Near-Optimal Hot-Potato Routing on Trees

Costas Busch[1], Malik Magdon-Ismail[1], Marios Mavronicolas[2], and Roger Wattenhofer[3]

[1] Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[2] University of Cyprus, Nicosia CY-1678, Cyprus
[3] ETH Zurich, 8092 Zurich, Switzerland

Abstract. In *hot-potato (deflection) routing,* nodes in the network have no buffers for packets in transit, which causes conflicting packets to be deflected away from their destinations. We study *one-to-many* batch routing problems on arbitrary tree topologies. We present two hot-potato routing algorithms, one deterministic and one randomized, whose routing times are asymptotically near-optimal (within poly-logarithmic factors from optimal). Both algorithms are *distributed* and *greedy*; so, routing decisions are made locally, and packets are advanced towards their destinations whenever possible.

1 Introduction

Packet routing is the general task of delivering a set of packets from their sources to their destinations. *Hot-potato* (or *deflection*) routing is relevant in networks whose nodes cannot buffer packets in transit – any packet that arrives at a node must immediately be forwarded to another node at the next time step, as if it were a "hot potato." A *routing algorithm* (or *protocol*) specifies at every time step the actions that each node takes while routing the packets. Hot-potato routing was introduced by Baran [4], and since then, hot-potato routing algorithms have been extensively studied formally and observed to work well in practice for various network architectures [1, 5, 7, 8, 10, 12, 16, 17].

Here, we consider *synchronous tree* networks in which a global clock defines a discrete time. At each time step t, a node may receive packets, which it forwards to adjacent nodes according to the routing algorithm. These packets reach the adjacent nodes at the next time step $t + 1$. At each time step, a node is allowed to send at most one packet per link.[†] Bufferless routing in tree networks has generated considerable interest in the literature: most existing work is for the matching routing model, [2, 14, 18] or the direct routing model, [3, 9]; hot-potato routing of permutations on trees has been studied in [15].

We consider *one-to-many* batch routing problems on trees with n nodes, where each node is the source of at most one packet; however, each node may be the destination of multiple packets. The *routing time* of a routing-algorithm is

[†] At any time step, at most two packets can traverse an edge in the tree, one packet along each direction of the edge.

the time for the last packet to reach its destination. Denote by rt^* the minimum possible routing time for a given routing problem. For a given routing problem, and a set of paths to be followed from the sources to destinations, the *dilation* D is the maximum length of a path, and the *congestion* C is the maximum number of paths that use any link (in either direction). Since at most one packet can traverse an edge in a given direction at each time step, the routing time for the specified paths is $\Omega(C + D)$. On a tree, any set of paths must contain the shortest paths, from the sources to the destinations. Let C^* and D^* be the congestion and dilation for the shortest paths respectively. We immediately get that $rt^* = \Omega(C^* + D^*)$. For *store-and-forward* routing, in which nodes have buffers for storing packets in transit, there are routing algorithms whose performance on trees is close to rt^* [11, 13]. However, such algorithms are not applicable when buffers are not available.

We consider *greedy* hot potato routing. A routing algorithm is greedy if a packet always follows a link toward its destination whenever this is possible. In hot-potato routing, a problem occurs if two or more packets appear at the same node at the same time, and all these packets wish to follow the same link at the next time step. This constitutes a *conflict* between the packets because only one of them can follow that particular link. Since nodes have no buffers, the other packets will have to follow different links that lead them further from their destination. We say that these packets are *deflected*. In a greedy algorithm, a packet π can be deflected only when another packet makes progress along the link that π wished to follow.

Our Contributions. We present two hot-potato routing algorithms on trees with near optimal routing time. Our algorithms are *local,* and thus *distributed*: at every time step, each node makes routing decisions locally based only on the packets it receives at that particular time step. In our algorithms, every source node determines the time at which its packet will be injected. From then on, the packet is routed greedily to its destination. We assume that each source node knows the tree topology, as well as C^* and D^* for the batch routing problem; we emphasize, however, that it need not know the specific sources and destinations of the other packets. The assumption that C^* and D^* are known is common to many distributed routing algorithms [7, 13]. Our two algorithms are sumarized below:

i. Algorithm Deterministic has routing time $O((\delta \cdot C^* + D^*) \lg n) = O(\delta \cdot rt^* \cdot \lg n)$, where δ is the maximum node degree in the tree. All routing choices are deterministic.
ii. Agorithm Randomized has routing time less than $O((C^* + D^*) \lg^2 n) = O(rt^* \cdot \lg^2 n)$ with probability at least $1 - \frac{1}{n}$. Randomization is used when packets select priorities. These priorities are then used to resolve conflicts.

