Indirect Routing Using Distributed Location Information

Aline C. Viana[1,2], Marcelo D. de Amorim[1], Serge Fdida[1], and José F. de Rezende[2*]

[1]LIP6 – UPMC
8, rue du Capitaine Scott
75015 – Paris – France
{viana,amorim,sf}@rp.lip6.fr

[2]COPPE/EE - UFRJ
P.O. Box 68504 – 21945-970
Rio de Janeiro – RJ – Brasil
rezende@gta.ufrj.br

Abstract

This paper proposes the Tribe protocol, an indirect routing strategy for wireless self-organizing networks. The protocol is intended to be applied in environments with large number of users, where mobility is taken into account, and the correct operation of the system does not require the support of a fixed (wired or wireless) infrastructure. In Tribe, nodes build a network infrastructure which describes the node's relative location according to the current node's neighborhood. Furthermore, routing is unique and completely independent of any global connectivity ensured by a network-level routing protocol. The architecture is generic, self-organizing, and independent of IP-like addressing limitations.

1 Introduction

Most of the laptops and personal digital assistants will be equipped with radios enabling them to form spontaneous wireless networks. Such self-organizing networks are supposed to be unsupported by an underlying IP infrastructure and independent of the IP-like hierarchical addressing. The main reason is the need for frequent network addressing updates caused by node mobility.

With IP, when a node moves from one subnetwork to another, the network hierarchical addressing must be continuously updated. The Mobile IP [1] solution works well if there is a fixed infrastructure supporting the concept of "home agent". When all nodes move, including the home agent, such a strategy cannot be directly applied. An interesting technique is to use multi-home agents in the same network, where the corresponding database is distributed among some home-agents. This approach do improve the reliability of the system. Nevertheless, if all home agents move, the system will face the same problems described above.

Tribe is a distributed systems without any centralized control, in which all nodes have identical responsibilities and all communication is symmetric. Nodes build an network infrastructure which describes the node's relative location according to the current node's neighborhood. Each node has its own universal identifier and is assigned a temporary identifier according to its relative location in the network. In Tribe, each node of the network may play the role of home agent, which is similar, in some concepts, to the Mobile IP architecture. Nevertheless, in our approach, the home agent functionality is completely distributed and can be executed by any node in the network. Furthermore, the routing process is assured even when the home agent moves or fails and routing information is completely distributed throughout the network.

The indirect routing strategy of Tribe is similar to recent document location architectures for peer-to-peer networks like CAN [2], Chord [3], Tapestry [4], and Pastry [5]. In these systems, however, routing is performed at the application level and is completely dependent on a global connectivity ensured by a network-level routing protocol, like IP. Tribe is not an overlay network. Tribe routing is performed of an unique manner and is completely independent of a protocol at the network-level.

Terminodes [6, 7] and Grid [8, 9] projects, built around the concept of geographic routing, propose a routing information distribution which is similar to our Tribe location information distribution mechanism, in that they use a translation function to distribute information in the topology. The key difference is that they use geographic information to construct the topology and perform routing. Tribe, instead, is completely independent of any geographic information and the topology is a virtual network representation, where nodes are identified by their neighborhood in the physical network. Terminodes also proposes a GPS-free positioning for the situation where GPS is not available. This pro-

*This work has been supported by CAPES, COFECUB, UFRJ, and CNRS.

posal 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.

The rest of this paper is organized as follows. The main properties of Tribe are discussed in Section 2. Section 3 describes the basic design for the Tribe indirect routing protocol. Section 4 specifies and describes the details of Tribe. The node mobility is treated in the Section 5. Finally, Section 6 concludes the paper.

2 Tribe properties

As stated before, the Tribe protocol takes advantage of some characteristics of completely decentralized systems, self-organizing networks, and propose an architecture independent of any IP-like infrastructure. In the following, we describe the basic characteristics of the Tribe protocol.

