An Underlay Strategy for Indirect Routing

Aline Carneiro Viana* (aline.viana@lip6.fr)
Laboratoire LIP6/CNRS – Université Pierre et Marie Curie
8, rue du Capitaine Scott – 75015 – Paris – France
Phone: +33 1 44.27.71.26
Fax: +33 1 44.27.74.95

Marcelo Dias de Amorim (marcelo.amorim@lip6.fr)
Laboratoire LIP6/CNRS – Université Pierre et Marie Curie
8, rue du Capitaine Scott – 75015 – Paris – France
Phone: +33 1 44.27.87.72
Fax: +33 1 44.27.74.95

Serge Fdida (serge.fdida@lip6.fr)
Laboratoire LIP6/CNRS – Université Pierre et Marie Curie
8, rue du Capitaine Scott – 75015 – Paris – France
Phone: +33 1 44.27.30.58
Fax: +33 1 44.27.53.53

José Ferreira de Rezende (rezende@gta.ufrj.br)
GTA/COPPE/Poli – Universidade Federal do Rio de Janeiro
P.O. Box 68504 – 21945-970 – Rio de Janeiro – RJ – Brasil
Phone: +55 21 2562-8645
Fax: +55 21 2290-6626

Abstract. The evolution of the Internet toward ubiquity, mobility, and independence of wired infrastructure requires revising routing in large dynamic clouds. The need for frequent address updates caused by node mobility suggests decoupling the permanent node identifier from its topological address. This paper proposes Tribe, an indirect and scalable routing protocol for self-organizing networks. Tribe provides an anchor-based abstraction, where the communication is split into two phases: location of the destination node and direct communication between source and destination, associated with appropriate addressing schemes. Tribe anchor nodes play the role of rendezvous points and are responsible for translating a node's identifier into a topology-dependent address. Tribe achieves high scalability by distributing location information among all nodes in the network using peer-to-peer concepts. By managing regions of a logical addressing space, Tribe nodes route in a hop-by-hop basis with small amount of information and communication cost. A qualitative analysis of the Tribe topology and a performance evaluation of the protocol behavior are provided. Tribe raises fundamental issues and triggers a high potential for future work.

Keywords: Indirect routing, peer-to-peer communication, self-organization.

* Contact author.

© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

1. Introduction

In the wide spectrum of networking technologies, connectivity is still the main service provided by networks. New scenarios (e.g. ad-hoc networks, sensor networks, and overlays), in addition to new routing requirements (multicast, QoS, mobility, scalability) have pushed the routing principles closer to their limits. We believe that it is time to re-assess the relationship between the various components of a routing protocol in order to appropriately address above mentioned challenges, recognizing that today's improvements incorporated in the architecture will suffer to survive the network evolution.

In this paper, we propose Tribe, a protocol that is based on innovative principles and raises a number of design issues. Tribe questions the existence of an addressing structure and organization, and a mathematical space that ease routing in a self-organized network.

In fixed networks, routing information is embedded into the topological dependent node address, which also uniquely identifies the node in the network. In self-organizing networks, however, nodes are supposed to spontaneously join/leave, which invalidates routing information. In such a scenario, the permanent node identifier cannot include dynamic location information. The need for frequent network addressing updates suggests the proposition of a network infrastructure where a permanent node identifier is independent of a topological-dependent address.

Similar papers have been published lately on this issue but Tribe is fundamentally different as it raises many issues. How many addresses per node should we use, and of what type, for which purpose? What is the impact of the addressing organization on the topological space of the network? How can we design a routing transfer function that is robust to mobility with minimum effect on the addressing scheme? How can we enforce routing continuity when a node moves or leaves?

Recent location architectures like CAN [15], Chord [19], Pastry [16], and *i*3 [18] employ a similar concept of indirect routing. In these architectures, nodes are organized in an overlay network which depends on a global connectivity ensured by a network-level routing protocol. Tribe, instead, is not an overlay network. Tribe is a network-level protocol and intends to provide full connectivity among nodes.

Tribe distributes location information throughout the topology and identifies anchor nodes that are responsible for storing such information. Any node in Tribe may play the role of a Tribe anchor node – a "rendezvous" point – and be responsible for some nodes' location information. The applied information distribution mechanism also yields scalability and more dynamics in the network.

A small amount of information suffices to implement the Tribe routing protocol. Each node stores only information about itself, its immediate neighbors, and the nodes under its responsibility. Additionally, routing is performed in a hop-by-hop basis. During the routing procedure, each node forwards the messages to the immediate neighbors that get the messages as close as possible to the destination. Tribe limits to $O(k)$ the number of routing table entries (k is the number of immediate neighbors of a node) and to $O(1)$ the routing communication cost.

Tribe was first introduced in [20]. This paper differs from the previous one as it formalizes the problem, develops the region management and the design improvement issues, and provides a qualitative and quantitative analysis of the network.

The remainder of this paper is structured as follows. Section 2 discusses and defines the concept of indirection used in the Tribe routing protocol. Section 3 presents the aspects of the Tribe design. The routing procedure and its performance evaluation are addressed in Section 4. Section 5 describes the Tribe maintenance when a node decides to leave the network. Design techniques for improving Tribe robustness are presented in Section 6. The Tribe simulator and experimental results are presented in Section 7. Section 8 describes related works. Finally, we summarize our contributions and outline subject of future work in Section 9.

2. Tribe overview

This section presents a general view of the architecture defined by Tribe.

2.1. Properties

Tribe takes advantage of some characteristics of completely decentralized systems and self-organizing networks. We present some of these features below.

Distributed control — A number of recent models for wireless and mobile networks, including interactive smart devices [14], peer-to-peer applications [8, 9], and pervasive systems in environments without infrastructure [17, 21] are inherently decentralized. An intrinsic characteristic of decentralized systems is that the distribution of information and responsibilities should be scalable and fault tolerant. Tribe uses distributed hash tables (DHTs) [15, 16, 19, 24]. With DHTs, routing information is completely distributed throughout the topology and the network is able to achieve scalability and more dynamics.

Physical-logical association — Tribe creates a topology that is a logical representation of the network and reflects the relative location of nodes in the network. Nodes that are neighbors in the logical topology are also neighbors in the physical network. Thus, routes established in the logical topology reflect physical routes. We use the term "logical" to indicate that the addressing scheme used to identify nodes and to distribute location information in the topology is the translation of an address in the real addressing space – a universal identifier – into an address in the addressing space used by Tribe – a relative address. The node responsible for performing this translation is called *Tribe anchor node*. Any node can play the role of an anchor node.

Topological-independent identification — In Tribe, the identifier of a node is independent of its topological location. A hash function maps the destination's universal identifier into a valid logical address. The destination's logical address is used to identify the Tribe anchor node responsible for storing the destination's topological-dependent address (the relative address).

Self-organization — As stated before, our proposal is self-organizing and does not require any fixed infrastructure. A self-organizing architecture depends only on the correct operation of its nodes, and does not require the existence of administrative entities or dedicated servers. Furthermore, they are non-authority based networks.

2.2. Indirect Service Model

The purpose of Tribe is to provide indirection for routing between peers in large-scale spontaneous networks. A routing procedure is referred to as indirect when it is performed in two phases. This allows the network to decouple the information about the location of a node from the location itself. This approach has many advantages. Firstly, routing information can be completely distributed. This issue is important for achieving scalability in large-scale networks.

In different contexts, like completely decentralized peer-to-peer systems, the distribution of routing information among the nodes limits the routing overhead at each node. For instance, CAN [15] limits the number of routing information to $O(d)$ and Pastry [16] to $O(\log n)$, where d is the number of dimensions of the CAN addressing space and n is the number of neighbors of a node.