For bounded-degree trees, algorithm Deterministic is within a logarithmic factor of optimal. Algorithm Randomized is only an additional logarithmic factor away from optimal for arbitrary tree topologies.

The general idea of our algorithms is to divide packets into levels based on the position of their source in the tree. Packets at different levels are routed in different phases. We show that there are at most $O(\lg n)$ such phases. In algorithm Deterministic, each phase has a duration $O(\delta \cdot C^* + D^*)$, while in algorithm Randomized, each phase has duration $O((C^* + D^*) \lg n))$. Combining these bounds with the bound on the number of phases leads to our routing time bounds. The heart of both of our algorithms lies in the use of *canonical* deflections, in which packets are only deflected onto edges used by other packets that moved forward in the previous time step.

Paper Outline. We first introduce trees in Section 2, and packet behavior in hot-potato routing in Section 3. We then present our deterministic and randomized routing algorithms in respective Sections 4 and 5.

2 Trees

A tree $T = (V, E)$ is a connected acyclic graph with $|V| = n$ and $|E| = n - 1$. The *degree* of node v is the number of nodes adjacent to v. Let $v \in V$; then, T induces a subgraph on $V - \{v\}$ which consists of a number (possibly zero) of connected components. Each such connected component is a *subtree of v in T*.[‡] If v is adjacent to K nodes in T, then there are k disjoint subtrees $T_1, \ldots, T_k$ of v, one for each node $v_i \in T_i$ that is adjacent to v. The *distance* from v to u, is the number of edges in the (unique) shortest path from v to u.

The main idea behind our algorithms is to look at the tree from the point of view of a short node. A node v in the tree is *short* if every subtree of v contains at most $n/2$ nodes. At least one short node is guaranteed to exist, and by starting at an arbitrary node and moving along a path of largest subtrees (if the size of the subtree is greater than $n/2$), a short node can be found in in $O(n)$ time. A tree T may have many short-nodes, however, a deterministic algorithm always starting at a particular node will always return a particular short-node. So, from now on, we will assume that the short-node of a tree is uniquely determined.

We now define (inductively) the *level* ℓ of a node, and the *inner-trees* of T as follows. The tree T is the only inner-tree at level $\ell = 0$. The only node at level $\ell = 0$ is the short node of T. Assume we have defined inner-trees up to level $\ell \geq 0$. Every connected component obtained from the inner-trees of level ℓ by removing the short nodes of these inner-trees at level ℓ is an inner-tree at level $\ell + 1$. The level $\ell + 1$ nodes are precisely the short nodes of the inner-trees at level $\ell + 1$.

It is clear that the above definition inductively defines the inner-trees at all levels; it correspondingly assigns a level to every node. We can easily construct an $O(n^2)$ procedure to determine the node levels and inner-trees of T at every level. Further, the following properties (which we state here without proof) hold:

[‡] Note that for unrooted trees which we consider here, a subtree of a node v originates from every adjacent node of v; in contrast, the convention for rooted trees is that a subtree of v is any tree rooted at a child of v.

(i) every inner-tree is a tree, (ii) the maximum level of any node and inner-tree is no more than $\lg n$, (iii) an inner-tree T' at level ℓ contains a unique node x at level ℓ, which is the short node of the inner-tree (we say that x is the inducing node of T'), (iv) any two inner-trees at the same level are disconnected, and (v) all nodes in a level ℓ inner-tree other than the inducing node have a level that is smaller than ℓ.

3 Packets

Packet Paths. A *path* is any sequence of nodes $(v_1, v_2, \ldots, v_k)$. The length of the path is the number of edges in the path. After a packet has been routed from its source to its destination, it has followed some path. We define the *original path* of a packet π as the shortest path from the source node of the packet to its destination node. This will be the path that would be greedily followed if the packet experiences no deflections.