2.1 Distributed control

A number of novel network models, including interactive smart devices [10], peer-to-peer applications [11, 12], and pervasive environments in environments without infrastructure [13, 14], are inherently decentralized. An intrinsic characteristic of decentralized systems is the procedure to distribute information and responsibilities, which should be scalable and fault tolerant. In such systems, a mechanism is applied to distribute information and to identify the node which is responsible for storing information that could be retrieved by any node in the network. One technique, also applied by Tribe, is to use a hash function in order to distribute information among the nodes in the topology. Similarly, this technique can also be applied to identify a node which is responsible for storing information (e.g. the location) about any other node in the network. In this way, routing information is completely distributed throughout the network and the network can achieve scalability and more dynamics.

2.2 Physical-Virtual association

Tribe creates a topology which is a virtual network representation and reflects the relative location of nodes in the network. Nodes that are neighbors at the created virtual topology are also neighbors in the physical network. Thus, routes created in the virtual topology will reflect real physical routes. We use the term “virtual” 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 – to an address in the addressing space employed at the created Tribe topology – the relative address.

2.3 IP-independent addressing

Tribe is independent of a fixed IP-like addressing. A hash function maps the universal node identifier into a valid virtual address in a virtual addressing space. The virtual addressing space and the hash function should be chosen in order to minimize the probability of two nodes having the same virtual address. Based on the nodes’ virtual addresses, their location information are distributed to the nodes in the entire topology.

In Tribe, every node is assigned a region (in the virtual addressing space) from on of its neighbors. The node region indicates its relative location in the network and messages are routed to the node whose region contains the searched address. Every node knows the regions of its neighbors, which will serve for routing on a per-hop basis. Thus, Tribe requires only a small amount of routing information and results in low overhead since control messages are exchanged among a node and its direct neighbors. This is the key of the scalability of Tribe.

2.4 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, i.e., they can act in an independent way from any administrative entity. In a self-organizing network, nodes are not regularly distributed and the network density is supposed to be heterogeneous [7].

Besides the self-organizing characteristic, Tribe has some other properties. First, the arrival (and the departure) of a node does not affect the global network structure and the network adapts autonomously to the changes. Second, the region assignment mechanism is the result of the cooperation of a group of nodes. Third, by knowing the universal identifier of a destination node, the routing is performed in a unique manner, independent of the distance between the source and destination. Furthermore, this process depends only on the node’s neighborhood.

3 Indirect Routing

In Tribe, routing is performed indirectly. We say that a routing procedure is indirect when it is performed in two phases. As we will see, this separation allows the network to decouple the information about the location of a node from the location itself. This approach has many advantages. First, routing information can be completely distributed. This issue is important for achieving scalability in large-scale networks. Completely decentralized peer-to-peer systems that distribute routing information among the nodes limit the routing overhead at each node. For instance, CAN limits the number of routing information to $O(d)$ and Pastry limits to $O(\log n)$, where n in the number of direct neighbors of a node and d is the number of dimensions of the CAN addressing space.

Another advantage of such an approach is that the network supports more dynamics. Since the location information concerns only few nodes, the overhead associated with node mobility is also low. Indeed, when a node moves only some other nodes must be informed about the new location of the node and any other node which tries to communicate with the node that moved must first contact the node which has the updated location of the destination (i.e. the home agent).

Tribe uses a topology-dependent routing technique. Every node is identified by its position in the topology (the neighborhood). This position is translated into an address, which is assigned to the node. The only manner of forwarding a message to a destination is by using this address.

Every node in Tribe has three identifiers. The first, the universal identifier U, is supposed to be known by any other node and is network-level independent. It can be a word, a numerical value, or even an IP address. The second identifier, the virtual address V, is a translation of U into the Tribe's virtual addressing space, $\mathcal{V}$. This identifier is used to forward a location query to the home node of the searched node. The last identifier, the relative address E, is the current topology-dependent address of the node. Figure 1 illustrates the steps of Tribe's routing procedure and, consequently, the use of the described identifiers.

