Comment router dans un arbre ?

Pierre Fraigniaud[1] and Cyril Gavoille[2]

[1] *CNRS, Laboratoire de Recherche en Informatique, Université Paris-Sud, 91405 Orsay cedex, France.* pierre@lri.fr. *Part of this work was done while the first author was visiting Carleton University at Ottawa, and Concordia University at Montréal. Additional support from Carleton U, the NATO, and the RNRT project R.O.M.* [2] *LaBRI, Université Bordeaux 1, 351, cours de la Libération, 33405 Talence, France.* gavoille@labri.fr

---

Dans cet article nous montrons comment construire une structure de données distribuées permettant à un message de circuler dans un arbre à n sommets le long de plus courts chemins. Plus précisement la structure de données que nous proposons occupe un espace de $O(\log n)$ bits par sommet et les en-têtes des messages (qui correspondent aux adresses de destination) sont codés sur $O(\log n)$ bits. De plus le temps de routage est constant en chaque sommet de l'arbre et la structure de données peut être établie en temps $O(n \log n)$ pour l'ensemble des sommets. Ce résultat doit être rapproché de la borne inférieure qui établit que si les en-têtes sont codés sur seulement $\log n$ bits (c'est-à-dire sont des entiers uniques pris entre 1 et n), alors la taille de la structure de données locale pour permettre un routage de plus courts chemins doit être $\Omega(\sqrt{n})$ bits pour certains arbres. Notre résultat démontre qu'une petite variation sur la longueur des en-têtes, par exemple un facteur additif de $\log n$ bits, a un impact considérable sur l'espace mémoire local des sommets.

L'ensemble des preuves des résultats annoncés dans cet extrait est disponible dans la version longue de l'article [FG01a]. Un résumé étendu apparaît aussi dans [FG01b].

Keywords: routage compact, arbres, algorithmes de routage

---

1 The Routing Problem

Delivering messages between pairs of processors is a basic and primary activity of any distributed communication network. This task is performed using a *routing scheme*, that is a mechanism working in a distributed fashion, and allowing source-to-destination messages delivery in the network, for any source and any destination.

Quite often, the design and the optimization of the management and control systems of the network, including routing, is done after the network construction. In this case, the routing problem is stated as follows: given a graph (i.e., the underlying topology of a communication network), design for each node of the graph (i.e., for each router of the network) a efficient computable function that determines, for every message entering a node, the link on which the message has to be forwarded. This computation is performed as function of the information contained in the header of the message, e.g., the destination node address, and of local information stored at the node. The term "efficient" groups a set of desirable quality factors like: routes of reasonable length (compared to shortest paths); low local computation time; limited link-congestion; compact local data structures; fault tolerance; etc.

We want to point out that the complexity results on routing are quite strongly dependent of the model, including, e.g., the ability to setup the addresses of the nodes, the ability to modify the headers, etc. Moreover, we also want to stress that results strongly differ according to slight variations of parameters, including, e.g., the size of the header, the length of the routes, etc. Therefore, let us make clear the framework of this paper.

In the remaining, the communication network is denoted by a connected and undirected n-node graph $G = (V, E)$. The routing scheme makes use of *addresses*, every node being identified by an address which is unique in the network. Hereafter, the address of node u is denoted by $\ell(u)$. The routing scheme makes also use of a *routing algorithm* that is a distributed algorithm whose role is to route messages. Upon reception of a message by node u, the routing decision taken at u is performed by a *routing function*. It depends solely on (1) the message header, (2) the incoming link, and (3) local data structures stored at u. The design of the routing scheme includes not only de design of the routing algorithm via the definition of the routing function, but also the setting of the address-set, the header-set, and of the local data-structures. Moreover, it has to incorporate the link labeling.

More precisely, incoming and outgoing links from every node u are identified by numbers from 1 to $\deg(u)$, the degree of u. This labeling is local, and should not necessarily be consistent between neighbors. In particular, the edge $\{u,v\}$ can be labeled i by u, and j by v, with $i \neq j$. Each node (i.e., each router) is also supposed to have an extra link, connecting it to its local host. This link is labeled 0. Once the routing function has selected the outgoing link on which the message must be forwarded, i.e., a number between 0 and $\deg(u)$, it creates a new header for that message, and it sends the message with its header through that link. The original header of a message is given by the source host; it simply consists of the address of the destination, and, upon reception of a message from the source host, the attached router may modify the header, but the result of this modification depends solely on the destination address, and, again, local data structures stored by the router. We point out here that no time or space consuming computations are hidden in this model.

