# Party: Pastry-Like Multi-hop Routing Protocol for Wireless Self-Organizing Networks

Ghazi AL SUKKAR
Institut National des Télécommunications
Evry-France
ghazi.al_sukkar@int-evry.fr

Hossam AFIFI
Institut National des Télécommunications
Evry-France
hossam.afifi@int-evry.fr

Sidi-Mohammed Senouci
France Telecom R&D
Lannion-France
sidimohammed.senouci@francetelecom.com

**Abstract:**
In this paper we propose Party, a new routing protocol for wireless Self-Organizing Networks. This protocol is intended to be applied in environments with large number of nodes where the scalability of the routing protocols plays an important issue; it is well known that the current ad hoc routing protocols do not scale to work efficiently in networks of more than a few hundred nodes. In Party, nodes build a network infrastructure which allocates each node a unique temporary address according to its current relative location, our routing is also unique and only depends on the current node's neighborhood, where in order to implement the routing table, each node needs only to exchange local information with its direct neighbors.

**Keywords:** Multi-hop Routing, Distributed Hash tables, Self Organizing Networks, Ad Hoc networks, Mesh Networks, Peer to Peer Communication, Dynamic Address Allocation, Location Service.

## I. Introduction:

Wireless Self-Organizing Networks (SONs) are expected to play an important role in future communications, where they will find wide application scenarios in daily life events. Large-scale events such as disaster relief or rescue efforts are highly dependent on effective communication capabilities. Such efforts could benefit tremendously from the use of self-organizing networks to improve the communications and monitoring capabilities available. Other interesting candidate scenarios are community networks in dense residential areas, large scale, long-range networks in developing regions, and others, where no central administrator exists, or where administration would prove to be too costly. Already, non-military technologies and applications seem to point towards future networks such as: *Ad hoc networks*, *Mesh Networks*, and *Sensor Networks*. All of these applications will place increased scalability demands on Self-Organizing Networks. Scalability is a critical requirement if we want these networking technologies to reach their full potential.

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, this introduces new networking requirements, and we may need novel network architecture. Here, we focus on the network layer of such a future architecture.

As mentioned in [1], the new network layer should be designed to a) minimize the need for manual configuration, b) avoid centralized solutions and node specialization in favor of distributed and peer-to-peer solutions, and c) localize control traffic overhead.

One of the most important components of the network layer is the routing protocol, the current ad hoc routing protocols and architectures work well only up to a few hundred nodes. Most of the current research in wireless SONs routing protocols focus more on performance and power consumption related issues in relatively small networks, and less on scalability. The main reason behind the lack of scalability is that these protocols rely on flat and static addressing. With scalability as a partial goal, some efforts have been made in the direction of hierarchical routing and clustering [2] [3] [4]. These approaches are promising, but they do not seem to be actively pursued. Moreover it appears to us as if these protocols would work well in scenarios with group mobility [5], which is also a common assumption among cluster based routing protocols.

In this paper, we present Party, a new network layer in which the integrated routing protocol is very simple and depends only on node's neighbors. Party is a distributed system without any centralized control, in which all nodes have identical responsibilities. Each node has its own universal identifier (we can use as an identifier, the node's IP address or its MAC address[1]) and is assigned a temporary address according to its relative location in the network, thus the address in this protocol is dynamically changed, with dynamic addressing, nodes change addresses as they move, so that their addresses have a topological meaning.

In order to map the node identifier to its current temporary address, in Party each node of the network may play the role of a home agent (we called it here the rendez-vous node), which is similar, in some concepts, to the Mobile IP [6] 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. Routing in Party resembles to some degree the routing algorithm in Pastry protocol [7].

We have to mention here that Party is not an overlay network protocol, where nodes communicate in an application level fashion, like Pastry [7], CAN [8], Chord [9], and Tapestry [10]. Instead, Party operates at the network level and is completely independent of a global connectivity ensured by a network-level routing protocol like IP. Party creates a topology which is a virtual network representation, where nodes are identified by their neighborhood in the physical network.

