a partition \mathcal{P} be illustrated in Figure 3. The backbone P has endpoints r_1 and r_2 ; indirect trees around the endpoints r_1 and r_2 are T'_i , $1 \leq i \leq m$, and T''_j , $1 \leq j \leq n$, respectively. Without loss of generality, we assume that

$$|T'_i| \geq |T'_{i+1}|,$$

$$|T''_j| \geq |T''_{j+1}|,$$

for all possible i and j . By definition,

$$(3.7) \quad \sum_{i=1}^m |T'_i| + 1 > \frac{1}{3}n$$

and

$$(3.8) \quad \sum_{i=1}^n |T''_i| + 1 > \frac{1}{3}n.$$

Let T_r denote the union of all directed trees. Set

$$M = \max\{|T_r|, |T'_1|, |T''_1|\}.$$

Since \mathcal{P} is $(\frac{1}{3}, \frac{1}{3})$ -caterpillar partition, $M \leq \frac{1}{3}n$. Further, by inequality (3.7), there exists c such that

$$\sum_{i=c+1}^m |T'_i| + 1 \leq M < \sum_{i=c}^m |T'_i| + 1,$$

where the sum is 0 if $c=m+1$. Similarly, there exists d such that

$$\sum_{j=d+1}^n |T''_j| + 1 \leq M < \sum_{j=d}^n |T''_j| + 1.$$

Let T'_{r_1} be the union of $T'_{c+1}, T'_{c+2}, \dots, T'_m$ and r_1 , and let T''_{r_2} be the union of $T''_{d+1}, T''_{d+2}, \dots, T''_n$ and r_2 . Then disjoint subtrees $T_r, T'_{r_1}, T''_{r_2}, T'_i$ ($1 \leq i \leq c$), and T''_j ($1 \leq j \leq d$) form a partition of T (see Figure 3). For routing permutation π we move all packets into their destination subtrees in the partition and then route each subtree in parallel as in Theorem 2.1. First, we establish the following fact, a generalization of Lemma 2.2 in section 2, the proof of which is somewhat lengthy and involved.

LEMMA 3.3. *Let c' and d' denote the numbers of indirect subtrees T'_i and T''_j satisfying $|T'_i|, |T''_j| \geq 2$, respectively. Then, all improper packets can be moved into their destination subtrees in at most*

$$(3.9) \quad n + c' + d' + \left\lceil \frac{c - c'}{2} \right\rceil + \left\lceil \frac{d - d'}{2} \right\rceil$$

steps.

Proof. Note that $c - c'$ and $d - d'$ are the numbers of indirect, 1-node subtrees around r_1 and r_2 in the partition, respectively. The algorithm for routing improper packets into their destination subtrees consists of two phases.

Let x denote the total number of proper packets with respect to the partition of T , which is described above. Note that an improper packet in each direct subtree has its destination in some indirect subtree around endpoints. In the first phase, improper packets are moved up as many as x steps in each direct or indirect subtree. This phase is detailed in the proof of Theorem 2.1. Roughly speaking, in each direct or indirect subtree T' , whenever we can interchange an improper and proper packet so as to bring the improper packet close to the root $r(T')$ of T' , we do it.

Recall that after the first phase, in each direct or indirect subtree T' the vertices that improper packets occupy form a subtree T'' . In particular, the root of each subtree T' holds an improper packet. Furthermore, if there exist some improper packets out of T'' in T' , one improper packet can be moved into T'' in every two steps. This fact guarantees that all improper packets will be moved out of each subtree (and into their destination subtrees) during the second phase.