Another advantage of such an approach is that the network supports more frequent node arrivals and departures. In Tribe, location information is completely distributed and the location information stored at each node concerns only a few other nodes. In this way, the overhead produced by the departure/arrival of a node is also low.

Every node in Tribe has three identifiers. The first one, called universal identifier, U, is supposed to be known by any other node that wishes to communicate with the node. This identifier is independent of any network-level characteristics. It can be a word, a numerical value, or even an IP-like address. The second identifier, the logical address V, is a translation of U into the Tribe's logical addressing space, $\mathcal{V}$. This identifier is used to identify any Tribe anchor node. The third identifier, the relative address E, is the current topology-dependent address of the node. It is important to note here that the relative address changes if the node moves, but both the universal and logical identifiers remain unchanged. Fig. 1 illustrates the steps of Tribe's routing procedure and the use of the described identifiers.

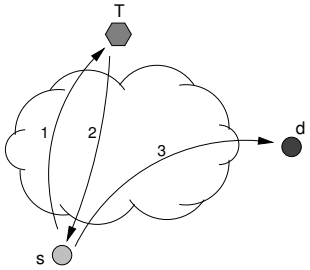


Figure 1. Routing steps in Tribe.

When source s wants to communicate with destination d and has no idea of d's relative address, it first contacts the node responsible for storing the relative address of node d (arrow 1). Call this node T_d. Thus, the message sent by s will travel in the network until it is received by T_d. Note that node s does not know E_d, but it knows V_d (obtained from U_d). Node T_d knows the relative address E_d because node d has previously informed T_d about its current address. The anchor node T_d plays the role of a "rendezvous" point where the location of node d is stored. The particularity of this approach is that the rendezvous point is logically identified and can be any physical node in the network. Anchor nodes are distributed and depend only on the nodes' identifiers. When contacted by s, T_d responds with a message containing the relative address of node d, E_d (arrow 2). Node s can now communicate directly with d (arrow 3).

3. Tribe design

The main component of Tribe is the addressing space used to assign addresses to nodes. Nodes that are physically close in the network also manage close regions in the Tribe logical topology. Each node is

assigned a control region $\mathcal{R} = [R^{\ominus}, R^{\oplus}]$ of the logical addressing space $\mathcal{V}$. We define $\mathcal{V}$ to be the set of integers in the range $[0, 2^m - 1]$.

There are two important properties of the assignment mechanism. First, at any time, the addressing space is completely shared among the nodes, i.e.

$$\mathcal{V} = \bigcup_{k=1}^{N} \mathcal{R}_k, \tag{1}$$

where N is the total number of nodes in the network. Second, these control regions are mutual exclusive:

$$\mathcal{R}_i \bigcap \mathcal{R}_j = 0, \quad \forall i, \forall j, i \neq j. \tag{2}$$

The lower limit of the control region of node i, $R_i^{\ominus}$, uniquely identifies the relative location of node i in the topology. This identification is the relative address E_i of node i:

$$E_i \equiv R_i^{\ominus}. \tag{3}$$

When a node joins the network, it performs three tasks as shown below:

1. The node identifies its neighbors.
2. One of its neighbors will delegate it a part of its own control region.
3. The arriving node identifies its corresponding anchor node and registers with it its relative address.

In the rest of this paper we will see that the control region received by node i serves for two purposes: node identification and routing.

3.1. Bootstrap

Let $t = t_0^i$ be the time node i joins the network and $\mathcal{R}_i(t_0^i)$ be the control region assigned to i by one of its neighbors. This region is called the *original control region* of node i.

- *Original control region*: $\mathcal{R}_i(t_0^i) = [R_i^{\ominus}(t_0^i), R_i^{\oplus}(t_0^i)] \subseteq [0, 2^m)$, is the control region assigned to node i when i joins the network.[1]

[1] For the sake of simplicity, we use the usual mathematical notation $[X, Y)$ to represent $[X, Y - 1]$.

The exact mechanism works as follows. We first assume the existence of some bootstrap mechanism that allows new nodes to identify their neighbors in the network.[2] This results in the discovery by the new node of its *neighbor set* and *neighbor list*, which contains information about all neighbors.

- *Neighborhood set*: $\mathbf{A}_i = \{a_1, \ldots, a_{A_i}\}$ is the set of A_i nodes in the immediate neighborhood of node i.
- *Neighborhood list*: represented by the set $\mathbf{L}_i = \{[E_{a_1}, \mathcal{R}_{a_1}, \mathcal{R}_{a_1}(t_0^{a_1})], \ldots, [E_{a_{A_i}}, \mathcal{R}_{a_{A_i}}, \mathcal{R}_{a_{A_i}}(t_0^{a_{A_i}})]\}$, is the list containing the relative address, the control region, and the original control region of all nodes in the neighborhood of node i.

The neighborhood list is used to determine which neighbor will delegate a portion of its own region to the arriving node.

3.2. Region assignment

Tribe applies two criteria to delegate a region to the new node i. The first selects, among the neighbors of node i, the one that will share its region. This node will be the neighbor that holds the largest region.[3] Call this node the *parent neighbor* of node i, noted, p_i. Second, the region size that will be assigned to i is a fraction of p_i's control region, $\alpha\mathcal{R}_{p_i}, 0 < \alpha < 1$. Unless otherwise specified, we will use in this paper $\alpha = 1/2$.

- *Parent neighbor*: p_i is the node that holds the largest region among the nodes in the neighborhood list of node i. This node will share its region with node i.

Let $(t_0^i - \epsilon)$ be the time just before node i joins the network (ϵ small). At this time, we have (for $\alpha = 1/2$):

$$\mathcal{R}_i(t_0^i) = \begin{cases} R_i^{\ominus}(t_0^i) = \left\lceil \frac{R_{p_i}^{\oplus}(t_0^i-\epsilon)+R_{p_i}^{\ominus}(t_0^i-\epsilon)}{2} \right\rceil, \\ R_i^{\oplus}(t_0^i) = R_{p_i}^{\oplus}(t_0^i - \epsilon). \end{cases}$$

After assigning a control region to i, the new control region of p_i becomes:

$$\mathcal{R}_{p_i}(t_0^i) = \begin{cases} R_{p_i}^{\ominus}(t_0^i) = R_{p_i}^{\ominus}(t_0^i - \epsilon), \\ R_{p_i}^{\oplus}(t_0^i) = \left\lceil \frac{R_{p_i}^{\oplus}(t_0^i-\epsilon)+R_{p_i}^{\ominus}(t_0^i-\epsilon)}{2} \right\rceil - 1. \end{cases}$$

[2] Trivial in wireless environments.

[3] The region size $\mathcal{R}$ is given by $S(\mathcal{R}) = R^{\oplus} - R^{\ominus} + 1$.

We can now redefine Eq. 3 and obtain:

$$E_i = R_i^{\ominus}(t_0^i). \tag{4}$$

Because every node always gives the highest part of its control region to a new arriving node, the node's relative address remains the same.

3.3. Finding a Tribe anchor node

The next step is to identify the node that will be responsible for storing the location information of node i, i.e. the *Tribe anchor node* of node i, T_i.

- *Tribe anchor node*: represented by T_i, is the node whose control region contains node i's logical identifier V_i, i.e. $V_i \in \mathcal{R}_{T_i}$. Node T_i will be responsible for storing the node i's location information, i.e. its relative address E_i.

Observe that i does not know the actual identity of T_i. Node i simply knows that its anchor is the node whose control region contains V_i. Node i sends then a register message to its anchor node indicating its relative address. Here we only show how to identify an anchor node. The way messages are routed will be subject of Section 4.

We described above the procedure used by a node to identify its anchor. This same procedure is also used by a source to discover the relative address of a destination. If the source, s, wishes to communicate with the destination, d, and s is unaware of d's relative address, it must first contact the corresponding d's anchor node in order to be informed about d's up-to-date relative address. By applying the same hash function on U_d, s obtains the same V_d. The source s contacts T_d and communicates directly with d after receiving E_d.

