Many-to-One Packet Routing via Matchings

Danny Krizanc[*] Louxin Zhang[†]

Abstract

In this paper we study the packet routing problem under the matching model proposed by Alon, Chung and Graham [1]. We extend the model to allow more than one packet per origin and destination node. We give tight bounds for the many-to-one routing number for complete graphs, complete bipartite graphs and linear arrays. We also present an efficient algorithm for many-to-one routing on an trees (and therefore any graph). Finally, we give bounds for routing arbitrary relations in this model.

1 Introduction

Routing packets arises naturally in the design of large-scale parallel computers and the study of data flow in parallel computing. Packet routing consists of moving packets of data from each node of a network to the other nodes in the network. The goal is to move all of the packets to their desired locations as quickly as possible. Various routing problems have been extensively studied under different models. We refer the reader to [4] for a survey of the topic. In this paper, we study the packet routing problem under the matching model, which was proposed by Alon, Chung and Graham in [1]. The matching model assumes that at each step, a disjoint collection S of links of a network G is selected, and packets at the endpoints of each link in S are interchanged. The permutation routing problem under the matching model was studied in [1, 7, 8]. In [1], many interesting results were obtained on the permutation routing for different classes of networks, such as complete networks, arrays, trees, and expander graphs. For example, they proved that any permutation can be routed in at most $3n$ steps on any n-node tree. Both [7] and [8] proposed new routing algorithms with improved bounds for trees.

The purpose of this note is to investigate general packet routing under the matching model. Formally, the problem can be described as follows. Suppose we are given a connected graph $G = (V, E)$ with vertex set V and edge set E. Initially, each vertex v of G hosts a set $S(v)$ of packets $p(v)$. To each packet is assigned a destination $\pi(v) \in V$. The goal is to route all packets to their destinations in a minimal number of steps. A packet assignment is called

[*]School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ont K1S SB6. E-mail: krizanc@scs.carleton.ca. Partially supported by NSERC.

[†]Department of Computer Science, University of Waterloo, Waterloo, Ont N2L 3G1. E-mail: lzhang@daisy.uwaterloo.ca. Partially supported by CGAT. Part of the work was done while the author was in University of North Texas.

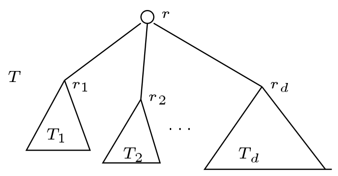

Figure 1: *Decomposing a tree T*

many-to-one if $S(v)$ contains just one packet; it is called *one-to-many* if all packets have different destinations. In a routing problem, multiple packets may have the same destination and may originate at the same node. Thus, for our purposes, we adapt the matching model as followings. If multiple packets are initially in a node, only one of them can leave the node at a step. Once a packet arrives its destination, it is dropped. In other words, after a packet arrives its destination v, v can hold other packets during the subsequent steps.

The rest of this paper is divided into two sections. In section 2, we present optimal routing algorithms for many-to-one assignments on complete graphs, complete bipartite graphs and linear arrays. We also present a reasonably good algorithm for trees (and thus for any graph). In Section 3 we generalize the algorithms to the general routing problem.

2 Many-to-one Routing

Given a many-to-one assignment π on a graph G, as in [1], we define $rt(\pi, G)$ to be the minimal possible number of steps required to route π. We also define the *(many-to-one) routing number*, $rt(G)$, of G by $rt(G) = \max_\pi rt(\pi, G)$.

2.1 Trees

The following result gives an upper bound on the routing number of trees.

Theorem 2.1 *For any n-node tree T, $rt(T) \leq 9n$.*

Proof of Theorem 2.1. We prove Theorem 2.1 by induction on the number of vertices in a tree. For any n-node tree T, it can be decomposed into a set of disjoint subtrees T_i with $|T_i| \leq n/2$ after removing a node r as showed in Figure 1.

Given a many-to-one assignment π on T, we route π using the above decomposition of T. For a packet p, we call a subtree T' a *destination* subtree of p if the destination $\pi(p)$ of p is in T'. We first move all packets to their destination subtrees and then route subtrees (in parallel).