In the second phase, improper packets will be moved into their destination subtrees. For convenience, we denote all nodes on the backbone P from r_1 to r_2 as

$$v_0(= r_1), v_1, v_2, \dots, v_k(= r_2).$$

We also assume that k is an odd integer (the case that k is even is similar). Let p_i denote the packet at the node v_i before step t . At step t , we move the packets according to the following rules:

- When $t = 0 \pmod{2}$, for every i such that $0 \leq i \leq \lfloor k/2 \rfloor$, (1) if either p_{2i+1} has destination in a subtree around r_1 and p_{2i} does not, or p_{2i} has destination

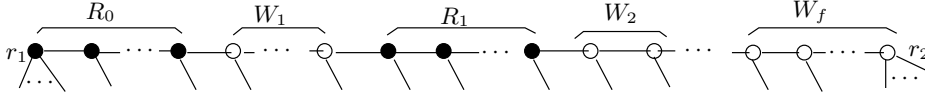


FIG. 4. Packet distribution on the path P . All packets are divided into alternating blocks of red packets and nonred packets, where a red packet is one with destination at some indirect subtree around r_1 .

in a subtree around r_2 and p_{2i+1} does not, then interchange the packet p_{2i} and p_{2i+1} . Otherwise, (2) for $i > 0$, interchange p_{2i} with an improper packet in a subtree right below it if p_{2i} is proper. For $i = 0$, either move p_0 into its destination subtree if p_0 is improper, or move p_0 into a chosen subtree around r_1 (to be specified later) if p_0 is proper and T'_{r_1} is pure.

- When $t = 1 \pmod{2}$, for every i such that $0 \leq i \leq \lfloor k/2 \rfloor$, (1) if either p_{2i+2} has destination in a subtree around r_1 and p_{2i+1} does not, or p_{2i+1} has destination in a subtree around r_2 and p_{2i+2} does not, interchange the packet p_{2i+1} and p_{2i+2} . Otherwise, (2) for $i < \lfloor k/2 \rfloor$, interchange p_{2i+1} with an improper packet in a subtree right below it if p_{2i+1} is proper. For $i = \lfloor k/2 \rfloor$, move p_k into its destination subtree if p_k is improper, or move p_k into a chosen subtree around r_2 (to be specified later) if p_k is proper and T''_{r_2} is pure.

Since r_1 is the parent of all subtrees T'_i , $1 \leq i \leq c$, any improper packet must cross r_1 for being moved into subtrees T'_i . Recall that a subtree is pure if it does not contain any improper packets and is mixed otherwise. After T'_{r_1} is pure, the packet $p(r_1)$ located at root r_1 is proper. If there still exists a mixed subtree T'_i (for some $i \leq c$) around r_1 , the packet $p(r_1)$ has to be moved into a mixed subtree and moved back later. Such moves of $p(r_1)$ cost extra steps. For reducing extra steps, $p(r_1)$ will be moved into the largest mixed subtree T'_j and back to vertex r_1 during the last step before T'_j becomes pure. Such a choice will reduce the number of extra steps spent at r_1 to c , the number of subtrees around r_1 . However, the extra cost can be reduced further if we take care of all 1-node indirect subtrees around r_1 .

When all remaining mixed subtrees around r_1 are 1-node subtrees, by our assumption $(\star\star\star)$, an improper packet having destination in these subtrees is still in the middle subtree T_r or in a subtree around r_2 . We choose a mixed 1-node subtree into which $p(r_1)$ is moved temporarily according to the arrival time of those improper packets. Suppose the first improper packet of this kind has destination in a 1-node subtree T'_j ; we choose any other 1-node, mixed subtree as a temporary destination of $p(r_1)$, unless T'_j is the only one remaining. Such a choice will further reduce the extra steps spent at r_1 from c to $c' + \lceil \frac{c-c'}{2} \rceil$, where c' is the number of multinode subtrees T'_i ($i \leq c$) around r_1 . The strategy of reducing extra cost at r_2 is the same.

Now we use the potential method to analyze the complexity of the second phase, which was first used in [2] for studying matching routing. Notice that the second phase will finish once all improper packets with destinations in subtrees around r_1 and r_2 have been moved into their destination subtrees. Therefore, without loss of generality, we may assume that the improper packet that is moved during the last step has destination in a subtree around r_1 . For analysis purpose, we call a packet *red* if its destination is in a subtree around r_1 and *blue* if its destination is a subtree around r_2 . After each step t , there is a certain distribution of packets on the backbone P . We denote the packet distribution on P in terms of alternating blocks of red and nonred packets as illustrated in Figure 4, where $R_j(t)$ denotes the j th block of red