3.4. Region management

As a result of the region assignment algorithm, we can derive some important properties from the manipulation of the control regions. These properties will be used to show that routing in Tribe is simple and efficient.

In order to describe the region management in Tribe, we make the following definitions:

- *Children set*: represented by $\mathbf{C}_i = \{c_1, \ldots, c_{C_i}\}$, is the set of C_i nodes that have node i as parent neighbor, i.e. $p_{c_k} = i$, $1 \leq k \leq C_i$.

– *Descendant set* of node i: $\mathbf{D}_i = \{d_1, d_2, \ldots, d_{D_i}\}$ is the set of D_i nodes that have i as ascendant. Thus, we have $\mathcal{R}_{d_k}(\tau) \subset \mathcal{R}_i(t_0^i)$. This means that the control region of a descendant is always a subset of one of its ascendants' original control region.

The Tribe join procedure establishes a "kinship" between the node that gives a portion of its control region and the node that receives this control region. This kinship is determined by the descendant set. A child is necessarily a descendant. In this way, we also have $\mathbf{C}_i \subseteq \mathbf{D}_i$.

Let $\mathbf{C}_{p_i}(\tau) = \{c_1, \ldots, c_{C_{p_i}}\}$ be the children set for node p_i at time $\tau > t_0^{p_i}$. Suppose that the elements of $\mathbf{C}_{p_i}$ are organized such that c_x is a node that joined the network before node c_y, $x < y$, and $(x, y) \in \{1, 2, \ldots, C_{p_i}\}$, i.e. $t_0^{c_x} < t_0^{c_y}$. Thus:

1. $R_{c_1}^{\ominus} > \ldots > R_{c_{x-1}}^{\ominus} > R_{c_x}^{\ominus} > R_{c_{x+1}}^{\ominus} > \ldots > R_{c_{C_{p_i}}}^{\ominus}$.
2. $S(\mathcal{R}_{c_x}(t_0^{c_x})) = \frac{1}{1-\alpha} S(\mathcal{R}_{c_{x+1}}(t_0^{c_{x+1}})), \quad \forall c_x \in \mathbf{C}_{p_i}(\tau)$.
3. $\mathcal{R}_{c_x}(\tau) \subset \mathcal{R}_{p_i}(t_0^{p_i}), \quad \forall c_x \in \mathbf{C}_{p_i}(\tau)$.
4. $R_{p_i}^{\oplus}(\tau) = \min\{R_{c_x}^{\ominus}(\tau)\} - 1$, $\forall c_x \in \mathbf{C}_{p_i}(\tau)$.
5. $\bigcup_{c_x \in \mathbf{C}_{p_i}} \mathcal{R}_{c_x}(t_0^{c_x}) \subset \mathcal{R}_{p_x}(t_0^{p_i})$.

In Tribe, every node maintains information about its original control region. This information is used by the routing procedure (Section 4) in order to select the next hop when forwarding messages.

In the Tribe protocol, routing performance is strongly influenced by the way the topology is built. In this paper, we propose to adopt a region assignment mechanism that uses a half division criterion to delegate control regions (Section 3.2). This criterion determines the number of children that each node can have and generates the Tribe topology. In Tribe, topologies tend asymptotically toward a subtree of a binomial tree [5].

4. The Routing Procedure

In Tribe, every node stores information concerning its immediate neighbors, as shown in Table. I.

Suppose node s must forward a message to destination d and s knows the d's relative address. Node s first verifies if the destination address corresponds to one of its descendants, in the set $\mathbf{D}_s$. According to the definition of node's descendants (Section 3.4), the control regions of

Table I. Routing table of node i at time τ.

Neighbor	**Relative address**	**Control region**	**Original region**
a_1	E_{a_1}	$\mathcal{R}_{a_1}(\tau)$	$\mathcal{R}_{a_1}(t_0^{a_1})$
a_2	E_{a_2}	$\mathcal{R}_{a_2}(\tau)$	$\mathcal{R}_{a_2}(t_0^{a_2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
a_k	E_{a_k}	$\mathcal{R}_{a_k}(\tau)$	$\mathcal{R}_{a_k}(t_0^{a_k})$

all descendants of s are always a subset of s's original control region, $\mathcal{R}_s(t_0^s)$. Then, if $E_d \in \mathcal{R}_s(t_0^s)$ holds, s concludes that node d is one of its descendants , i.e. $\mathcal{R}_d(\tau) \subset \mathcal{R}_s(t_0^s)$, $t_0^s < \tau$. Source s forwards then the message to its child which is the closest to the destination. This child, c_x, is chosen such that:

$$E_{c_x} = \max\{E_{c_j}\}, \quad c_j \in \mathbf{C}_s(\tau), \tag{5a}$$

$$E_{c_x} \leq E_d. \tag{5b}$$

If, on the other hand, $E_d \notin \mathcal{R}_s(t_0^s)$, then node s sends the message to its parent p_s. The procedure is repeated until the destination is reached.

4.1. Routing toward an anchor node

Now, suppose node s wishes to communicate with the destination d and s is unaware of E_d. Node s should contact d's anchor node, T_d. The same procedure used for routing toward a destination is used for routing toward an anchor node. The only difference is that the target address is the logical identifier of the destination d. As described in Section 3.3, by applying the hash function on U_d, s obtains V_d. The T_d will be the node whose $V_d \in \mathcal{R}_{T_d}$. Thus, in order to routing a message to T_d, node s first verifies if V_d corresponds to one of its descendants, i.e., if $V_d \in \mathcal{R}_s(t_0^s)$. If so, s sends the message to its child c_x such as defined in Eq. 5. On the other hand, node s sends the message to its parent p_s. The procedure is repeated until the node whose control region contains V_d is reached.

4.2. Routing through shortcuts

In practice, it is likely that the set of neighbors of a node is not restricted to its parent and/or children. These links (inherent to wireless environments) may constitute shortcuts when routing messages.

Let us now define the *remaining nodes set* of node i.

- *Remaining nodes set*: $\mathbf{U}_i = \{u_1, \ldots, u_{U_i}\}$ is the set of U_i nodes that are neighbors of node i but there is no kinship between them, i.e. $\mathbf{U}_i = \{\mathbf{A}_i - (\mathbf{C}_i \cup \{p_i\})\}$.

Suppose again that s wishes to forward a message to d and $E_d \notin \mathcal{R}_s(t_0^s)$, i.e. node d is not a s's descendant. At the standard routing procedure, s sends the message to its parent p_s. Nevertheless, considering the shortcuts, node s can also send the message to one of the remaining neighbors $u_k \in \mathbf{U}_s$. In this case, s first verifies if the destination address corresponds to one of its remaining neighbors' descendants. Thus, by using the information concerning its neighbors stored in its routing table, node s forwards the message to its immediate neighbor x such that $E_d \in \mathcal{R}_x(t_0^x)$, for $x \in \mathbf{U}_s$. On the other hand, the message will be forwarded to p_s, which will perform the same procedures.

Shortcuts can also be used if node s wishes to send a message to T_d. In this case, if $V_d \notin \mathcal{R}_s(t_0^s)$, s verifies if V_d is one of its remaining neighbors' descendants, i.e. if $V_d \in \mathcal{R}_x(t_0^x)$, for $x \in \mathbf{U}_s$. If not, s forwards the message to p_s.

The routing procedure always converge because these conditions guarantee that a message is forward to the node that either (i) has node d or T_d as a descendant, or (ii) can find an ancestral of node s that has node d or T_d as a descendant.