The rest of this paper is organized as follows. In section II we describe the related work; Party basic operation is discussed in section III. In section IV we describe the join procedure,

[1] We currently use IP addresses as identifiers. Thus, the transport and application layers do not need to change, and the temporary address is only seen at the network layer.

section V describes the routing procedure, address registration procedure is described in section VI, node lookup procedure is explained in section VII, node mobility and address reassignment are treated in section IIIV, Performance analysis is reported in section IX. Finally we conclude with section X.

## II. Related Work:

In the recently years several routing protocols have been proposed for wireless SONs especially for Ad hoc networks, most of them are IP-based [11] [12] [13], where addresses are static and used to identify the nodes. Since the topology in such networks is dynamic (caused by the mobility of nodes), and the architecture is infrastructure-less, the design of a routing protocol for these networks has two choices: 1) either keep routing entries for every node in the network, or 2) resort to flooding route requests throughout the network upon connection setup. Neither of these alternatives scales well as the network size gets larger.

In order to achieve scalability some protocols were proposed. In the Zone Routing Protocol (ZRP) [14] and Fisheye State Routing (FSR) [15], nodes are treated differently depending on their distance from the destination. In FSR, link updates are propagated more slowly the further away they travel from their origin, with the motivation that changes far away are unlikely to affect local routing decisions. ZRP is a hybrid reactive and proactive protocol, where the network topology is divided in multiple zones.

In multilevel-clustering approaches such as Landmark [16], LANMAR [17], L+ [18], MMWN [2] and Hierarchical State Routing (HSR) [19], certain nodes are elected as cluster heads (also called Landmarks). These cluster heads in turn select higher level cluster heads, up to some desired level. A node's address is defined as a sequence of cluster head identifiers, one per level, allowing the size of routing tables to be logarithmic in the size of the network, but easily resulting in long hierarchical addresses. In HSR, for example, the hierarchical address is a sequence of MAC addresses, each of which is 6 bytes long.

A problem with having explicit cluster heads is that routing through cluster heads creates traffic bottlenecks. In landmark, LANMAR and L+, this is partially solved by allowing nearby nodes route packets instead of the cluster head, if they know a route to the destination. All of the above schemes have explicit cluster heads, and all addresses are therefore relative to these, and are likely to have to change if a cluster head moves away. This reliance on cluster head nodes makes the above schemes best suited to scenarios involving group mobility, such as troop movements.

Another way to achieve scalability is to use geographic location information to assist in the routing, in these protocols [20] [21] it is assumed that each node knows its location coordinates using some technologies (e.g. GPS), although they scale well in large network size, location information is not always available. Taking this in consideration a number of new routing protocols where invented that try to estimate node coordinates in a relative way without the assistant of any positioning system, examples of these protocols NoGeo [22] and GEM [23], where NoGeo embeds the network graph in a virtual 2-dimensional coordinate space, and uses geographical forwarding techniques for routing. The approach is interesting, in that it achieves the *O(1)* complexity of geographical routing, but does not require actual geographical coordinates. However, the scheme will only work on certain types of graphs (typically unit-disk like graphs).

In Area Routing [24], nodes that are close to each other in the network topology have similar addresses, without any explicit nodes hierarchy. PeerNet[1] [25], Tribe [26], and our proposed protocol Party, exploit this idea, i.e. nodes that are neighbors in the topology[2] take addresses that are close to each other. PeerNet is a network layer where node's address reflects its location in the network and are registered with the respective node global identifier in the distributed node lookup service, addresses are organized as leaves of a binary tree (called address tree), PeerNet routing is a recursive procedure descending through the address tree, where routing disseminates information about the global state of the network and nodes maintain routing table that has $l = \log N$ entries (where N is the number of nodes in the network).

In Tribe, nodes are assigned a part of a logical region and a relative address, the relative address of the node also reflects its physical location in the network, Tribe routing depends on the assigned regions, in Tribe the number of entries in the node's routing table is *O(k)* where *k* is the number of immediate neighbors of that node.

Our protocol, Party, resembles PeerNet in the way of address allocation and build routing tables that have the same number of entries as in Tribe, where a small amount of information is suffices to implement routing table, (in contrary to PeerNet[3] where the routing information pass through the whole network). Here each node stores information about itself and its immediate neighbors. Routing is performed in a hop by hop basis, during the routing procedure, each node forward the message to its immediate neighbor which gets the message as close as possible to the destination in a way that resemble Pastry forwarding procedure.