A specific type of routing is called *direct*. Direct routing imposes that the message header is fixed at once by the source host, and cannot be modified by intermediate nodes on its route toward the destination. In other words, the header set is exactly the node address set. Direct routing schemes are desirable because they do not require extra hardware facilities to rewrite headers. Note though that relaxing this constraint allows to construct sophisticate routing schemes for arbitrary graphs (cf., e.g., the hierarchical routing schemes introduced in [AP90, PU89]).

To summarize, solving the routing problem consists to design the whole routing scheme, including (1) the routing function, (2) the format of the addresses, (3) the local labeling of the links, (4) the format of the headers, and (5) the local information to be stored locally at each node in order to perform the computation of the routing function. The complexity of the setting-up of these data is measured by the time it takes to perform those computations from given the graph G, and is refered as the *pre-processing time*. The other parameters that are typically addressed in the literature are the following. The *size of the addresses* (that is also the *size of the headers* for direct routing schemes) is measured by the range of the set in which these addresses are taken. More precisely, for $m \geq n$, a routing scheme is of $[1,m]$-range address if the addresses are taken from the set $\{1,2,\ldots,m\}$. (Alternatively, such a scheme is called a $\log m$-bit address routing scheme.) The *local memory space* is the size of all the data structures required by each node to compute the routing function. The *routing time* is the worst-case time-complexity of the routing function to perform its computation. The *length of the routes* is measured by competition against the shortest paths.

In this paper we consider shortest-path routing only. Moreover, all the routing schemes we will construct are direct, i.e., routing will be performed based on the destination address only, and headers are not modified once set by the source. Regarding the local labeling of the links, we will consider two main variations of the problem: the *designer-port* model, and the *fixed-port* model. The former allows the designer of the routing scheme to freely label (between 1 and $\deg(u)$) the incident links of each node u while constructing and optimizing the data structures of the routing scheme. On the contrary, the latter assumes that the local labeling of the links is given is advance. The fixed-port model forces the use of correspondence tables if the routing function needs a specific labeling to perform efficiently, and therefore this model may lead to a significant increase of the size of the data structures compared to the designer-port model. On the positive side though, the fixed-port model allows to superpose several routing schemes such as, e.g., hierarchical routing schemes [AP90, PU89].

2 Related Works

One of the most widely used routing scheme for arbitrary graphs is the *routing tables*. It consists to label the nodes from 1 to n, and to store an n-entry table at each node, such that its i-th entry returns the local number of the adjacent link on which a message of destination address i has to be transmitted. Routing tables are direct and use $[1,n]$-range addresses, require $O(n\log d)$ bits of memory at each node of degree d in the fixed-port model, and perform routing in $O(1)$ time at each node. Although $O(n\log d)$ memory bits seems quite high, it must be pointed out that this is the best that can be achieved for arbitrary graph. More precisely, it has been proved in [GP96] that $\Omega(n \log d)$ bits is a lower bound for shortest-path routing schemes using addresses on $c \log n$ bits, for every constant $c \geq 1$. This lower bound holds in the designer-port model. According to this result, a tremendous effort has been done to derive routing schemes requiring smaller memory space for specific classes of networks, in particular for trees.

In trees, a common strategy consists to set the addresses via a Depth-First Search (DFS) traversal of the tree, and then to use the *interval routing scheme* (IRS) of [SK85, vLT87] (see [Gav00] for a survey on IRS). More precisely, at every internal node u of the tree, the edge leading to a child v is associated to an interval containing the set of the addresses of the nodes in the subtree rooted at v, and the link leading to the root is associated to the complementary interval (this latter interval may wrap modulo n). The memory space is $O(d \log n)$ bits at each node of degree d in the fixed-port model, and the use of a binary search in the set of intervals provides a $O(\log n)$ routing time at each node. The total amount of routing informations, summed over all nodes, is low compared to the routing tables, i.e., $O(n \log n)$ bits in total. However, IRS does not evenly balance routing information among the nodes since $O(n \log n)$ bits could be required locally for trees of large degree.

Cowen [Cow99] proposed a direct routing scheme in the fixed-port model for trees of large degree using $O(\sqrt{n} \log n)$ bits of memory, $3 \log n$ bits of addresses, and $O(\log n)$ routing time. The address-space can be optimized considering the designer-port model: the direct routing scheme in [EGP98] uses $O(\sqrt{n} \log n)$ bits of memory with $[1,n]$-range addresses, and $O(\log n)$ routing time. Actually, if we insist on small memory space, [Gav00] shows that there exists a routing scheme for trees with $[1,n]$-range addresses, $\alpha\sqrt{n}$ bits of memory, where $\alpha \approx 3.7$. However, this latter scheme may involve exponential routing times.