Note that the number of immediate neighbors and, consequently, the signaling overhead depend only on the node's range and is independent of the total number of nodes in the system. Each node only stores information about itself and about its immediate neighbors (region and relative address) in its routing table. Furthermore, a small amount of information suffices for routing in the relative addressing space. Tribe limits to $O(k)$ the number of routing table entries in each node, where k is a number of immediate neighbors of a node. Messages are forwarded hop-by-hop until they find the node whose region contains the required address.

5. Node departure

Tribe also deals with nodes that voluntarily join or leave the network. With a node departure, the system must guarantee that an abandoned control region is assigned to a remaining node. We consider here that before leaving its location, a node explicitly hands over its control region and the associated location information database to its parent neighbor.[4]

In the ideal situation, each node manages one control region. Nevertheless, due to the network dynamics, this is difficult to obtain. In some situations, the control region of the leaving node and the one of its parent neighbor do not form a contiguous region. We will see that if the regions are not correctly managed, nodes may misforward packets and the leaving node's remaining children may become unreachable.

5.1. Topology continuity

The stability of the system and of the routing protocol are enforced by the continuity of the regions in the topology. Tribe must guarantee that after a node departure every message addressed to one of its children will be correctly delivered.

First of all, let us define the continuity of regions.

DEFINITION 1. *A subnetwork is said to be continuous if it satisfies the region continuity law.*

DEFINITION 2. (Region continuity law) *Nodes satisfies the region continuity law if messages can be correctly exchanged among them using the Tribe routing procedure.*

A node departure may cause a discontinuity in the topology and make some descendants unreachable. We call these descendants orphans.

- *Orphans set*: represented by $\mathbf{O}_i = \{o_1, \ldots, o_{O_i}\}$, is the set of O_i remaining children of the leaving node i. In this case, $\mathbf{O}_i \subset \mathbf{A}_i$ and $\mathbf{O}_i \equiv \mathbf{C}_i$.

Consider first that i was not the last p_i's child to join the network. Thus, there exists $c_j \in \mathbf{C}_{p_i}$ such that $R^{\ominus}_{c_j} < R^{\ominus}_i, c_j \neq i$.

Recall that the representation of $\mathbf{C}_{p_i}$ satisfies

[4] We assume the existence of some mechanism that allows a node to determine when it is leaving its location. Details on these mechanisms are beyond the scope of this paper.

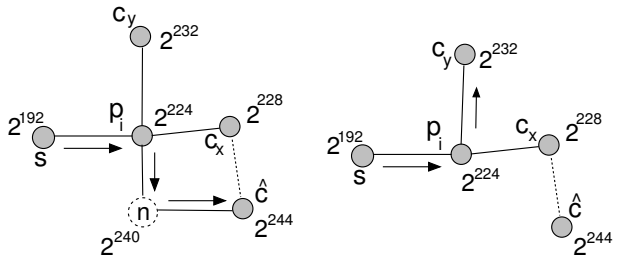


Figure 2. Unreachable orphans because of topology discontinuity.

$$R^{\ominus}_{c_1} > \ldots > R^{\ominus}_{c_{x-1}} > R^{\ominus}_{c_x} > R^{\ominus}_{c_{x+1}} > \ldots > R^{\ominus}_{c_{C_{p_i}}},$$

such that c_k is a node that joined the network before node c_{k+1}, $1 \leq k \leq (C_{p_1})$. Suppose that node i is p_i's xth child, i.e. $i = c_x$. Then, for $\mathbf{O}_i = \{o_1, \ldots, o_{O_i}\}$, we have that $R^{\ominus}_{c_{x-1}} > R^{\ominus}_{o_k} > R^{\ominus}_{c_{x+1}}, \forall o_k \in \mathbf{O}_i$.

Let $\hat{c}$ be the last i's child to join the network, i.e. $\mathcal{R}_{\hat{c}}$ and $\mathcal{R}_i$ form a contiguous region ($R^{\oplus}_i = R^{\ominus}_{\hat{c}} - 1$). At the departure of node i, c_{x+1} will be the p_i's child whose relative address $E_{c_{x+1}}$ is the inferiorly closest to $\hat{c}$'s relative address, $E_{\hat{c}}$. According to the Tribe routing protocol, node p_i will forward to node c_{x+1} any message destined to $\forall o_k \in \mathbf{O}_i$. Thus, if c_{x+1} is not in the i's neighborhood, i.e. ($c_{x+1} \notin \mathbf{A}_i$), and if there is no path respecting the region continuity between c_{x+1} and any other $o_k \in \mathbf{O}_i$, a number of the i's remaining children become unreachable. Fig. 2 shows an example of such a situation.

Before the departure of node i, messages are correctly forwarded to $\hat{c}$ through i, as shown in Fig. 2(a). In this example, the source is node s, whose relative address is $E_s = 2^{192}$. The relative address of the destination $\hat{c}$ is $E_{\hat{c}} = 2^{244}$. Fig. 2(b) shows the topology after node i leaves the network. When node p_i must forward a message addressed to $\hat{c}$, it concludes that it has to choose between two candidate next-hops: node c_x ($E_{c_x} = 2^{228}$), or node c_y ($E_{c_y} = 2^{232}$). Since c_y is the node that gets the message as close as possible to the destination relative address 2^{244}, p_i will forward the message to c_y. Nevertheless, from a global viewpoint of the topology, it is clear that the message should be sent to node c_x. The orphans of node i's become then unreachable after the departure of i.

Therefore, we must guarantee that the orphans of the leaving node remain reachable through some (alternative) valid path. We propose in the following a region reassignment mechanism that guarantees the correct execution of the routing procedure when nodes leave the network.

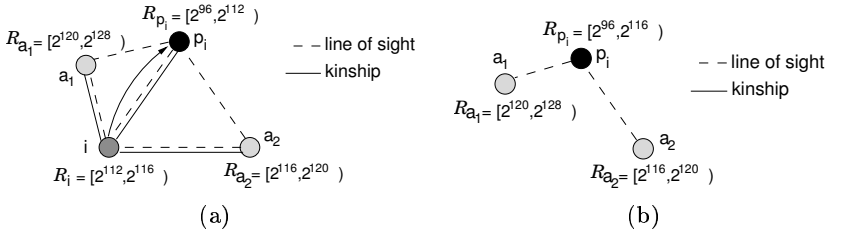


Figure 3. Smooth reassignment.

By allowing nodes to manage discontinuous regions in an adaptive fashion, the global operation of the protocol is maintained. Furthermore, in order to converge to one control region per node, new nodes joining the network will be assigned control regions that have been previously owned by other leaving nodes.

Two cases may happen after a node departure. In the first one, the parent neighbor of the leaving node can assume the responsibility of the abandoned region and nothing else is required to maintain the topology continuity. In the second case, the departure of a node creates some discontinuities in the network topology. The following sections explain these cases and the proposed region reassignment mechanisms in detail.

5.2. Smooth region reassignment

A reassignment operation is said to be smooth when the parent neighbor of the leaving node i, p_i, is also neighbor of i's remaining children. In such a case, i gives its control region and its associated location information database to p_i and nothing else must be done.

A smooth reassignment can be performed if the following conditions hold:

CONDITION 1. $R_{p_i}^{\oplus} = R_i^{\ominus} - 1$.

CONDITION 2. $\mathbf{O}_i \subset \mathbf{A}_{p_i}$.

Fig. 3 shows an example of such a situation. Nodes p_i and i have contiguous regions and p_i sees the two i's orphans a_1 and a_2. Thus, the departure of node i will cause no impact on the functioning of the routing procedure.

5.3. Clone-based region reassignment

When only one or none of the conditions presented in the previous section is respected, the smooth reassignment algorithm cannot be applied.

This section presents the general mechanism to deal with region discontinuity. In this case, the parent neighbor p_i of the leaving node i does not see any i's orphans. We suppose that the network is a connected graph, i.e. there is some path in the network between p_i and $\forall o_k \in \mathbf{O}_i$ (Fig. 4).