## III. Party Basic Operation:

Party is a completely decentralized and self-organized system. In Party (like in [25] and [26]) each nodes has a globally know and unique identifier[4] and dynamically assigned a unique address which changes with node movement to reflect node's location in the network, this address is used to simplify routing in the network. Since the address of the node changes with node movement, we need a lookup service which will provide the address of a given node identifier.

To join the network, a node establishes a physical connection to at least one node already in the network and requests an address. The neighbor node(s) answer(s) with an address. The joining node then "registers" its identifier together with the address in the distributed node lookup service. As a node moves, it requests and receives new addresses from its new neighbors. Each time the address change, the node updates its entry in the lookup service, each node in the network share in the lookup service, where it can store the mapping entries of other nodes in the network in a way similar to the functionality of home agent in mobile IP [6].

The sender node only needs to know the *identifier* of the receiver. Before sending its first packet to some destination,

---

[1] Which is now called DART protocol.

[2] In wireless SONs, nodes that are neighbors in the network topology are also neighbors in the physical network.

[3] PeerNet uses a modified version of the distance vector routing algorithm to implement the routing tables.

[4] This identifier remains the same and reliably identifies the node.

the sender looks up the current address of the destination node using the lookup service. The routing is done in a way similar to the one done in Pastry [7] one hop at a time, where each node forwards the message to its immediate neighbor which gets the message as close as possible to the destination. If the destination cannot be reached, the lookup table is consulted along the way to find the new address of the destination.

A. Assumptions and Definitions:

We make the following assumptions. First, each node in the network has one unique identifier, call it ID; the ID of the node remains the same during the network lifetime, reliably identifying the node despite its movements and corresponding geographical location changes. Second this ID is supposed to be known by any other node and is network-level independent. Third, an integrated hash-table-like mapping scheme maps identifiers into *rendez-vous node address* (*RA*), we assume the use of a *Consistent Hashing* technique [27] to balance, with high probability, the load among nodes; this hash function *h*(.) is known and common to all nodes. This rendez-vous node is the one responsible for storing the current address of the node with identifier ID.

We illustrate this in the following example, let A be a node in the network with identifier $ID_A$, assume that the temporary address of A is $R_A$. The address of the rendez-vous node $R^R_A$ of A is given by hashing $ID_A$, i.e. $R^R_A = h(ID_A)$, which is node B, then node B store mapping of node A.

B. Address Allocation:

Party enables nodes to allocate addresses in a local way i.e. without the need to contact faraway nodes in the network, at any given time; each node manages a range of addresses including its own address. Node addresses are dynamically assigned depending on the node's current position in the network. More specifically, the addresses are organized as a tree. We call this the *address tree*, see Fig. 1.

Let us assume that addresses are *k* digits decimal[1] numbers, $a_{k-1}, \ldots, a_0$, the first node exist in the network take the all zeroes address 00. . .0, call it the root node, as nodes arrive[2] in the neighborhood of this node (i.e. they are in the transmission range of it), they contact this node to obtain an address (call these nodes level 1 nodes). The root node control the first digit (leftmost digit) of the address, where it give the first arriving node address 100…0, the second arriving node 200…0 and so on up to 900…0. These first level nodes control the second digit (from left) in the address, so when nodes connect to any of these nodes and ask for address, they fix the first digit as their address and change the second digit according to node arriving sequence. For example if a node arrive and it is in the neighborhood of the node with address 100…0 and ask this node for an address, then node 100…0 will give it the address 110…0, the second node ask 100…0 for an address will take 120…0 as an address and so on (we call node 100…0 parent of nodes 110…0, 120…0,…,190…0 and thus they are its children).

These second level nodes take control of the third digit and so on. Fig. 1 show an example of an address tree with three digits addresses, for $k = 3$ digits, the entire address space can be represented by *xxx*, where $x \in \{0, 1, \ldots, 9\}$, nodes in level *l* subtree are the children of the node in level *l-1*. We call the last level nodes in the tree *leaves*.