Let ℓ be the smallest level of any node in the original path of π. Then, there is a unique node v with level ℓ in the path of π (since otherwise inner-trees of the same level would not be disconnected). Let T' be the inner-tree that v is inducing. The whole original path of π must be a subgraph of T' (from the definition of inner-trees). We say that the level of packet π is ℓ, and that the inner-tree of π is T'.

After injection, the *current path* of packet π, at any time step t, is the shortest path from its current node to its destination node. At the moment when the packet is injected, its current path is its original path. While packet π is being routed to its destination, it may deviate from its original path due to deflections. However, the packet traverses each edge of its original path at least once before reaching its destination.

A packet moves *forward* if it follows the next link of its current path; otherwise, the packet is deflected. When a packet moves forward, its current path gets shorter by removing the edge that the packet follows. If a packet is deflected, its current path grows by the edge on which the packet was deflected, and its new current path is the shortest path from its current node to its destination node.

Packet Routing and Deflections. In our algorithms, a packet remains in its source node until a particular time step at which the packet becomes *active*. When the packet becomes active, it is injected at the first available time step on which the first link of its original path is not used by any other packets that reside at its source node. We call such an injection a *canonical injection*.

At each time step, each node in the network does the following: (i) the node receives packets from adjacent nodes, (ii) the node makes routing decisions, and (iii) according to these decisions, the node sends packets to adjacent nodes. We say that two or more packets *meet* if they appear in the same node at the same time step; they *conflict* if they also wish to follow the same link forward. In a conflict, one of the packets will successfully follow the link, while the other packets must be deflected. Our algorithms are greedy: a packet always attempts move forward unless it is deflected by another packet with which it conflicts.

In our algorithms, packets are deflected in a particular fashion so as to ensure that the congestion of any edge (with respect to the set of current paths) never increases. Consider a node v at time t. Let S_f denote the set of packets which moved forward during the previous time step and now appear in v. Let E_f be the set of edges that the packets in S_f followed during the previous time step. Let π be a packet in node v that is deflected at time t. Node v first attempts to deflect π along an edge in E_f. Only if all the edges in E_f are being used by packets moving forward, or by other deflected packets, then some other edge adjacent to v is used to deflect π. We call this process of deflecting packets *canonical deflection*. The deflection is *safe* if it was along an edge in E_f. One can show that if injections are canonical, then all deflections are safe. Safe deflections simply "recycle" edges from one path to another path, and thus cannot increase an edge's congestion.

4 A Deterministic Algorithm

Here we present the algorithm Deterministic in pseudo-code format.

Algorithm: Deterministic

Input: Tree T with max. node degree δ; A set of packets Π with shortest path congestion C^* and dilation D^*; each node knows T, C^*, D^*;

for *every packet* π *at level* ℓ **do**

π gets active at time $\tau \cdot \ell$, where $\tau = 2(\delta \cdot C^* - 1) + D^*$;

The injection and deflections of π are canonical;

π moves greedily to its destination;

end

Theorem 1. *The routing time of algorithm* Deterministic *is* $O((\delta \cdot C^* + D^*) \lg n)$.

We proceed by sketching the proof of Theorem 1. Let m be the maximum level in T. Since a level $l+1$ inner-tree has fewer than half the nodes of the level l inner-tree that gave rise to it, it is easy to see that $m \leq \lg n$. We divide time into consecutive phases $\phi_0, \phi_1, \ldots, \phi_m$, such that each phase consists of τ time steps. Denote the level i packets by Π_i, $0 \leq i \leq m$. The packets in Π_i become active at the first time step of phase ϕ_i. We will show that all packets of level i are absorbed during phase ϕ_i. In particular, we will show that the following invariants hold, where $i \geq 0$:

P_i: all packets of $\Pi_0 \cup \Pi_1 \cup \cdots \cup \Pi_i$ are absorbed by the end of phase ϕ_i.