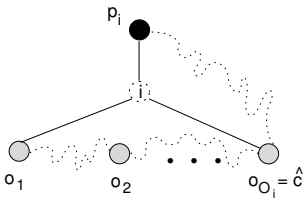


Figure 4. The general case where the smooth reassignment algorithm cannot be applied. Dotted lines represent the existence of (non direct) a path between the nodes.

After the departure of i, a solution to overcome the discontinuity is to establish "tunnels" between p_i and all i's orphans. The idea behind tunneling is to replace i by a mechanism that makes the absence of i completely transparent for the rest of the network. Nodes that are connected through tunnels act as if they were neighbors. The procedure of establishing tunnels is based on the arrival order of the nodes. Let $h(k,l)$ be a tunnel between nodes k and l. Our objective is to establish:

$$\begin{gathered} h(p_i, o_{O_i}) \\ h(o_{O_i}, o_{O_i-1}) \\ \vdots \\ h(o_2, o_1) \end{gathered}$$

For the sake of notation's simplicity, also call $\hat{c} = o_{O_i}$. According to the routing procedure, each node forwards the messages to the immediate neighbor that gets the message as close as possible to the destination. Since we considered that p_i and $\hat{c}$ are connected through a tunnel, any message handled by p_i and addressed to any $d_k \in \mathbf{D}_i$ will be forwarded to $\hat{c}$.

In the same way, the same procedure is performed if there is a tunnel between $\hat{c}$ and o_{O_i-1}. Thus, any message received by $\hat{c}$ and not addressed to one of its descendants will be forwarded to o_{O_i-1}. In this way, the continuity of the routing protocol is maintained if the tunnels are correctly configured.

We supposed above that there was a path between the nodes considered. We show now how these paths are obtained. Let us first show the procedure for creating the tunnel $h(p_i, \hat{c})$. Node p_i performs a limited flooding by sending a `Discover` message destined to $\hat{c}$. Upon the

reception of this message, $\hat{c}$ responds with a `Path` message containing a list $\mathbf{P}(\hat{c} \rightarrow p_i)$ of the intermediate nodes forming the reverse path traversed by the `Discover` message:

$$\mathbf{P}(\hat{c} \rightarrow p_i) = \{b_1, b_2, \ldots, b_M\},$$

where M is the number of hops traversed by the `Discover` message, $b_1 = \hat{c}$, and $b_M = p_i$.

Tunnels are implemented through *clones*. Clones are logical representations of the nodes in the path between two nodes. Every node in $\mathbf{P}(\hat{c} \rightarrow p_i)$ is a potential clone.

The "source" of the tunnel $h(p_i, \hat{c})$, i.e. the clone of node p_i, is responsible for stocking all location information database left by i. All message addressed to any $d_k \in \mathbf{D}_i$ and handled by p_i will be managed by this source and will be forwarded toward the "sink" of the cloned path, the clone of node $\hat{c}$. The other clones between the source and the sink (let us call them "intermediate" clones) simply forward the message to the next clone in the cloned path. The only action of that intermediate clones is to decide if the forwarding should be toward the source or the sink.

Let us now show that the same procedure can be used for establishing tunnels between orphans. Each $o_k \in \mathbf{O}_i$ sends a `Discover` message to o_{k-1}. Upon the reception of the `Path` message, o_k establishes a path from itself to o_{k-1} by cloning all nodes in $\mathbf{P}(o_{k-1} \rightarrow o_k)$. In order to determine the i's descendants that can be reached from each cloned path, i's original control region should be correctly distributed between that cloned paths.

For the tunnel $h(o_{O_i}, o_{O_i-1})$ to be established, the clone of node $o_{O_i} = \hat{c}$ will be responsible for forwarding through the intermediate clones all messages handled by $\hat{c}$ and addressed to $d_j \in \mathbf{D}_i$ if $R^{\ominus}_{d_j} \in (\mathcal{R}_i(t_0^i) - \mathcal{R}_i - \mathcal{R}_{\hat{c}}(t_0^{\hat{c}}))$. Call of $\mathcal{R}_{h(\hat{c}, o_{O_i-1})}$ the resulting control region managed by the tunnel $h(\hat{c}, o_{O_i-1})$.

For the paths $h(o_k, o_{O_k-1})$, $2 \leq k \leq (O_i - 1)$, the control regions managed by each tunnel are calculated recursively. The source of each $h(o_k, o_{O_k-1})$ – i.e. the clone of node o_k – will be responsible for forwarding through the intermediate clones all messages handled by o_k and addressed to $d_j \in \mathbf{D}_i$ if $R^{\ominus}_{d_j} \in (\mathcal{R}_{h(o_k, o_{O_k-1})} - \mathcal{R}_{o_k}(t_0^{o_k}))$. Call of $\mathcal{R}_{h(o_k, o_{O_k-1})}$ the resulting control region managed by each tunnel $h(o_k, o_{O_k-1})$, $2 \leq k \leq (O_i - 1)$. Fig. 5 shows all the tunnel created after the departure of node i.

Consider the following example where node i leaves four orphans $\mathbf{O}_i = \{o_1, o_2, o_3, o_4\}$, for $o_4 = \hat{c}$. The execution of the cloning-based mechanism will result in the configuration shown in Table II.

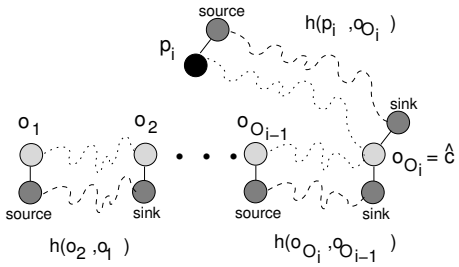


Figure 5. Cloned logical paths between p_i and $\hat{c}$, and the node i's orphans.

Table II. Control regions of the cloned paths.

$\mathcal{R}_{h(p_i,o_4)}$	$= \mathcal{R}_i(t_0^i)$
$\mathcal{R}_{h(o_4,o_3)}$	$= \mathcal{R}_i(t_0^i) - \mathcal{R}_i - \mathcal{R}_{\hat{c}}(t_0^{\hat{c}})$
$\mathcal{R}_{h(o_3,o_2)}$	$= \mathcal{R}_{h(o_4,o_3)} - \mathcal{R}_{o_3}(t_0^{o_3})$
$\mathcal{R}_{h(o_2,o_1)}$	$= \mathcal{R}_{h(o_3,o_2)} - \mathcal{R}_{o_2}(t_0^{o_2})$

An alternative to the cloning-based mechanism would be to change the descendants' addresses and redefine the entire scope from the leaving node. In this case, with a node departure, all its descendants will be forced to rejoin the network, to obtain a new relative address, and to update the corresponding new location information at the anchor node. The overhead of this implementation would affect the protocol scalability and stability.

On the other hand, the maintenance performed by the clone-based mechanism affects only the remaining orphans and some nodes within the discover range. We will see in the section reporting our experimental results that, despite the average number of neighbors per node, the degree of kinship is low. This contributes to reduce the overhead caused by cloning.

5.4. Clone-undoing

The region-reassignment mechanism also aims at keeping one control region per node. Each time a new node arrives in a location that has been previously occupied by another node, the parent neighbor verifies if the new node is appropriate to receive the previous abandoned control region.

The parent neighbor compares the neighborhood list left by the previous node with the one of the arriving node. Cloned paths will be

undone according to the result of that neighborhood comparison. Let node i be the leaving node and node i' be the new arriving node. If the new node i' is neighbor of p_i and of all $o_k \in \mathbf{O}_i$, p_i undoes the clones and assigns the abandoned control region $\mathcal{R}_i$ to node i'. The location information database and the original control region $\mathcal{R}_i(t_i^0)$, are also sent to i'. Nevertheless, if the new node cannot be used to restore the previous region assignment, a region will be attributed to it, according to the described Tribe join procedure.