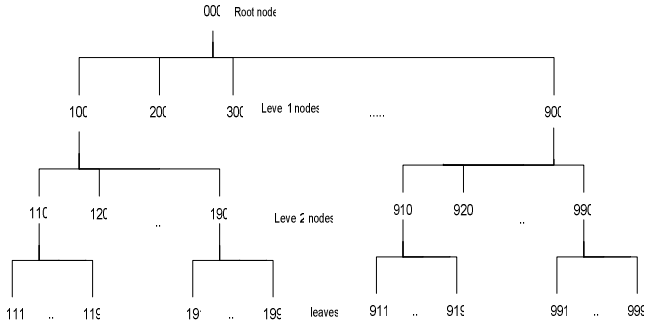


Fig. 1. Address tree with three digits decimal address space.

These leaves do not take control of addresses since address space reaches its limit.

Address tree illustrates how addresses are allocated in Party, it does not represent the actual network topology although address of a node depends on its current position in the network. Fig. 2 shows an example of a network topology with Party protocol in use.

## IV. The Join Procedure:

When a new node *i* arrives in the network, it receives an address $R_i$ (call it temporary address) which will be used for routing. A new node in the network receives the temporary address from one of its neighbors (we call this neighbor the *parent neighbor* $P_i$). 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 $N_i = \{n_1, n_2, \ldots, n_k\}$ be the set of *k* nodes in the neighborhood of node *i* (in its transmission region). The neighborhood list $L_i$ of node *i* is defined as

$$L_i = \left\{ \left( n_1, R_{n_1}, C_{n_1} \right), \left( n_2, R_{n_2}, C_{n_2} \right), \ldots, \left( n_k, R_{n_k}, C_{n_k} \right) \right\}$$

where $C_{n_j} = \left\{ R_{c_1}, R_{c_2}, \ldots, R_{c_m} \right\}$ is the *children list* managed by node $n_j$, $\forall\, n_j \in N_i$.

The neighborhood list is used to determine which existing node in the neighborhood will give a temporary address to the arriving node. Several factors must be taken into account.

Party applies the following criteria to assign one temporary address to a new node. It selects, among a set of candidate neighbors, the node which will be the parent neighbor of the arriving node. This node will be the one with the least level i.e. the nearer to the root. If two or more nodes have the same level then it chooses the node with the least number of children, if a gain two or more nodes satisfy this condition then it will choose the one with the least address.

After the new arriving node chooses the parent neighbor it asks that parent for a temporary address which will be assigned according to our address allocation algorithm, we said that an *association relationship* established between the two nodes. In Fig. 2 this association relationship is represented by continuous thick lines, where the dotted thin lines represent the *neighborhood relationship*.

## V. The Routing Procedure:

Address allocation algorithm in Party simplifies the routing procedure. Routing is performed in a hop by hop basis.

[1] We can use hexadecimal numbers or any base numbers.

[2] We assume that nodes arrive in the network one by one.

Having obtained its temporary address, the new node *i* also learns the temporary addresses of its immediate neighbors. This neighborhood information will compose its routing table. In Party, a node routes a message by simply forwarding to the neighbor whose address is the closest to the searched temporary address of the destination until the messages reaches the destination. This forwarding procedure resembles the forwarding procedure in Pastry [7]; where the message is forwarded to a node from the routing table that has a temporary address with longer shared prefix with the temporary address of the destination.

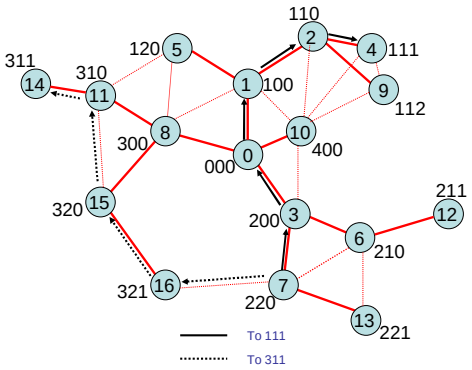


Fig. 2. An example of network topology with 17 nodes and three digits address space. Numbers in the circles are nodes identifiers and at the same represent the sequence of nodes arrival at the network; numbers beside the circles are nodes addresses.

If the node can not find in its routing table such a node that have a longer shared prefix matching, it simply forward the message to its parent an so on until the message reach its destination.