Set P_{-1} to true. It suffices to show that the following statement holds, for $i \geq 0$:

Q_i: if P_{i-1} holds, then all packets in Π_i are absorbed during phase ϕ_i.

Consider a particular level $\ell \geq 0$ and phase ϕ_ℓ. Suppose that $P_{\ell-1}$ holds. In phase ϕ_ℓ the only packets in the network are those of Π_ℓ. During phase ϕ_ℓ, let π be the first packet to leave its inner-tree going from node u to node v. Since deflections are safe, this means that some packet π' moved from v to u in the previous time step. Since the entire original path of π' was contained in some level ℓ inner-tree, this means that π' left its inner-tree before π did, a contradiction. Thus, we have:

Lemma 1. *During phase ϕ_ℓ, each packet of Π_ℓ remains inside its inner-tree.*

Since only packets of the same inner-tree may conflict with each other, we only need to show that every level ℓ packet with a particular inner-tree T' is absorbed during phase ϕ_ℓ.

We adapt a technique developed by Borodin *et al.* [6, Section 2], called a "general charging scheme", based upon deflection sequences. Their result implies that for greedy routing on trees, whenever a packet is deflected, some other packet makes it to its destination. Thus if the number of packets is k, then a packet can be deflected at most $k-1$ times, giving:

Corollary 1 ([6]). *Each packet is absorbed in at most $D^* + 2(k-1)$ time steps after injection, where k is the number of packets with inner-tree T'.*

All packets with inner-tree T' use the inducing short node r. Since the degree of r is at most δ, and the congestion on an edge never increases, $k \leq \delta \cdot C^*$. Subsequently, all packets of inner-tree T' are absorbed in at most $2(\delta \cdot C^* - 1) + D^* \leq \tau$ time steps, i.e., by the end of phase ϕ_ℓ. Since there are at most $O(\lg n)$ phases, the theorem is proved.

5 A Randomized Algorithm

Algorithm Randomized is similar to algorithm Deterministic, except that packets now have priorities: low or high. High priority packets have precedence (to move forward) over low priority packets in a conflict. Conflicts between equal priority packets are arbitrarily resolved canonically.

Algorithm: Randomized

Input: A tree T; A set of packets Π with shortest path congestion C^* and dilation D^*; each node knows T, C^*, D^*;

for *every packet π at level ℓ* **do**

π gets active at time step $\tau \cdot \ell$, where $\tau = 16 \cdot (C^* + D^*) \cdot (2 \lg n + \lg\lg 2n) + 3D^* + 1$;

The injection and deflections of π are canonical, and initially π has low priority;

π moves greedily to its destination;

When packet π becomes active it has low priority;

Let $p = \frac{1}{4(C^* + D^*)}$; if π is deflected, on the next time step, its priority becomes high with probability p, and low with probability $1-p$, independent of its previous priority; π keeps its new priority until the next deflection;

end

Theorem 2. *With probability at least $1 - 1/n$, the routing time of algorithm* Randomized *is at most $\kappa(C^* + D^*) \lg^2 n$, for some constant $\kappa \approx 33$.*

We very briefly sketch the proof of Theorem 2. We define P_i and Q_i as in Section 4, and we show that P_i holds with high probability by showing that Q_i holds with high probability. Once again we focus on the level ℓ packets of a particular inner-tree T', and suppose that all packets of the previous phases are absorbed. We will show that every packet with inner-tree T' will be absorbed in phase ϕ_ℓ, with high probability. Let $T_1, T_2, \ldots, T_w$ denote the subtrees of inducing node r in T'. We summarize some useful properties of these subtrees (without proof).

Lemma 2. *At any time step during phase ϕ_ℓ, at most C^* packets are in any subtree T_j.*

Using Lemma 2, since at most one new packet can enter a subtree in one time step, we obtain:

Lemma 3. *During any time period of length x time steps, at most $C^* + x$ different packets have appeared in any subtree T_j.*

Corollary 2. *If packet π is not deflected for a period of x time steps, then it may have conflicted at most once with at most $2C^* + x$ different packets.*

The *depth* of node v (or packet currently at node v), is its distance from r.