packets and $W_j(t)$ the j th block of nonred packets, which may contain blue improper packets and proper packets, which have destinations in the middle subtree T_r . Each $R_j(t)$ and $W_j(t)$ is nonempty except $W_{f(t)}$ and $R_0(t)$. Let $I(t)$ denote the set of all improper packets that are off P , and let $g_1(t)$ and $g_2(t)$ denote the numbers of 1-node and multinode, mixed subtrees around r_1 or r_2 after step t , respectively. Finally, define the *potential* $\phi(t)$ after step t by

$$\begin{aligned} \phi(t) = & \max(1, |R_0(t)|) - |R_0(t)| + \max(1, |W_{f(t)}(t)|) \\ & - |W_{f(t)}(t)| + \sum_{j=1}^{f(t)-1} |W_j(t)| - f(t) + |I(t)| \\ & + g_2(t) + g_1(t)/2, \end{aligned}$$

where $|W_j(t)|$ and $|R_j(t)|$ denote the numbers of packets in blocks $W_j(t)$ and $R_j(t)$, respectively.

We will show that if the last red packet is not moved into its destination subtree, the potential $\phi(t)$ must decrease at the next step. In the rest of our proof, we refer to the packet of a block that is closest to r_1 as the *first* packet and the packet that is closest to r_2 as the *last* packet.

During step $t + 1$, the following changes on P may happen:

- Some proper packet p in a nonred block $W_j(t)$, $1 \leq j \leq f$, is interchanged with an improper packet in the subtree right below p , which has destination in an indirect subtree around an endpoint (i.e., r_1 or r_2).
- The last packet in a nonred block $W_j(t)$ ($1 \leq j \leq f(t) - 1$) is interchanged with the first packet of the red block $R_j(t)$.
- In the middle of a nonred block $W_j(t)$ ($1 \leq j \leq f(t)$), some improper packet p having destination in a subtree around r_2 is interchanged with a proper packet right to p , which has destination in the middle subtree T_r .
- If $W_{f(t)}(t)$ is nonempty, the packet $p(r_2)$ at r_2 can be a proper packet (having destination in T''), an improper packet having destination either in a subtree T''_j around r_2 or in the middle subtree T_r . If $p(r_2)$ is a proper packet and there are still mixed subtrees around r_2 , $p(r_2)$ is moved into a chosen mixed subtree; if it is improper and has destination in some T''_j around r_2 , it is moved into T''_j ; if it is improper and has destination in T_r , then it is interchanged with an improper packet to the left of it, which has destination in a subtree around r_2 . Similar events happen at the endpoint r_1 .

Based on these observations, we can prove the following facts.

FACT 1. $|I(t)|$, $g_1(t)$, and $g_2(t)$ is nonincreasing with $t > 0$.

Proof. The proof follows from definitions. \square

FACT 2. Let $t > 0$. For any nonempty blocks $W_i(t)$ and $R_i(t)$, $1 \leq i \leq f(t)$, one of the following occurs during step $t + 1$:

- (i) The last packet of $W_i(t)$ and the first packet of $R_i(t)$ are interchanged, or
- (ii) The last packet of $W_i(t)$ is moved into the subtree right below it. In this case, it is a proper packet with destination in the middle subtree T_r .

Proof. The events listed in the fact do not happen for some i , $1 \leq i \leq f(t)$, only if the last packet of $W_i(t)$ is not compared with either the first packet of $R_i(t)$ or a packet in the subtree right below it. As we will show, this cannot happen when $t > 0$. We prove this fact by induction on t .

The basis case $t = 1$ can easily be verified. Assume that the fact is true for $t' \leq t - 1$. For any nonempty and adjacent blocks, $W_i(t)$ and $R_i(t)$, we consider the

following two cases.

Case 1. $W_i(t)$ is a single-packet block.

Denote the single packet in $W_i(t)$ by q . Consider three subcases.

(a) q was the last packet of some multipacket block $W_{j'}(t-1)$ after step $t-1$. By the induction hypothesis, q is moved right 1 node and formed the single block $W_i(t)$, or the second last packet was interchanged with the red packet right below it. Thus, during step $t+1$, q will be compared with the red packet following it or a packet right below it. So q either moves right or is interchanged with a packet right below it at step $t+1$.