When node A wants to communicate with node B, it first contacts the node which is responsible for storing the relative address of node B (arrow 1 in the figure). [1] Call this node H. Thus, A first sends a message to E_H, which is the node with the relative address numerically closest to V_B. Note that node A does not know E_B, but it knows V_B, which is a translation of U_B (universally known). Node H knows the relative address E_B because node B has previously informed H about its current address. Node H acts then as a "rendezvous" point where the location of node B is stored. The particularity of this approach is that the rendezvous point is distributed and depends on each destination. Node H responds to A with a message (arrow 2) containing the relative address of node B, E_B. Node A can now communicate directly with B (arrow 3).

The indirect routing technique incurs some initial delay for retrieving the location information of the destination. Nevertheless, nodes will experience this delay only once, in the beginning of the communication. Furthermore, this delay is compensated by a strong reduction of routing states at the nodes. Many techniques can also be applied in order to reduce this delay, like information replication, caching, and message interception. These mechanisms will be subject of future work.

[1] As stated before, the only manner of forwarding a message to a destination is by using its destination relative address.

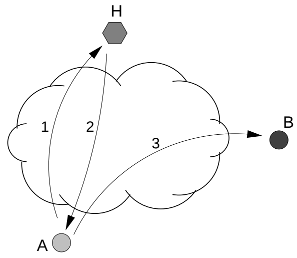

Figure 1. Routing steps in Tribe.

4 Tribe design

Tribe specifies how to locate nodes in the network, how nodes join and leave the system, and how to recover from node failure or mobility. In this section, we describe how nodes are located and inserted in the Tribe topology. In Section 5 we will present how Tribe deals with node mobility.

Tribe creates a network infrastructure which describes the relative location of the nodes according to their neighborhood. It specifies a distributed protocol which allows nodes to manage independent regions of a virtual space. Nodes that are physically close in the network also manage close regions in the Tribe topology. Each node that arrives in the network is associated to a control region of the virtual addressing space $\mathcal{V}$. In Tribe, we define $\mathcal{V}$ to be the set of integers in the range $[0, 2^m - 1]$. The lower limit of the received control region uniquely identifies the node in the topology and is the relative address E of the node. Thus, E is a topology-dependent address and describes the relative position of a node according to its neighborhood.

It is important to note that Tribe topology is not an overlay network, where nodes communicate in an application-level fashion, like CAN [2], Chord [3], Tapestry [4], and Pastry [5]. Instead, Tribe operates at the network level and is completely independent of a global connectivity ensured by a network-level routing protocol like IP. Tribe creates a topology which is a virtual network representation, where nodes are identified by their neighborhood in the physical network.

4.1 The Join Procedure

When a node arrives in the network, it receives a control region which will serve for two purposes: node identification and routing. We note the control region as $\mathcal{R}_i = [R_i^{\ominus}, R_i^{\oplus}[$, where $[R_i^{\ominus}, R_i^{\oplus}[\subset [0, 2^m[$, and the relative address of the node as $E_i = R_i^{\ominus}$.[2] A new node in the network receives the control region from one of its neighbors. We assume the existence of some bootstrap mechanism which allows new nodes to identify their neighbors in the network. This mechanism results in a list containing information about all neighbors. Let $\mathbf{A}_i = \{a_1, a_2, \ldots, a_k\}$ be the set of k nodes in the neighborhood of node i. The *neighborhood list* $\mathbf{L}_i$ of node i is defined as

$$\mathbf{L}_i = \{[E_{a_1}, \mathcal{R}_{a_1}], [E_{a_2}, \mathcal{R}_{a_2}], \ldots, [E_{a_k}, \mathcal{R}_{a_k}]\},$$

where $\mathcal{R}_{a_j}$ is the control region managed by node $a_j, \forall a_j \in \mathbf{A}_i$, i.e. $\mathcal{R}_{a_j} = [R_{a_j}^{\ominus}, R_{a_j}^{\oplus}[$.

The neighborhood list is used to determine which existing node in the neighborhood will give a portion of its own region to the arriving node. Several factors must be taken into account. This paper describes the most important subset of the protocol functionalities. Other characteristics, like node heterogeneity and intelligent space sharing, will be subject of future work.