Fig. 2 shows an example of how the routing algorithm works, here node 7 with $R_7 = 220$ want to sent for the destination 14 with $R_{14} = 311$. Node 7 find it is routing table that node 16 has a temporary address that matches the destination temporary address in the first digit, so it forwards the message to this neighbor, in its turn node 16 forwards this message to node 15 which is its parent neighbor since it does not have in it routing table any node that has a longer prefix matching with the destination node's temporary address. Node 15 forward the message to node 11 which has a temporary address that matches the destination's temporary address in two digits. Finally, this node forwards the message to node 14 which is the destination node.

Also Fig. 2, illustrates another routing example, where the source is node 7 and the destination is node 4. As you can note from this example, the message forwarded back to the root node 0 which in its tern forward it to the destination.

The arrival of a new node affects only a limited number of existing nodes (nodes that are in its direct transmission region). The number of neighbors and, consequently, the signaling overhead, depend only on the node's transmission range and are independent of the total number of nodes in the system. Furthermore, a small amount of information suffices to implement Party routing. Each node only stores information about itself and about its neighbors.

## VI. Address Registration Procedure:

After joining the network, the new node *i* has a temporary address $R_i$. The next step is to identify the node which will be responsible for storing its mapping information, i.e. the rendez-vous node of node *i*. The operation of registering the temporary address in the corresponding rendez-vous node is mandatory for every arriving node.

By using any well-known functions like SHA-1 [28], each node hashes its identifier, ID, and obtains an m-bit number. This number is then translated using certain function into a temporary address $R_r$, this address is used to find the rendez-vous node of node *i* as the following. Node *i* forwards a *registration request* message using $R_r$ as a destination address, by applying the routing procedure as in section V. This request will be forwarded until it reaches the node having temporary address that has the longest prefix matching with $R_r$.

This node is the one responsible for storing the mapping information of node *i*; ($ID_i$, $R_i$, $N_i$, $P_i$). This mapping information will be refreshed periodically, as long as node *i* maintain its current position in the network. Also every node in the path to the rendez-vous node will store this mapping in its cache for a certain period of time. This cached mapping is used to avoid the need to contact faraway nodes in the network for the mapping information of a node located in the vicinity of the source node. This will assure that the signaling overhead will be localized as possible.

## VII. Node lookup Procedure:

Since the ID of a node is not its address, Party provides a distributed node lookup service for looking up a temporary address given an identifier. Intuitively, each identifier is mapped through some function to a single address and the node that currently controls that address is required to store the mapping and responding to requests for this mapping.

The source node apply the globally know hash function on the destination node $ID_d$, so it well get a temporary address $R_d$ , this temporary address is the one used to find the rendez-vous node of the destination.

To find the rendez-vous node, the source forwards a *mapping request* message using $R_d$ as a destination address, applying the same routing procedure in section V, each time the message reaches a new node, this node will check its cache for a fresh mapping information, if it find this mapping, then it will respond with *mapping reply* message to the source node, otherwise it forwards the request to its neighbor whose address is the closest to the searched temporary address $R_d$. If no such cached information available in the path, then the request will be forwarded until it reaches the node with the longest prefix matching with $R_d$, this node is the rendez-vous node of the destination. This rendez-vous node will respond with the mapping information for the desired destination node. In the backward path from the rendez-vous node to the source node, this mapping information will be cached for a certain time in each node on the path. This cached information is used in later *mapping requests* by other nodes for the same mapping information.

## VIII. Mobility and Address Reassignment:

Since we are considering here wireless SONs, Party has to deal with nodes that voluntarily join or leave the network caused by node mobility. With a node *i* departure, the system must guarantee the stability of the routing protocol. We consider that before leaving its location, a node explicitly hands over its temporary address $R_i$, its neighborhood set $N_i$,

neighborhood list $L_i$, its children list $C_i$, and the associated mapping information database to its parent neighbor[1].
In this situation Party has to deal with one of the following cases:
Case 1: The leaving node is a leaf node, Fig. 3, shows an example, where node 9 leaves the network (or changes its position), in this case, the node mobility will cause no impact on the organization of the topology, the only process that will take place is the handover of the mapping information database, to the parent $P_i$, and the temporary address of the leaving node will be available again for its parent to be assigned to another node.