(b) q was the first packet of a two-packet block $W_{j'}(t-1)$ after step $t-1$. Then one of the edges adjacent to the node holding the last packet of $W_{j'}(t-1)$ was active during step t (by the induction hypothesis). This implies that now it is the turn of one of the edges adjacent to node holding q to be active. Thus, q either moves right or is interchanged with a packet right below it at step $t+1$.

(c) q is a packet in the middle of a multipacket block $W_{j'}(t-1)$ after step $t-1$. For q to form a single block after step t , during step t the nonred packet p left to q in $W_{j'}(t-1)$ must be interchanged with a red packet in the subtree right below it and the nonred packet right to q must be moved right one step or moved into a subtree below it. This implies that after step t , the packet right to q is a red packet and also that one of the edges adjacent to the node holding q from the right side is active during step $t+1$. Thus, q either moves right or is interchanged with a packet right below it at step $t+1$.

Case 2. $W_i(t)$ is a multipacket block. By the induction hypothesis, after step $t-1$, the last packet q of the nonred block $W_i(t)$ was either the second last packet of a nonred block $M_{j'}(t-1)$, or it was a middle packet of the block and the packet right to it was moved right one step or moved off the backbone P . This implies that the edges adjacent to q are active during step $t+1$.

Thus we finish the proof of Fact 2. \square

FACT 3. For any $t > 0$, $\phi(t+1) < \phi(t)$.

Proof. Since in the middle of the backbone moving an improper packet into the backbone from a direct subtree decreases $|I(t)|$ by 1 and $f(t)$ at most by 1, it will not increase the potential ϕ . Therefore, we may assume that no improper packets are moved out of direct subtrees in the middle of the backbone P in the rest of our proof.

We consider the following cases.

Case 1. $R_0(t)$ is a nonempty block.

Since no improper packets are moved into the middle of the backbone, by Fact 2, previously stated, the last packet of $W_j(t)$ is interchanged with the first packet of $R_j(t)$. These moves do not decrease $f(t)$, the number of alternating blocks. Now consider the endpoints r_1 . Since R_0 is nonempty, the first red packet is moved into its destination subtree. Thus, the number of improper packets off P , $|I(t)|$, decreases by 1. If the packet swapped out is still red, then the potential ϕ also decreases by 1. Otherwise, $f(t)$ increases by 1 and the potential ϕ decreases by 2. At the endpoint r_2 , if the packet $p(r_2)$ at r_2 does not compare with the packet at v_{k-1} and p_{r_2} is an improper packet having destination in a subtree around r_2 , then it is moved into its destination subtree and a packet q_{out} is swapped out. If q_{out} is red and improper, then $f(t)$ increases by 1 and $|I(t)|$ decreases by 1. Thus, the potential ϕ decreases by at least 2. If q_{out} is blue and proper, then its destination is in the subtree T''_{r_2} and thus $|I(t)|$ decreases by 1. So is ϕ . If T''_{r_2} is pure, then the number of mixed, indirect 1-node subtrees around the endpoints, $g_1(t)$, decreases at least by 2 or the number

of mixed, indirect multinode subtrees around the endpoints, $g_2(t)$, decreases at least by 1; thus ϕ decreases at least by 1. If q_{out} is nonred and improper, the number of improper packets off P , $|I(t)|$, decreases by 1. So is the potential ϕ .

Case 2. $R_0(t)$ is empty.

Since no improper packets are moved into the backbone, by Fact 2, mentioned above, the last packet of $W_j(t)$ is interchanged with the first packet of $R_j(t)$.

If W_1 is a multipacket block, $f(t)$ does not decrease after step $t + 1$. We consider the following subcases.

Subcase 2.1. $W_{f(t)}(t)$ is empty. If $R_{f(t)}(t)$ is a multipacket block, the number of alternating blocks, f , increases at least by 1, which implies that the potential ϕ decreases at least by 1. If $R_{f(t)}(t)$ is a singleton, then $W_{f(t+1)}(t + 1) = 1$, which implies that $\max(1, |W_f|) - |W_f|$ decreases by 1. Since f does not decrease after step $t + 1$, ϕ decreases by 1.