Tribe applies two criteria to attribute one region to a new node. The first one is to select, among a set of candidate neighbors, the node which will share its region. This node will be the one which has the largest region. This node is called the *parent neighbor*. Second, the assigned region is the highest half part of the original region.

Again, let $\mathbf{A}_i = \{a_1, a_2, \ldots, a_k\}$ be the set of nodes in the neighborhood of node i and $\mathbf{L}_i$ be the corresponding neighborhood list. The parent neighbor p_i is chosen such that

$$S(\mathcal{R}_{p_i}) = \max\{S(\mathcal{R}_{a_j})\}, \quad p_i \in \mathbf{A}_i, \forall a_j \in \mathbf{A}_i,$$

where $S(\mathcal{R}_{a_j})$ is the size of a_j's region:

$$S(\mathcal{R}_{a_j}) = R_{a_j}^{\oplus} - R_{a_j}^{\ominus}.$$

After receiving the region from p_i, we have

$$\mathcal{R}_i = \begin{cases} R_i^{\ominus} = & \frac{R_{p_i}^{\oplus} + R_{p_i}^{\ominus}}{2}, \\ R_i^{\oplus} = & R_{p_i}^{\oplus}. \end{cases}$$

Similarly, the new control region of p_i will be

$$\mathcal{R}_{p_i} = \begin{cases} R_{p_i}^{\ominus} = & R_{p_i}^{\ominus}, \\ R_{p_i}^{\oplus} = & \frac{R_{p_i}^{\oplus} + R_{p_i}^{\ominus}}{2} - 1. \end{cases}$$

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

Note that Tribe guarantees that the node's relative address $E = R^{\ominus}$ uniquely identifies the node in the network infrastructure and remains unchanged even if the node's control region is constantly divided.

4.2 The Register Procedure

After joining the network, the new node i has a relative address E_i. The next step is to identify the node which will be responsible for storing its location information, i.e. the *home node* of node i, h_i. The operation of registering the relative address in the corresponding home node is mandatory for every arriving node.

By using any well-known functions like SHA-1 [15], each node hashes its universal identifier, U, and obtains an m-bit universal identifier, U'. The length m must be large enough to make negligible the probability of two nodes being hashed to the same identifier. The m-bit identifier U' is then translated into the virtual addressing space $\mathcal{V} = [0, 2^m[$. This translation is performed through a classical linear congruential hash function:

$$f(U') = V = aU' + (b \bmod p),$$

where a and b are constants and p is the range of the hash function. This hash function is known by all nodes participating in the network. Note that by setting p as the whole virtual addressing space $\mathcal{V}$, $p = 2^m$, $f(U')$ accomplishes with our purposes of distributing location information of nodes in the entire topology.

The role of the home node is similar to the one of a home agent in Mobile IP [1]. Nevertheless, our approach has a dynamic nature – any node can be responsible for the information about the location of other nodes in the network. Additionally, the control region of each node in Tribe topology determines the virtual addresses and the associated location information which the node is a home node and is responsible for.

4.3 The Routing Procedure

Having obtained its control region, the new node i also learns the relative addresses and control regions of its neighbors. This neighborhood information will compose its routing table. In Tribe, a node routes a message by simple forwarding to the neighbor whose region is the closest to the searched relative address of the destination until the messages reaches the destination.

The arrival of a new node affects only a limited number of existing nodes in a small part of the virtual space. The number of 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. Furthermore, a small amount of information suffices to implement Tribe routing in the relative addressing space. Each node only stores information about itself and about its neighbors (region and relative address). Messages with a given V or E are forwarded hop by hop until they find the node whose region contains the required address. For the consistency of the distribution of the virtual space, when a node gives a portion of its region, it updates its region and sends an message to its neighbors. These neighbors also update their respective table entries of the corresponding region.