6. Design Improvements

In this section, we describe some of design techniques for improving Tribe robustness in terms of routing, location information availability, and load balancing.

6.1. Multiple anchor nodes

For improved location information availability, one could assign multiple k different anchor nodes to each node. To accomplish this, we could concatenate a small, globally constant sequence of k values (eg. 1, 2, 3) to each universal identifier, then hash the result to identify the appropriate k anchor nodes. Thus, each node joining the topology would have k different registrations in k distinct locations in the network.

The query for the location of a destination could be done in two ways. First, the query could be sent to all k anchor nodes in parallel. This would reduce the average query latency but, of course, increase the traffic in the network and the cost of storage.

Instead of querying all k nodes, a node might instead choose to contact the anchor node which is the closest to itself in the addressing space. This would also reduce the average query latency.

6.2. Replication and Failures

The advantages of information replication are twofold. First, it increases the availability of a specific location information. Thus, the overhead at the hot spots are reduced. The second advantage is that it also increases the ability of dealing with arbitrary node failures and simultaneous failures of multiple adjacent nodes.

We presented in Section 5 some mechanisms that guarantee the stability of the system and of the routing protocol when a node leaves the network. We had supposed that a node notifies its parent neighbor about its departure by sending the information necessary to assure the topology continuity. Nevertheless, the region continuity law cannot be

assured if a node suddenly fails or does not have enough time to notify about its departure. By replication at k anchor nodes, these information can be restored.

Any anchor node can be informed about a node failure by one of the failed node's remaining neighbors. Once the anchor node has detected that a node has failed or left, it sends a `Repair` message toward the previous emplacement of the failed node. The `Repair` message contains the information necessary to assure the topology continuity. The `Repair` message is intercepted by the failed node's parent neighbor that will initialize the Tribe region reassignment mechanism.

In the ideal case, one of the k failed node's anchor nodes either (i) is an ancestral of the failed node, or (ii) is located in a different subtree from the failed node at the Tribe tree. These conditions guarantee that the `Repair` message will necessarily go through the failed node's parent neighbor. Nevertheless, it is possible that all of the k anchor nodes are descendants of the failed node. In this case, the `Repair` message will not reach the failed node's parent neighbor. In order to deal with this particular case, we can:

- Estimate a value for the replication degree k that makes negligible the probability that all anchor nodes become to the same subtree.
- Discover an alternative path to the failed node's parent neighbor using the same `Discover` message described in Section 5.3.

7. Analysis

In this section, we present a number of simulation results for Tribe in terms of performance and scalability.

7.1. Tribe Simulator

We evaluated Tribe through an implementation of a simulator consisting of the Tribe engine and a network emulation environment. This latter allows experiments with large networks with up to 10,000 nodes.

Each node is supposed to have a 100-meter signal range. The emulated network environment is a 4000x4000 meters square with a granularity of 1x1 m^2, as shown in the grid of Fig. 6. Points in the grid represent locations that may be occupied by nodes. Every point is associated with a degree that is equivalent to the number of nodes whose range contains this point. A *set of degree l* is composed by the

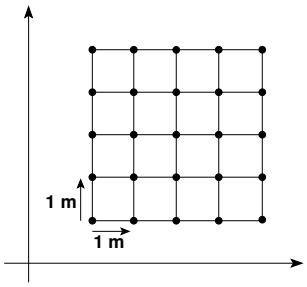
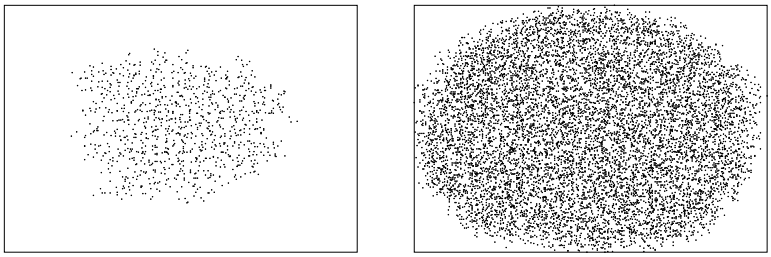


Figure 6. A piece of the universe partitioning.



(a) 1,000 Tribe nodes. (b) 10,000 Tribe nodes.

Figure 7. Emulated network environment.

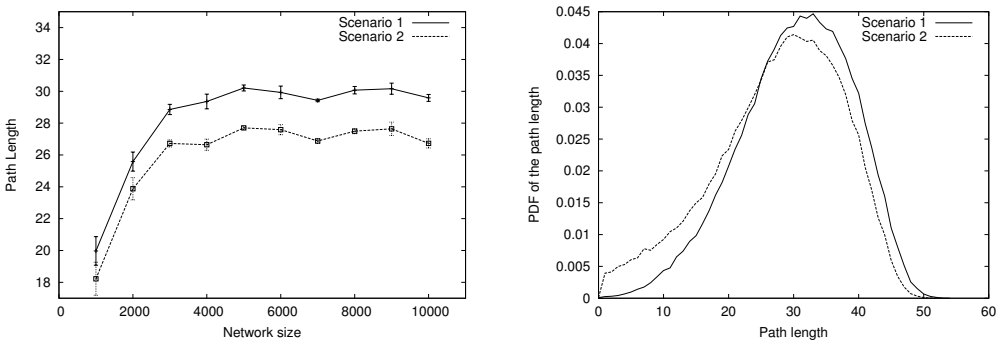
points that have the same degree l. At startup, all points have a degree equal to 0.

The emulated network environment uses a pure random model to determine the initial neighborhood degree of a new node [23]. A probability q to be occupied by the new arrived node is assigned to each point in the grid, being dependent of the point's degree. The simulator increments of 1 the degree of all points in the range of the new node. Fig. 7(a) shows the topology obtained for a 1,000-node network and Fig. 7(b) shows the resulting topology for a 10,000-node network.

7.2. Tribe Results

The basic parameter for estimating the performance of any routing protocol is given by the length of the path between two arbitrary nodes in the network. In the context of the analysis presented in this paper, we consider an addressing space of size $m = 30$ for $\mathcal{V} = [0, 2^m)$. We varied the number of nodes N from 1,000 to 10,000 and conducted a separate experiment for each value. For each experiment two Tribe nodes are selected randomly and a message is routed between this pair of nodes using Tribe protocol. Then, we measured the path length traversed by the message.

The measurements were performed considering two scenarios:



(a) (b)

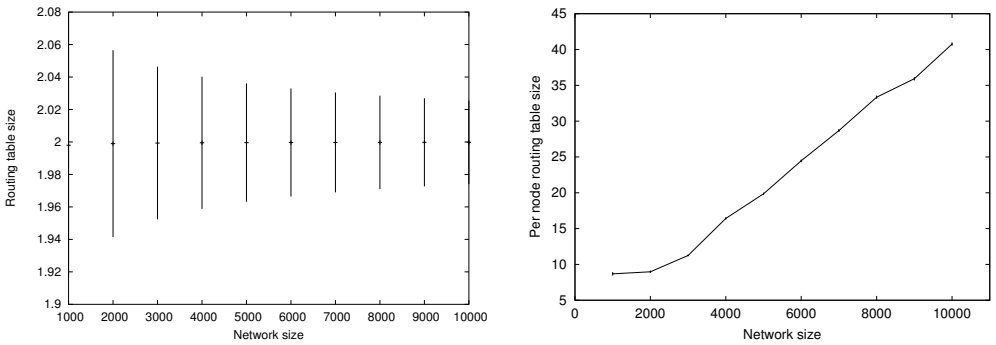
Figure 8. (a) Path length as a function of network size. (b) PDF of the path length in the case of $N = 10,000$-node network and 100,000 trials.