Consider a packet p initially located in a subtree T_i. If its destination does not belong to T_i, we call it *improper*, and *proper* otherwise. Similarly, the packet initially located at the node r is improper if its destination is not r. Without loss of generality, we may assume that the packet at r is proper. (Otherwise, if the packet in r has its destination in T_i, we consider the subtree $r \cup T_i$ instead of T_i.)

In order to move all improper packets to their destination subtrees, we first move all improper packets in each T_i towards r_i, the root of T_i, so that the vertices they occupy form a subtree T_i' of T_i and on every path form r_i to a leaf, all improper packets with destinations in T_j is closer to r_i than those with destinations in T_{j+1} (or T_{j+2} if $j = i-1$) for any $j \neq i$.

Claim. The subtree T_i' can be formed in at most $|T_i|$ steps.
Proof. It can be found in [7].

The next objective is to move improper packets in T_i into their destination subtrees. Since π is a many-to-one assignment, the numbers of improper packets being moved out and being moved in a subtree T_i are not equal in general. Thus, the methods used in both [1] and [7] fails. Here, we use a kind of brute-force method. With $T_1, T_2, \cdots T_d$ denoting the subtrees formed by the removal of the node r, let $I(T_i, T_j)$ denote the set of improper packets initially locating in T_i and having their destinations in T_j, and let $I(T_i) = \cup_{j \neq i} I(T_j, T_i)$. Notice that $I(T_i)$ is the set of all improper packets in T_i. It can easily be shown that using two steps, we can move an improper packet into its destination subtree. In fact, using the standard pipeline technique, we can move $\sum_{j \neq 1} |I(T_j, T_1)|$ to their destinations in T_1 using at most $2|T_1| + \sum_{j \neq 1} 2|I(T_j, T_1)|$ steps while keeping other packets in their initial positions. Applying the same procedure to $T_2, T_3, \cdots, T_d$ in order, we guarantee that all improper packets reach their destinations in at most

$$\sum_{1 \leq i \leq d} 2(|I(T_i)| + |T_i|)$$

steps.

Recall that by the claim, T_i' can be formed in at most $|T_i|$ steps. By induction, all subtrees can be routed (in parallel) in fewer than $9 \max |T_i|$ steps. Thus, π can be routed in less than

$$\max |T_i| + \sum_{1 \leq i \leq d} 2(|I(T_i)| + |T_i|) + 9 \max |T_i| \leq 9n.$$

This completes the induction step. Since any tree with two nodes can be routed in 1 step, Theorem 2.1 is proved. $\Box$

We have given a reasonably good upper bound on the routing number of trees. The bound seems too large. A trivial lower bound of it is $2n-3$. We mention the following open problem.

Problem 1. What is the (many-to-one) routing number of trees?

Since for any spanning subgraph H of G we have $rt(G) \leq rt(H)$, thus $rt(G)$ is bounded above by $rt(T)$ for any spanning subtree T of G. Thus, Theorem 2.1 also gives an upper bound on the routing number of any graph.

Theorem 2.2 *For any n-node graph G, $rt(G) \leq 9n$.*

2.2 Complete graphs

Let K_n be the complete graph on n vertices. Although K_n is highly connected, routing on K_n still takes as long as $n-1$ steps.

Theorem 2.3 *For the complete graph K_n,*

$$rt(K_n) = n - 1.$$

Proof. To see that $rt(K_n) \geq n-1$, it is enough to consider the assignment π that all packets are assigned to the same destination. At each step, only one packet can be moved to the destination. Thus π cannot be routed in less than $n-1$ steps.

Now we prove that any many-to-one assignment π can be routed in at most $n-1$ steps. For this purpose, we use the following greedy algorithm:

```
 When $\pi$ is not trivial do
    Find a maximal one-to-one subassignment $\pi'$ of $\pi$ and route $\pi'$ at most 2 steps.
    $\pi := \pi - \pi'$.
Enddo
```

Obviously the algorithm is correct. Now we prove the algorithm takes at most $n-1$ steps by induction on the number of packets. We consider the following cases.