4.4 The Location Procedure

In Tribe, the translation function $f(U')$ is also employed by sources to locate a destination in the topology. This hash function, as well as the universal identifiers of the nodes in the network, are known by all nodes participating in the topology. If a source, s, wishes to communicate to a destination, d, and s is unaware of d's relative address, it must first contact d's corresponding home node, using d's universal identifier. As stated before, messages can only be routed using the relative address of the destination and control regions in the employed virtual space. Thus, s performs $f(U')$ and obtains d's virtual address V_d, which is used to identify the home node of d, h_d. Finally, node s contacts h_d and, after receiving the destination's location information, E_d, it can communicate directly to d.

4.5 Exchanging control messages

Without loss of generality, we use a simple example to show how nodes exchange control messages. Consider the topology shown in Figure 2, where the control regions of each node are represented. Recall that the nodes are uniquely identified by the lower limit of the respective control regions. The dashed lines indicate the kinship between two nodes.

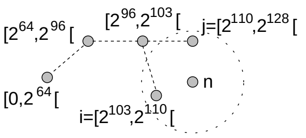

Figure 2. Tribe topology.

When node n arrives, it applies the bootstrap mechanism to identify its neighbors. In our example, node n receives information about nodes i and j and computes $\mathbf{L}_n$. Node n sends then a Join message to nodes i and j containing $\mathbf{L}_n$

$$\texttt{Join}(n) : \{E_k, [R_k^{\ominus}, R_k^{\oplus}]\}, \quad \forall k \in \mathbf{A}_n.$$

Thus,

$$\begin{aligned}\texttt{Join}(n): \quad & \{(E_i, [R_i^{\ominus}, R_i^{\oplus}]), (E_j, [R_j^{\ominus}, R_j^{\oplus}])\} = \\ & \{(2^{103}, [2^{103}, 2^{110}]), (2^{110}, [2^{110}, 2^{128}])\}\end{aligned}$$

The Join message will be received by both i and j and will be used to determine which one will divide its region with node n. As previously described, this node will be the one with the largest region. Since $S(\mathcal{R}_j) > S(\mathcal{R}_i)$, node i will discard the Join message and node j will assign its higher half region to node n, using the Assign message. When a node gives a portion of its region, it updates its region and sends an Update message to its neighbors. Since every node keeps information about its neighbors, they will also update their new region information. The resulting topology is shown in Figure 3.

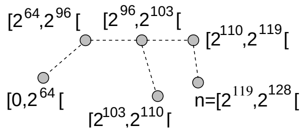

Figure 3. Topology after the arrival of n.

After being assigned a region, node n needs to register itself in the home node. Suppose that $f(U') = V_n = 2^{100}$. Node n sends a Register message toward the node whose control region contains V_n, in this case, $E_{h_n} = 2^{96}$. The Register message contains the received relative address E_n, its universal identifier U_n, and the corresponding virtual address V_n.

When the source, s, $E_s = 0$, wants to send data to the destination, d, $E_d = 2^{119}$, and has no idea of d's relative address, it contacts d's corresponding home node by using V_d. This virtual address, together with U_d, are placed into a Search message that will be sent toward the node whose region contains V_d (the home node), in this case $E_{h_d} = 2^{96}$. The Search message also contains the source's universal identifier U_s and the corresponding relative address E_s. What s seeks is exactly the relative address, E_d, of the destination.

When the home node 2^{96} receives the Search message, it can take two different actions. First, if it does not know the relative address of the destination and replies with a Fail message, indicating that the searched node does not exist or is not registered. Second, if the home node knows the location of the searched node, it replies with a Located message containing the relative address of the researched destination.

5 Mobility and Region Reassignment

In practice, Tribe needs to deal with mobile nodes that join or leave voluntarily the network. With node mobility, two operations must be performed. First, the control region of the moving node must be taken over by one of the remaining nodes. Second, this node must be assigned a new region in the new location.