- **Scenario 1:** nodes communicate using the paths defined by the generated topological tree. Thus, the routing table of each node is only composed by the information about its parent and its children.
- **Scenario 2:** in this scenario shortcuts are considered. In this way, the routing table of each node stores information about all the immediate neighbors.

Fig. 8(a) plots the average path length and confidence intervals as a function of the network size for the two described scenarios. The error bars show the 99% confidence interval. The results confirm the expectation that the average path length is in the order of $O(m)$ for larger networks. We also observe that the utilization of shortcuts contributes to a reduction on the path length.

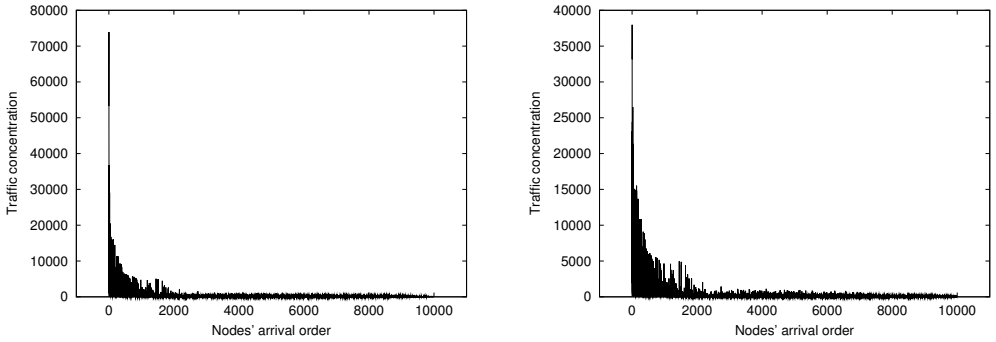
In the same experiment, Fig. 8(b) plots the PDF of the path length for a network with $N = 10,000$ and 100,000 data messages. The results show the path length required to route a message for the two proposed scenarios. We can observe the gain obtained by the use of shortcuts.

Fig. 9 shows the mean and the confidence intervals of the per-node routing table size as a function of the network size. It depicts the effect of shortcuts on the path length. For scenario 1 (Fig. 9(a)), the routing table is limited to the information about the parent and the children of each node. Although shortcuts increase the per-node routing table size, as they enable direct communication between nodes from different subtrees, they lead to significant reduction of the average path length (Fig. 9(b)). It is easier to see here the interest of using shortcuts. They are simply based on the probability that any hop in the path be a neighbor of one of the destination's ancestral. We can also observe that,



(a) (b)

Figure 9. The per-node routing table size (number of entries) as a function of the network size.



(a) Scenario 1. (b) Scenario 2.

Figure 10. The traffic concentration in each node for $N = 10,000$ nodes and 100,000 random communications.

despite the average number of neighbors per node increases with the network size, the degree of kinship is low. This contributes to equilibrate the overhead caused by the cloning algorithm.

It is also important to examine the impact of the topology on the paths between sources and destinations. We investigate the particular hole of the root in the next experiments.

Figure 10 shows the routing traffic concentration for each node for a network size of 10,000 nodes and 100,000 data messages. This experiment considers only data packets. As expected for the scenario 1 (Fig. 10(a)), there is a hot spot situated around the root node. Fig. 10(b), however, shows that when shortcuts are used, the traffic load is reduced of about 50%.

8. Related Works

Following the idea of indirection routing, the *i*3 [18] proposes an overlays-based infrastructure that offers a rendezvous based communication abstraction. *i*3 decouples the act of sending from the act of receiving: sources send packets to a logical identifier and receives express interest in packets sent to this identifier. *i*3 uses a set of servers that store identifiers and map packets with these identifiers to *i*3 nodes interested in receiving the packets. This approach combines the generality of IP-layer solutions with the versatility of overlay solutions. Tribe proposition uses a similar concept of indirect routing, however, it is not based in an overlay infrastructure and is independent of IP-layer.

The geographic forwarding [7, 12, 13, 22] is a routing proposition that better scales in large wireless self-organizing networks. Geographic forwarding requires that end-points find the destination's current geographic location. One approach to tracking geographic locations is through the GLS location service [13]. The Grid project [4, 10] proposes a scalable ad hoc routing protocol, which uses the combination of geographic forwarding and the GLS [13]. GLS builds a distributed location database, where each node chooses its location servers from nodes located at exponentially increasing distances. Both geographic forwarding and GLS use the Global Positioning System (GPS) for determining a node's geographic location. This dependence is likely to be impractical for many uses of self-organizing networks. Tribe routing and GLS use a location service combined with a routing system that uses locations as addresses. Tribe, instead, is completely independent of any geographic information and the topology is a logical network representation, where nodes are identified by their neighborhood in the physical network.

Terminodes [3, 11] project, also built around the concept of geographic forwarding, propose a routing information distribution which is similar to the Tribe location information distribution mechanism, in that they use a translation function to distribute information in the topology. The key difference is that Terminodes uses geographic information to construct the topology and perform routing. Terminodes also proposes a GPS-free positioning for the situation where GPS is not available. This proposal uses distance measurements between nodes to build a coordinate system in order to locate nodes in the network. In Tribe, node position reflects their relative location in the network and there is no need for any geographic positioning system or distance measurements.

Similarly to Tribe approach, the PeerNet [6] is a peer-to-peer based network layer for dynamic and large networks. The address reflects

the node's location in the network and are registered with the respective identifier in the distributed node lookup service. In Peernet, the addresses are organized as leaves of a binary tree – the address tree. PeerNet routing is a recursive procedure descending through the address tree. Thus, in contrast to Tribe, PeerNet routing disseminates information about the global state of the network, and nodes maintain a routing table that has $l = \log N$ entries, i.e. $O(\log N)$ per-node state (where N is a number of nodes in the network). Because the address tree organization, a node movement may require the assignment of new address to several nodes in PeerNet infrastructure, which generates many updates in lookup entries. In Tribe, the number of immediate neighbors and, consequently, the signaling overhead depend only on the node's range and are independent of the total number of nodes in the system. Therefore, Tribe node limits to $O(k)$ the number of routing table entries in each node, being k the number of immediate neighbors of a node. Contrarily to PeerNet, nodes departures in Tribe do not affect others nodes address or generate updates in several anchor node entries. The Tribe region reassignment mechanism is adaptive, and assures the system and routing stability under node departures.

9. Conclusion

This paper addresses the problem of efficiently routing content to a self-organized network. The need for frequent address updates caused by node departure suggests the proposition of a network infrastructure where a permanent node identifier is independent of a topological-dependent address. The Tribe protocol deals with this new requirement and proposes an indirect and scalable routing strategy for self-organizing wireless networks. According to the concept of indirection, Tribe provides an anchor-based abstraction, where the communication is split into two phases: the location of the destination node and the communication itself. In each phase, a distinction between the identity and address of a node is performed, which enables Tribe to handle node departures.

Using peer-to-peer concepts, Tribe achieves high scalability and distributes the location information database among all nodes in the network. The topology created is a logical network representation, which describes the relative location of the nodes according to their neighborhood in the physical network. Therefore, a small amount of information suffices to implement Tribe routing, being the routing table entries limited to $O(\textit{number of immediate neighbors})$ in each node. Upon node de-

parture, Tribe uses an adaptive region reassignment mechanism, which assures the global system and the routing protocol stability.

Some works have observed the power law nature of the load distribution across access points [1, 2]. Motivated by these observations and considering each Tribe node as a potential access point, we expect that most communications will be established within local distances, and little routing traffic overhead will travel the entire Tribe network, alleviating the problem discussed above. Furthermore, in many wireless self-organizing networks, mobility is very low. A notorious example is a wireless mesh network. In such networks, mesh routers are spontaneously placed to extend the wireless coverage. Once the routers have configured themselves, they do not move for a longtime until there in no more need for the network.