Lemma 4. *At any time during phase ϕ_ℓ, the depth of a packet is at most D^*.*

Consider some packet π. By Lemma 4, the current path length of a packet is always at most $2D^*$, so if it is not deflected for $2D^*$ time steps, then it reaches its destination. Suppose that π has high priority. It can only be deflected by another high priority packet σ. Each such packet σ has at most one chance to deflect it, with probability at most p, since σ is in high priority. If π is not deflected for $2D^*$ time steps, then at most $2C^* + 2D^*$ packets had a chance to deflect it (Corollary 2). Thus the probability of deflection is at most $2(C^* + D^*)p = \frac{1}{2}$, giving the main lemma:

Lemma 5. *A high priority packet reaches its destination without deflections with probability at least $1/2$.*

Thus, each time a packet is deflected, it has probability at least $\frac{1}{2}p = \frac{1}{8(C^*+D^*)}$ to reach its destination. If a packet has not reached its destination after τ steps, then it has been deflected at least $\frac{1}{2}(x - D^*)$ times, over the interval of times $[t_1, t_x]$ where $x = \tau - 2D^* - 1$. The probability of not reaching its destination after so many deflections is at most $(1 - \frac{1}{2}p)^{\frac{1}{2}(x-D^*)}$, which after substituting the expression for τ yields:

Lemma 6. *Packet π is absorbed in phase ϕ_ℓ with probability at least $1 - \frac{1}{n^2 \lg 2n}$.*

Since there are at most n packets in a phase, and at most $\lg n + 1$ phases, we use a union bound to obtain a lower bound on the probability that every phase is succesful, giving Theorem 2.

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. N. Alon, F.R.K. Chung, and R.L.Graham. Routing permutations on graphs via matching. *SIAM Journal on Discrete Mathematics*, 7(3):513–530, 1994.
3. Stephen Alstrup, Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Direct routing on trees. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 98)*, pages 342–349, 1998.
4. P. Baran. On distributed communications networks. *IEEE Transactions on Communications*, pages 1–9, 1964.
5. Constantinos Bartzis, Ioannis Caragiannis, Christos Kaklamanis, and Ioannis Vergados. Experimental evaluation of hot-potato routing algorithms on 2-dimensional processor arrays. In *EUROPAR: Parallel Processing, 6th International EUROPAR Conference*, pages 877–881. LNCS, 2000.
6. 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.
7. C. Busch. $\tilde{\mathrm{O}}$(Congestion + Dilation) hot-potato routing on leveled networks. In *Proceedings of the Fourteenth ACM Symposium on Parallel Algorithms and Architectures*, pages 20–29, August 2002.
8. 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, January 2000.
9. Costas Busch, Malik Magdon-Ismail, Marios Mavranicolas, and Paul Spirakis. Direct routing: Algorithms and Complexity. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA)*, September 2004.
10. W. D. Hillis. *The Connection Machine*. MIT press, 1985.
11. T. Leighton, B. Maggs, and A. W. Richa. Fast algorithms for finding O(congestion + dilation) packet routing schedules. *Combinatorica*, 19:375–401, 1999.
12. N. F. Maxemchuk. Comparison of deflection and store and forward techniuques in the Manhattan street and shuffle exchange networks. In *Proc. IEEE INFOCOM*, pages 800–809, 1989.
13. Friedhelm Meyer auf der Heide and Berthold Vöcking. Shortest-path routing in arbitrary networks. *Journal of Algorithms*, 31(1):105–131, April 1999.
14. Grammati E. Pantziou, Alan Roberts, and Antonios Symvonis. Many-to-many routing on trees via matchings. *Theoretical Comp. Science*, 185(2):347–377, 1997.
15. Alan Roberts, Antonios Symvonis, and David R. Wood. Lower bounds for hot-potato permutation routing on trees. In *Proc. 7th Int. Coll. Structural Information and Communication Complexity, SIROCCO*, pages 281–295, June 2000.
16. C. L. Seitz. The Caltech Mosaic C: An experimental, fine-grain multicomputer. In *Proc. 4th Symp. on Parallel Algorithms and Architectures*, June 1992. Keynote Speech.
17. 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.
18. L. Zhang. Optimal bounds for matching routing on trees. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 445–453, 1997.