5.1 Region Reassignment

When a node moves, it explicitly hands over its control region and the associated location information database to its parent neighbor. This operation is crucial and, if not correctly performed, may provoke some inconsistencies at the Tribe system. We propose a region-reassignment mechanism which is executed each time the control region of a leaving node cannot be smoothly merged with its parent neighbor's region. A reassignment operation is said to be smooth when that two control region (leaving node and its parent) and the region of leaving node's remaining children form a contiguous region. Beside that, the parent must be also neighbor of remaining child which the control region is contiguous to leaving node's region. This is illustrated in Figure 4. In this case, the leaving node gives its control region to its parent (Figure 4(a)). Nodes A is the leaving node's child. The parent sees the child A, which has the control region contiguous to the leaving node's region. Beside that, the control region of leaving node, of its parent, and of its child A form a contiguous region. In this case, the node mobility will cause no impact on the organization of the topology. Additionally, the parent neighbor will establish a kinship link between it and the leaving node's remaining child and no other operation will be required.

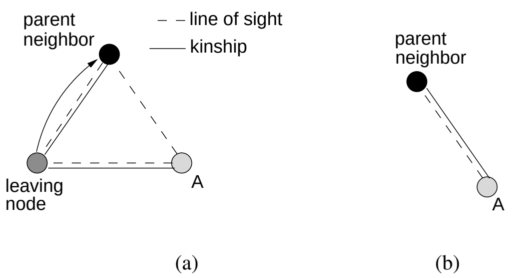

Figure 4. Smooth reassignment.

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 leaving node and its parent do not form a contiguous region and the leaving node's remaining children become unreachable via the routing protocol. It is then fundamental to keep the correct execution of the routing procedure and to ensure that the former neighbors of a mobile node remain reachable through some valid path. Our mechanism allows to uniform the region partitioning, in order to achieve load balancing, and to ensure the correct execution of the routing procedure.

We propose an region reassignment algorithm that aims at maintaining Tribe operational even if parent neighbors deal with discontinuous control regions. Additionally, 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 mobile nodes.

When a mobile node leaves its neighborhood, [3] it sends to its parent neighbor its control region, its associated location information database, and its neighborhood list, containing the relative address and control regions of its neighbors. In this list, the mobile node also points out the neighbors which are the mobile node's children. If the parent neighbor cannot merge its control region with the received region (smooth reassignment), it executes the region-reassignment algorithm.

The stability of the system, and of the routing protocol, is assured by the continuity of the regions inside the topology. The system must guarantee that after a node departure every message addressed to one of its children will be correctly delivered. The parent neighbor of the leaving node must then establish alternative paths to the nodes that have lost their kinship.

Based on the received neighborhood list, the parent neighbor performs a limited flooding by sending a `Discover` message to the node whose control region is contiguous to the handed over control region. Upon reception of the answers to the `Discover` message – called `Path` messages – and based on the received neighborhood list, the parent neighbor selects the node whose list of neighbors is the closest to the leaving node's list. The selected `Path` message conveys the relative addresses of every node traversed and, consequently, informs the number of hops in the path.

The region-reassignment mechanism at the parent neighbor *clones* every node in the selected path and establish an alternative route by using existing nodes in order to replace the previous path. Thus, each node belongs to the main topology and eventually to some other *virtual paths*. The control region handed over by the leaving node is divided by the nodes that will form the virtual path. Thus, each resulting subregion is assigned to one different *clone*. The subregions will be distributed, in an increasing order, among the clones from the parent neighbor to the searched node in the `Path` message. Note that a node can be a clone of several handed over control regions.

Under that new scenario, one node that was cloned will be responsible for managing the location information database associated to its original control region and to its received new subregion. During the routing procedure, a

[3] We also assume the existence of some bootstrap mechanism which allows nodes to previously determine if they will lose the connectivity with their parent neighbor. This is beyond the scope of this paper.