Subcase 2.2. $W_{f(t)}(t)$ is nonempty. Then the last packet p of the block $W_{f(t)}(t)$ is moved into a subtree around r_2 . If p is improper, then it moves into its destination subtree by interchanging with a packet q_{out} . The number of improper packets off P , $|I(t)|$, decreases by 1. If q_{out} is blue or having destination in the middle subtree T_r , then the potential ϕ also decreases by 1. If q_{out} is red and improper, then $f(t)$ increases by 1 and $\max(1, |W_f|) - |W_f|$ increases by 1. Thus, ϕ still decreases by 1. If q_{out} is blue and proper, then its destination is in the subtree T''_{r_2} . If T''_{r_2} is pure, then the number of mixed, indirect 1-node subtrees around the endpoints, $g_1(t)$, decreases at least by 2 or the number of mixed, indirect multinode subtrees around the endpoints, $g_2(t)$, decreases at least by 1; thus ϕ decreases at least by 1. If q_{out} is nonred and improper, then the potential also decreases by 1 because the number of improper packets off P , $|I(t)|$, decreases by 1.

If W_1 is a singleton, then $R_0(t + 1)$ is 1. Then the difference $\max(1, |R_0|) - |R_0|$ decreases by 1 after step $t + 1$. When $W_{f(t)}$ is empty, we consider two cases.

Subcase 2.3. $B_{f(t)-1}(t)$ is a multipacket block.

Then $f(t)$ increases by at least 1 after step $t + 1$, and so the potential ϕ decreases at least by 1 after step $t + 1$.

Subcase 2.4. $B_{f(t)-1}(t)$ is a singleton.

Then $W_{f(t+1)}(t + 1) = 1$. Thus $\max(1, |W_{f(t+1)}|) - |W_{f(t+1)}|$ is 0, which implies that $\phi(t + 1) < \phi(t)$. When $W_{f(t)}$ is nonempty, we consider the following cases. If the last packet at r_2 is improper, then it is moved into its destination subtree and so I decreases 1 after $t + 1$. Thus, ϕ decreases at least by 1. If the last packet at r_2 is proper and T''_{r_2} is pure, then some subtrees around r_2 have become pure after t . Thus, the number of mixed, indirect multinode subtrees around the endpoints, $g_2(t)$, decreases at least by 1 or the number of mixed, indirect 1-node subtrees around the endpoints, $g_1(t)$, decreases at least by 2. Thus, the potential ϕ decreases at least by 1 after step $t + 1$. This finishes the proof of Fact 3. \square

We have proved that the potential decreases with t before the last packet is moved into its destination subtree. By definition, the potential ϕ is at most $m = n - x + c' + d' + \lceil (c - c')/2 \rceil + \lceil (d - d')/2 \rceil$, where x is the number of proper packets contained initially in all subtrees. Thus, the second phase takes at most m steps. Since the first phase takes x steps, our greedy algorithm takes at most $n + c' + d' + \lceil (c - c')/2 \rceil + \lceil (d - d')/2 \rceil$ steps. This completes the proof of Lemma 3.3. \square

We now continue to prove our theorem. Let $B = c' + d' + (c - c')/2 + (d - d')/2$. Then B is bounded above by $5 + \frac{n-3M}{2}$.

LEMMA 3.4. $B \leq 5 + \frac{n-3M}{2}$.

Proof. Let $k' = |T'_c|$ and $k'' = |T''_d|$. Then

$$M \geq |T'_{r_1}| > M - k'$$

and

$$M \geq |T''_{r_2}| > M - k''.$$

If $M = |T_r|$, then

$$c' + \frac{(c - c')}{2} \leq \frac{\sum_{i=1}^m |T'_i| - M + k'}{\max\{k', 2\}} + 1$$

and

$$d' + \frac{(d - d')}{2} \leq \frac{\sum_{j=1}^n |T''_j| - M + k''}{\max\{k'', 2\}} + 1.$$