Case 1. There is only one destination. Then, any maximal one-one subassignment π' contains only one packet, and so we can move it to the destination in only 1 step. Since $\pi - \pi'$ contains $n-1$ packets, by induction, we can route $\pi - \pi'$ in at most $n-2$ steps. Hence, the algorithm routes π in at most $n-1$ steps.

Case 2. There are two destinations u, v. A packet is *stable* if its initial location and destination are same. Obviously, during routing, we do not need to move stable packets. Therefore, without loss of generality, we may assume there are no stable packets in the following discussion. Thus, any maximal one-to-one subassignment π' contains exactly two packets. Further, π' is either a transposition (v, u) or $(v' \to v)(u' \to u)$, where $(v' \to v)$ denotes a packet locating at v' and having destination v. It is not difficult to see we can route π' in only 1 step. By induction, we can route $\pi - \pi'$ in only $n-3$ steps. Thus, our algorithm routes π in at most $n-1$ steps.

3. There are at least three destinations. Since there are no stable packets, any maximal one-to-one subassignment π' of π contains at least three packets. By Theorem 2 in [1], π' can be routed in 2 steps. By induction, $\pi - \pi'$ can be routed in $n-4$ steps. Therefore, π can be routed in at most $n-1$ steps.

Combining all cases, we proves our theorem. $\square$

Any permutation can be routed in at most 4 steps on a complete bipartite graph[1]. Using this result, we are able to show

Theorem 2.4 *For the complete bipartite graph $K_{n,n}$, $rt(K_{n,n}) = 2n - 1$.*

The proof of Theorem 2.4 is similar to Theorem 2.3, and so is omitted.

2.3 Linear Arrays

The linear array is another class of graphs fro which the routing number is known.

Theorem 2.5 *For the n-node linear array L_n,*

$$rt(L_n) = 2n - 3.$$

Proof. Given an n-node linear array $L_n = (V, E)$, let $V(L_n) = \{1, 2, \cdots, n\}$ and $E(L_n) = \{(i, i+1) \mid i = 1, \cdots, n-1\ \}$. To see that $rt(L_n) \geq 2n - 3$, we consider the assignment π that every packet is associated with the destination n. After the packet at $n-1$ is moved to n in one step, a new packet arrives n in every other two steps. Thus, the routing of π cannot be achieved in less than $2n - 3$ steps.

To show that $rt(L_n) \leq 2n - 3$, it is enough to show that any assignment π can be routed in at most $2n - 3$ steps. Given any assignment π on L_n, we route it by adopting the even-odd-transposition sorting algorithm [3, 5]: at odd steps, we compare the packets at the endpoints of 'odd ' edges $(1, 2), (3, 4)$, etc. If we can interchange packets (at most two) at the endpoints of $(i, i+1)$ so as to bring them closer to their destinations, we do it. Similarly, at even steps, we compare the packets at the endpoints of 'even' edges $(2, 3), (4, 5)$, etc., and interchange them if necessary.

Now we analyze the complexity of the algorithm. For a packet p, we denote its initial location by $i(p)$ and its destination by $\pi(p)$. The packet p is said to be *black* if $\pi(p)$ is right to $i(p)$; it is called *white* otherwise. During routing, a black packet never moves leftwards and a white packet never moves rightwards. Consider the rightmost black packet p located initially at $i(p)$. If $i(p)$ is even, p does not move at the first step of the routing algorithm. But p will move rightwards until it arrives its destination. If $i(p)$ is odd, p moves rightwards at the first step, and it keeps to move rightwards until it reaches its destination $\pi(p)$.

We next consider the second rightmost black packet p_2. Let its initial location be $i(p_2)$ and its destination be $\pi(p_2)$. After step 2 (until it reaches its destination), the rightmost black packet p can never block the movement of p_2. Thus, no matter $i(p_2)$ is odd or even, p_2 will move rightwards after step 3 until it reaches its destination $\pi(p_2)$.