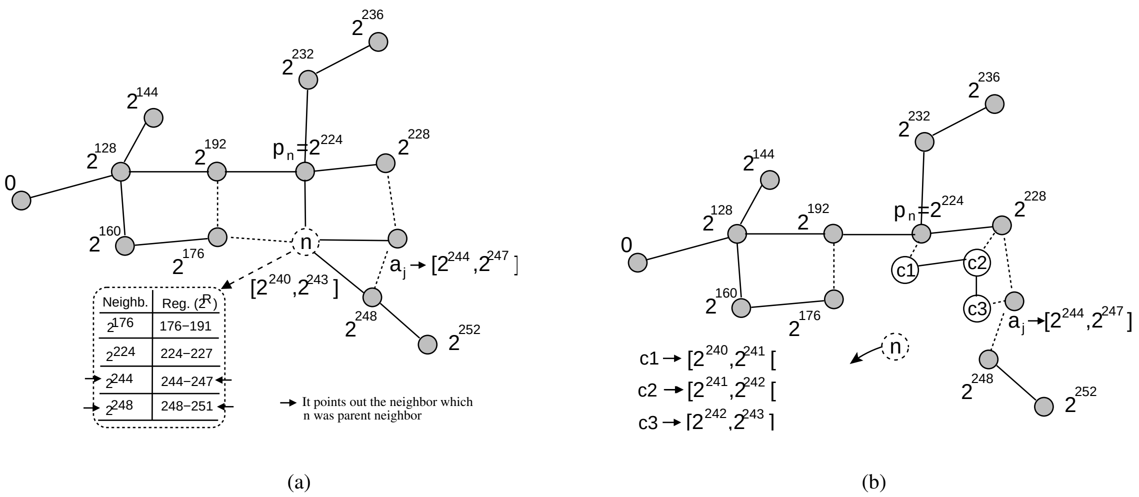

Figure 5. Moving of node n in Tribe topology.

clone first verifies if it can route using one of its subregions. If not, it uses the original region. Figure 5 shows how the region-reassignment mechanism is executed. First, node n hands over its control region, its associated database, and its neighborhood list $\mathbf{L}_n$ to its parent neighbor p_n.

Node p_n sends a `Discover` message implicitly destined to node $a_j \in \mathbf{L}_n$, whose control region is contiguous to $\mathcal{R}_n$. Thus, $p_{a_j} = n$ and $R^{\ominus}_{a_j} = R^{\oplus}_{n} + 1$, where $R^{\ominus}_{a_j} \in \mathcal{R}_{a_j}$ and $R^{\oplus}_{n} \in \mathcal{R}_n$.

When node p_n receives the `Path` messages from node a_j, it selects which one contains the highest number of n's neighbors in the path between p_n and a_j, $\overrightarrow{p_n a_j}$,

$$\mathsf{Path}(a_j) = \max\{S(\mathtt{Path} \cap \mathbf{A}_{n(0)})\} = [E_{b_1}, \ldots, E_{b_k}],$$

where $b_j \in \overrightarrow{p_n a_j}$, $E_{b_1} = p_n$, $E_{b_k} = a_j$ and $S(\mathtt{Path} \cap \mathbf{A}_{n(0)})$ is the number of n's neighbors in the `Path` messages received from a_j.

Consider the scenario illustrated in Figure 5(a), where $\mathsf{Path}(a_j) = \{2^{224}, 2^{228}, 2^{244}\}$. Node p_n divides the handed over n's control region $\mathcal{R}_n$ into three parts and gives one part to each of the nodes to be cloned. Thus, each clone will be assigned one subregion $\mathcal{R}_n^{\frac{i}{3}}$, $\forall \mathcal{R}_n^{\frac{i}{3}} \subset \mathcal{R}_n$. The location information databases associated to the subregions $\mathcal{R}_n^{\frac{i}{3}}$ are sent to the correspondent clones.

Note that the departure of node n neither compromises the correct operation of the Tribe routing procedure nor causes any location information loss. Figure 5(b) shows the distribution, in increasing order, of the resulting subregions among the clones in the scenario discussed above.