Since $n = \sum_{i=1}^m |T'_i| + \sum_{j=1}^n |T''_j| + M$,

$$B \leq \frac{n - 3M}{2} + 4.$$

If $M = |T'_1|$, then

$$c' + \frac{(c - c')}{2} = \frac{\sum_{i=1}^m |T'_i| - 2M + k'}{\max\{k', 2\}} + 2$$

and

$$d' + \frac{(d - d')}{2} \leq \frac{\sum_{j=1}^n |T''_j| - M + k''}{\max\{k'', 2\}} + 1.$$

Thus, $B \leq 5 + \frac{n-3M}{2}$. The case $M = |T''_1|$ is treated symmetrically. This finishes the proof of Lemma 3.4. \square

Let $rt(M) = \max_{|T| \leq M} rt(T)$. Combining Lemma 3.4 and (3.9), we have

$$rt(\pi, T) \leq n + 2 + B + rt(M) \leq n + 7 + \frac{n - 3M}{2} + rt(M).$$

By the induction hypothesis, we have

$$rt(\pi, T) \leq \frac{3}{2}n + 9 \log_3 n.$$

This completes our proof. \square

4. Concluding remarks. We have proved the optimal upper bound $\frac{3n}{2} + O(\log n)$ for the routing number of n -node trees. In order to prove the result, the caterpillar partition of trees is introduced. Such a partition is novel and may find some interesting applications in studying open problems remaining for the subject of matching routing (see [2]) and other combinatorial problems related to trees and graphs in general.

Acknowledgments. The author would like to thank J. Buss and M. Li for helpful conversations and A. Symvonis for useful discussions and reading the first draft of this paper. He would also like to thank the referee for helpful suggestions in revising the paper, especially for simplifying the proof of Theorem 3.1.

REFERENCES

- [1] P. BARAN, *On distributed communications networks*, IEEE Trans. Comm. Systems, 12 (1964), pp. 1–9.
- [2] N. ALON, F. R. K. CHUNG, AND R. L. GRAHAM, *Routing permutations on graphs via matchings*, SIAM J. Discrete Math., 7 (1994), pp. 516–530.
- [3] M. BAUMSLAG AND F. ANNEXSTEIN, *A unified framework for off-line permutation routing in parallel networks*, Math. Systems Theory, 24 (1991), pp. 233–251.
- [4] U. FEIGE AND P. RAGHAVEN, *Exact analysis of hot potato routing*, in Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 553–562.
- [5] W. GODDARD, *private communication*, 1997.
- [6] N. HABERMAN, *Parallel Neighbor-Sort (or the Glory of the Induction Principle)*, Tech. report AD-759 248, National Technical Information Services, U.S. Department of Commerce, Springfield, VA, 1972.
- [7] P. HOYER AND K. LARSON, *Permutation Routing via Matchings*, Tech. report 96-16, Institut for Matematik og Datalogi, Odense Universitet, Denmark, 1996.
- [8] D. KRIZANC AND L. ZHANG, *Many-to-one packet routing via matching*, in Proc. 3rd Annual International Conference on Computing and Combinatorics, Shanghai, 1997, Lecture Notes in Comput. Sci. 1276, Springer-Verlag, New York, pp. 11–17.
- [9] T. LEIGHTON, *Methods for message routing in parallel machines*, in Proc. 24th Annual ACM Symposium on Theory of Computing, ACM, New York, 1992, pp. 77–96.
- [10] T. LEIGHTON, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan–Kaufmann, San Francisco, CA, 1992.
- [11] G. PANTZIOU, A. ROBERTS, AND A. SYMVONIS, *Dynamic tree routing under the “matching with consumption” model*, in Proc. 7th International Symposium on Algorithms and Computation, Osaka, 1996, Lecture Notes in Comput. Sci. 1178, Springer-Verlag, New York, pp. 275–284.
- [12] M. RAMRAS, *Routing permutations on a graph*, Networks, 23 (1993), pp. 391–398.
- [13] A. ROBERTS, A. SYMVONIS, AND L. ZHANG, *Routing on trees via matchings*, in Proc. 4th Workshop on Algorithms and Data Structures, Kingston, Ontario, Canada, 1995, Lecture Notes in Comput. Sci. 955, Springer-Verlag, New York, pp. 251–263.