In general, it can easily be proved that the jth rightmost black p_j will move rightwards after step $j + 1$. Therefore, p_j reaches its destination after

$$j - 1 + \pi(p_j) - i(p_j) \leq 2n - 3$$

steps.

Similarly, any white packet can also reach its destination in at most $2n - 3$ steps. This finishes the proof. □

2.4 Cartesian products

For graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we define the Cartesian product $G_1 \times G_2$ be the graph with vertex set $V = V_1 \times V_2 = \{(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2\}$ and edge set $E = \{((v_1, v_2), (v_1', v_2')) | v_1 = v_1', (v_2, v_2') \in E_2 || v_2 = v_2', (v_1, v_1') \in E_1\}$. For permutation routing, it is unknown whether $rt(G \times G) \geq rt(G)$ is true or not[1]. However, for many-to-one routing problem, it is always true.

Theorem 2.6 *$rt(G \times G) \geq rt(G)$ for any graph G for any graph G.*

Proof. Since $G \times G$ has $|G|^2$ nodes, $rt(G \times G) \geq |G|^2 - 1$. On the other hand, by Theorem 2.1, $rt(G) \leq 9|G|$. Thus the fact is true for all graphs G with more than 9 nodes. For all graphs with fewer nodes, we can verify the fact directly. □.

3 Relation Routing - A Generalization

Given an n-node graph G, any assignment of n packets on G, can always be factored as a composition of a one-to-many assignment and a many-to-one assignment. Using this fact, we obtain a general result regarding to the routing problem.

Theorem 3.1 *On an n-node graph G, under the matching routing model, any assignment of k packets can be routed in $O(k + n)$ steps.*

Proof. For an assignment A of n packet on G, it can be factored into a composition of a one-to-many assignment A_1 and a a many-to-one assignment A_2. Thus, we can route A by routing A_1 and then A_2 separately. By Theorem 2.2, A_2 can be routed in at most $9n$ steps. Since a one-to-many assignment is a reverse of a many-to-one assignment, A_1 can also be routed in $9n$ steps. Therefore, A can be routed in $O(n)$ steps.

An assignment of $k < n$ packets can be considered as an assignment of n packets by introducing $n - k$ dummy packets. Thus, it can routed in $O(n) = O(k + n)$ steps.

Any assignment A of $k > n$ packets can be factored into $\lfloor k/n \rfloor$ assignments of n packets and an assignment of $k - \lfloor k/n \rfloor n$ packets. Therefore, we can route A by routing these subassignments of at most n packets one by one, which takes at most $k/n \times O(n) = O(k+n)$ steps. □

We can also establish the following generalization of Theorem 2.3 and Theorem 2.5, the proof of which is similar and hence is omitted.

Theorem 3.2 *(1) Any assignment of k packets on a complete graph can be routed in at most $k - 2$ steps.*

(2) Any assignment of k packets on an n-node linear array can be routed in at most $2k + n - 3$ steps.

References

[1] N. Alon, F. R. K. Chung and R. L. Graham. Routing permutations on graphs via matchings. in Proc. of the 25th ACM annual Symposium on Theory of Computing, 1993; SIAM J. Discrete Math. 7(1994), 516-530.

[2] M. Baumslag and F. Annexstein. A unified framework for off-line permutation routing in parallel networks. *Mathematical Systems Theory* 24(1991), 233-251.

[3] N. Haberman. Parallel neighbor-sort (or the glory of the induction principle). *Technical Report AD-759 248, National Technical Information Services, US Department of Commerce, Springfield VA 22151, 1972.*

[4] T. Leighton. Methods for message routing in parallel machines. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, 77-96, 1992.

[5] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes.* Morgan Kaufmann, Cal. 1992.

[6] M. Ramras. Routing permutations on a graph. *Networks* 23(1993), 391-398.

[7] A. Roberts, A. Symvonis and L. Zhang, Routing on Trees via Matchings, *The Workshop on Algorithms and Data Structures*, Kingston, Canada, 1995. Lecture Notes in Computer Sciences vol. 955, 251-263.

[8] L. Zhang, Optimal Bounds for Matching Routing on Trees. *Manuscript.*