We believe that Tribe is an innovative and promising approach for spontaneous networks with low mobility.

Additional related problems that are topics for future work include the extension of Tribe system to use a fairer region partitioning criterion in order to minimize the average number of routing hops in the topology and to enforce a better distribution of the nodes in the network.

Acknowledgements

This work has been supported by CAPES/COFECUB, UFRJ, CNRS, and Euronetlab. We would like to thank Farid Benbadis and Matthieu Delabarre for their useful discussions, and for the implementation and measurements of the Tribe protocol simulator. The authors would also like to thank the anonymous reviewers for their helpful comments.

References

1. Balachandran, A., G. M. Voelker, P. Bahl, and V. Rangan: 2002, 'Characterizing User Behavior and Network Performance in a Public Wireless LAN'. In: *Proceedings of ACM Sigmetrics'03.*
2. Balazinska, M. and P. Castro: 2003, 'Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network'. In: *The First International Conference on Mobile Systems, Applications, and Services.* San Francisco, California.
3. Blazevic, L., L. Buttyan, S. G. S. Capkun, J. P. Hubaux, and J. Y. L. Boudec: 2001, 'Self-Organization in Mobile Ad-Hoc Networks: the Approach of Terminodes'. *IEEE Computer Communications Magazine.*
4. Chambers, B. A.: 2002, 'The Grid Roofnet: a Rooftop Ad Hoc Wireless Network'. Master's thesis, Deparment of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology - MIT.

5. Cormen, T. H., C. E. Leiserson, and R. L. Rivest: 1990, *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company.
6. Eriksson, J., M. Faloutsos, and S. Krishnamurthy: 2003, 'PeerNet: Pushing Peer-to-Peer Down the Stack'. *Proceedings of International Workshop on Peer-To-Peer Systems (IPTPS'03)*.
7. Finn, G.: 1987, 'Routing and addressing problems in large metropolitan-scale internetworks'. Technical report, SI Res. Rep. ISI/RR-87-180, Univ. Southern California, Los Angeles.
8. Freenet website. `freenetproject.org/`.
9. Gnutella website. `gnutella.sourceforge.net/`.
10. Grid Project. `www.pdos.lcs.mit.edu/grid/`.
11. Hubaux, J. P., T. Gross, J. Y. L. Boudec, and M. Vetterli: 2001, 'Towards self-organized mobile ad hoc networks: the Terminodes project'. *IEEE Communications Magazine* **39**(1), 118–124.
12. Karp, B. and H. T. Kung: 2000, 'GPSR: Greedy Perimeter Stateless Routing for Wireless Networks'. In: *Proceedings of ACM Mobicom'00*.
13. Li, J., J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris: 2000, 'A Scalable Location Service for Geographic Ad Hoc Routing'. In: *Proceedings of ACM Mobicom'00*.
14. Peha, J. M.: 2000, 'Wireless Communications and Coexistence for Smart Environments'. *IEEE Personal Communications* **7**(5), 66–68.
15. Ratnasamy, S., P. Francis, M. Handley, R. Karp, and S. Shenker: 2001, 'A Scalable Content-Addressable Network'. In: *Proceedings of ACM Sigcomm'01*.
16. Rowstron, A. and P. Druschel: 2001, 'Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems'. In: *Proceedings of IFIP/ACM Middleware'01*.
17. Satyanarayanan, M.: 2001, 'Pervasive Computing: Vision and Challenges'. *IEEE Personal Communications* **8**, 10–17.
18. Stoica, I., D. Adkins, S. Zhuang, S. Shenker, and S. Surana: 2002, 'Internet Indirection Infrastructure'. In: *Proceedings of ACM Sigcomm'02*.
19. Stoica, I., R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan: 2001, 'Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications'. In: *Proceedings of ACM Sigcomm'01*.
20. Viana, A. C., M. D. Amorim, S. Fdida, and J. F. Rezende: 2003, 'Indirect Routing using Distributed Location Information'. In: *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PERCOM'03)*. Dallas-Fort Worth, Texas.
21. Weiser, M.: 1993, 'Hot Topics: Ubiquitous Computing'. *IEEE Computer Communications Magazine*. (reprinted as Ubiquitous Computing. Nikkei Electronics; December 6, 1993; pp. 137-143).
22. Woo, S.-C. M. and S. Singh: 2001, 'Scalable Routing Protocol for Ad Hoc Networks'. *Wireless Networks* **7**(5), 513–529.
23. Zegura, E., K. Calvert, and S. Bhattacharjee: 1996, 'How to model an internetwork'. In: *Proceedings of IEEE Infocom'96*.
24. Zhao, B. Y., J. D. Kubiatowicz, and A. D. Joseph: 2001, 'Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing'. Technical report, University of California, Berkeley - UCB/CSD-01-1142, EECS.

Authors' Vitae

Aline Carneiro Viana
Aline Carneiro Viana received the B.Sc. Degree in Computer Science in 1997 and the M.Sc. Degree in Electrical Engineering in 2000, both from the Federal University of Goiás, Brazil. In January 2002, she became a PH.D. student at the Network and Performance Group of the LIP6 Laboratory, University Pierre et Marie Curie (Paris 6), France. Her research interests are in self-organizing networks, mobility, and large-scale routing.

Marcelo Dias de Amorim
Marcelo Dias de Amorim is currently a research scientist at CNRS (French National Scientific Research Center). He received the "Cum Laude" Degree in Electronic Engineering from the Polytechnique School of the Federal University of Rio de Janeiro (UFRJ), Brazil, in 1996, and the M.Sc. Degree in Electrical Engineering from COPPE/UFRJ, Brazil, in 1998. He received his Ph.D. degree with honors in computer science from the University of Versailles, France, in 2001. He spent one year at the LRI Laboratory of the University of Paris Sud. From 2001 to 2003, he was a research scientist in EuronetLab, the research laboratory dedicated to next-generation Internet, France. His major research interests are in self-organizing networks, mobility, peer-to-peer networks, and large-scale routing.

Serge Fdida
Serge Fdida is currently a full professor at the University Pierre et Marie Curie (Paris 6). He has been an assistant professor from 1983 to 1989 and a professor with the university René Descartes (Paris) from 1989 to 1995. He also spent a sabbatical year in 1995 with IBM RTP (Raleigh, USA). Serge Fdida is heading the Network and Performance group of the Laboratoire d'Informatique de Paris 6 (LIP6), a research laboratory associated with CNRS (National Scientific Research Center). He is on the editorial boards of Computer Communication and Computer Networks Journals. He served as the program chair and program committee member of numerous international events. He has extensively published in the field of performance evaluation and networking and led several research grants. He was for 8 years, the chair of the French National Research Group on High-speed Networking (RHDM) and is currently the chairman of the European COST264 Action "Enabling Multimedia Group Communication", and the director of Euronetlab, a joint academic-industrial laboratory. Serge Fdida is a senior member of IEEE, member of ACM and IFIP TC6 (WG6.3

and WG6.4). Since early 2000, Serge Fdida is working part-time with CNRS-STIC (French National Scientific Research Center).

José Ferreira de Rezende
José F. de Rezende received the B.Sc. and M.Sc. degrees in Electronic Engineering from the Universidade Federal do Rio de Janeiro in 1988 and 1991, respectively. He received the Ph.D. degree in Computer Science from the Universit Pierre et Marie-Curie, Paris France, in 1997. He was an associate researcher at LIP6 (Laboratoire d'Informatique de Paris 6) during 1997. Since 1998 he is an Associate Professor at Universidade Federal do Rio de Janeiro (UFRJ). His research interests are in distributed multimedia applications, multipeer communication, high speed and mobile networks and quality of service in the Internet.