Peleg [Pel99] proposed the best known routing scheme for trees, inspired from the so-called "distance labeling" schemes, and stated in the fixed-port model. It uses $O(\log^2 n)$ bits for both the addresses and the memory at each node, and requires $O(\log n)$ routing time. Actually, the routing algorithm is direct and based on the destination-address only so that the node needs only to store its address. That type of routing schemes are called "memory-free" routing or "label-based" routing. As it is remarked in [Pel99], every direct routing scheme can be transformed into a memory-free routing scheme by extending the size of each address so that it contains all the local data structures of the node.

This paper improves all these previous routing schemes for trees, including the one in [Pel99].

3 Our Contributions

1. We propose a direct shortest-path routing scheme for trees using $5 \log n$ bits for the addresses, $3 \log n$ bits of memory space, and $O(1)$ routing time[†]. This scheme is asymptotically optimal for all of the considered measures: memory space, header size, and routing time. Reducing the address range can be done, but up to an increase of the routing time. More precisely, we also propose a direct shortest-path routing scheme using $2.8 \log n$ bits for the addresses, $3 \log n$ bits of memory space, but involving polynomial time routing decisions.

Attaching the address with the local data structure at each node, we thence show that there exists (1) a shortest-path memory-free routing scheme for trees with constant routing time and $7 \log n$ bits of addresses, and (2) a shortest-path memory-free routing scheme for trees with a $4.8 \log n$ bits of addresses and polynomial routing time. All these results are stated in the designer-port model.

2. For the fixed-port model, we propose a direct shortest-path routing scheme using $O(\log^2 n / \log\log n)$ bits for the addresses, $O(\log^2 n / \log\log n)$ bits of memory space, and with constant routing time. We obtain a shortest-path memory-free routing scheme using $O(\log^2 n / \log\log n)$ bits with constant routing time. These results improve the best known result by a factor $O(\log\log n)$.

3. We also show lower bounds. In particular, we show that, in the designer-port, model any shortest-path routing scheme for arbitrary tree requires a memory space of at least $\alpha\sqrt{n} - o(\sqrt{n})$ bits if using $[1, n + o(\sqrt{n}/\log n)]$-range addresses. This improves and generalizes the lower bound of [EGP98], and

[†] All the time complexities are computed in a word-RAM model where standard integer operations on words of $O(\log n)$ bits are performed in constant time, including logical bitwise operations, integer division, and integer multiplication. All the logarithms are in base 2.

shows the optimality of the results therein. By comparing this lower bound with our first contribution, it also shows that a little variation of the address size, e.g., by an additive $O(\log n)$ bits factor, has a significant impact on the local memory space.

For the fixed-port model, we prove a $n - o(n)$ bits lower bound on the memory space for shortest-path routing scheme on trees using $[1, n + o(n/\log n)]$-range addresses. In contrast with this bound, we construct a direct shortest-path routing scheme with $[1, n]$-range addresses, using at most $n + o(n)$ bits of memory space, and constant routing time. This demonstrates an optimal tradeoff between address-size and memory space.

References

[AP90] Baruch Awerbuch and David Peleg. Sparse partitions. In 31^{th} *Symposium on Foundations of Computer Science (FOCS)*, pages 503–513. IEEE Computer Society Press, 1990.

[Cow99] Lenore J. Cowen. Compact routing with minimum stretch. In 10^{th} *Symposium on Discrete Algorithms (SODA)*, pages 255–260. ACM-SIAM, 1999.

[EGP98] Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. In 17^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 11–20. ACM PRESS, August 1998.

[FG01a] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. Research Report RR-1252-01, LaBRI, University of Bordeaux, 351, cours de la Libération, 33405 Talence Cedex, France, January 2001. Voir http://dept-info.labri.fr/~gavoille.

[FG01b] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume Lecture Notes in Computer Science. Springer, July 2001. To appear.

[Gav00] Cyril Gavoille. A survey on interval routing. *Theoretical Computer Science*, 245(2):217–253, 2000.

[GP96] Cyril Gavoille and Stéphane Pérennès. Memory requirement for routing in distributed networks. In 15^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 125–133. ACM PRESS, May 1996.

[Pel99] David Peleg. Proximity-preserving labeling schemes and their applications. In Peter Widmayer, Gabriele Neyer, and Stephan Eidenbenz, editors, 25^{th} *International Workshop, Graph - Theoretic Concepts in Computer Science (WG)*, volume 1665 of Lecture Notes in Computer Science, pages 30–41. Springer, June 1999.

[PU89] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, July 1989.

[SK85] Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, February 1985.

[vLT87] Jan van Leeuwen and Richard B. Tan. Interval routing. *The Computer Journal*, 30(4):298–307, 1987.