In parallel with the cloning procedure, the region-reassignment mechanism is also responsible for retaining one control region per node. Thus, each time that 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 handed over control region. The parent neighbor compares the neighborhood lists sent by the previous mobile node, before its moving, and by the new arrived node. If the new node is also a neighbor of the node which the control region is contiguous to the previous node, the parent neighbor regroups the subregions distributed to the clones and assigns the resulting region to the new node, as in the Tribe join procedure. The clones are then undone, and the location information database associated to the regrouped region is also sent to the new node. If, however, the new node can not be used to restore the previous region assignment, a region will be attributed to it, according to the described Tribe join procedure execution.

5.2 New assignment

Once the mobile node arrives at a new neighborhood, it restarts the Tribe join procedure, which assigns a new control region to the node, corresponding to its new location. The node's universal identifier and, consequently, its virtual address remain unchanged. However, the node's relative address is assigned the lower limit of the new received control region. This new location information is updated with a `Register` message at the correspondent home node, whose identification also remains unchanged. In order to avoid the loss of transit messages with the node's previous relative address, the node also sends a `Register` message conveying its new location information to the previous parent neighbor, also called *secondary* home node, which assumed its former control region. The secondary home node stores the current node's relative address, its universal identifier, and the corresponding virtual address, both unchanged. This location information is stored during a timer period, and it is dropped when the timer goes off. Finally, according to the Tribe's routing procedure, messages sent to the previous relative address are received by the secondary home node and can be forwarded to the up-to-date node location.

6 Conclusion

In this paper, we proposed Tribe, an indirect routing strategy which is unsupported by an underlying IP infrastructure. Tribe is generic, self-organizing, and independent of IP-like addressing limitations. Tribe provides a distributed routing strategy: given a node's m-bit identifier, it determines a home node, which is responsible for storing the node's location information. Each node is assigned a control region which reflects its relative location and serves to identify the range of virtual addresses to which this node will serve as the home node. comprised within its region. During node mobility, Tribe keeps the correct execution of the routing procedure and ensures that the former neighbors of a mobile node remain reachable through some valid path. Additionally, the Tribe's region reassignment allows to uniform the region partitioning and to achieve load balancing. Features of Tribe include its simplicity and the independence of geographic positioning or hierarchical model. Future works include node failure and treatment of node heterogeneity capabilities.

References

[1] C. E. Perkins, *Mobile IP: Design Principles and Practices*. Addison-Wesley, 1997.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of ACM SIGCOMM'01*, Aug. 2001.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of ACM SIGCOMM'01*, Aug. 2001.

[4] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," tech. rep., University of California, Berkeley - UCB/CSD-01-1142, EECS, May 2001.

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proceedings of IFIP/ACM Middleware'01*, Nov. 2001.

[6] J. P. Hubaux, T. Gross, J. Y. L. Boudec, and M. Vetterli, "Towards self-organized mobile ad hoc networks: the terminodes project," *IEEE Communications Magazine*, vol. 39, no. 1, pp. 118–124, Jan. 2001.

[7] L. Blazevic, L. Buttyan, S. G. S. Capkun, J. P. Hubaux, and J. Y. L. Boudec, "Self-organization in mobile adhoc networks: the approach of terminodes," *IEEE Computer Communications Magazine*, June 2001.

[8] Grid: Scalable Wireless Mobile Networks Terminodes Project. `www.pdos.lcs.mit.edu/grid/`.

[9] B. A. Chambers, "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, June 2002.

[10] J. M. Peha, "Wireless communications and coexistence for smart environments," *IEEE Personal Communications*, vol. 7, no. 5, pp. 66–68, Oct. 2000.

[11] Gnutella website. `gnutella.sourceforge.net/`.

[12] Freenet website. `freenetproject.org/`.

[13] M. Weiser, "Hot topics: Ubiquitous computing," *IEEE Computer Communications Magazine*, Oct. 1993. (reprinted as Ubiquitous Computing. Nikkei Electronics; December 6, 1993; pp. 137-143).

[14] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Personal Communications*, vol. 8, pp. 10–17, Aug. 2001.

[15] "FIPS 180-1, Secure Hash Standard." U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, Apr. 1995.