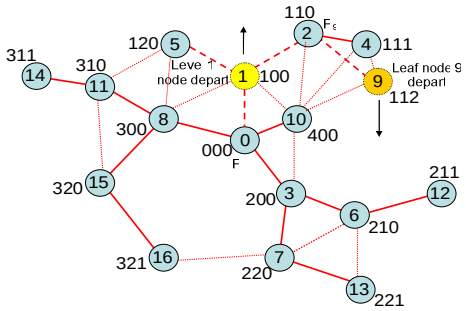


Fig. 3. Leaf node 9, and node 1 leaves (or changes its position).

Case 2: The leaving node is not a leaf node; it could be any node in any level of the address tree. The system must guarantee that after a node departure every message addressed to one of its children will be correctly delivered, and also messages directed through their parent will be correctly delivered. The parent neighbor $P_i$ of the leaving node *i* must then establish alternative paths to the nodes that have lost their association relationship. In Fig. 3 node 1 leaves the network, the parent neighbor of it, is node 0, and it has two children, node 2 and 5.
Based on the received neighborhood list $L_i$ of the depart node *i*, its parent neighbor $P_i$ will face one of the following:

- The children of the leaved node *i* are also neighbors of the parent node $P_i$ of *i* i.e. $C_i \subset N_{P_i}$, in this case the parent neighbor establishes an association relationship with these node, telling them that it now play the role of their previous parent *i* and no other operation will be required. Thus any messaged directed to (through) or from these children will be processed by the parent neighbor of the previously departed node. We call this a smooth reassignment.
- All or some children of the leaved node *i* are note neighbors of the parent node $P_i$ of *i* i.e. $C_i \not\subset N_{P_i}$, in this case, the parent neighbor $P_i$ try to find the set *S* of the children nodes that are also neighbors to it self, i.e. $S = C_i \cap N_{P_i}$, if it is not empty $S \neq \phi$, then the parent neighbor establishes an association relationship with these node as in the previous case. For establishing an association relationship with the rest of children $f = C_i - S$ that are note neighbors of $P_i$, the parent neighbor performs a limited flooding by sending a *reassignment message* directed to each node in set *f*. Upon reception of the *reassignment replay* messages[2], which convey the temporary addresses of every node traversed and, consequently, the number of hops in the path to each node in *f*, the parent neighbor decides which paths (call it *virtual paths*) are appropriate for establishing the association relationship with the nodes in *f* (it takes the paths with the least number of hops), see Fig. 4.
The parent neighbor then sends an *alternative association* message to each node in the path to each child in *f*, informing it of this alternative path. Thus each node in the network will include also in its routing table these virtual paths; it will contain entries for the children temporary address and the next hop node to reach it.
Taking this case in consideration, 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 temporary address and the associated mapping information database.

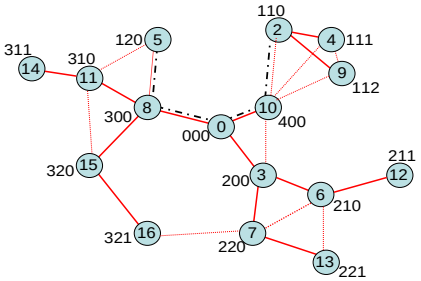


Fig. 4. This figure shows the established virtual paths after node 1 leaved the network.

The parent neighbor compares the neighborhood set sent by the previous mobile node, before it's moving, and by the new arrived node. If the new node is also a neighbor of all the children of the previous node i.e. if $C_i \subset N_{NEW}$, the parent neighbor assigns the temporary address and the mapping information database of the previous leaved node to the new node, and sends *release alternative association* message to each node in the virtual paths. However, if the new node can not satisfy this condition a new temporary address will be attributed to it, according to the described Party join procedure.

## IX. Performance Analysis:

As we said before, Party is a scalable routing protocol; the scalability of this protocol comes from that it is a completely decentralized and self-organized system.
The scalability in Party could be noticed from the following:

- Size of the routing table, in Party each node has a routing table of size *O(k)*, where *k* is the number of immediate neighbors of the node, contrarily to PeerNet [25] where nodes maintain routing table that has $l = \log N$ entries, where N is the number of nodes in the network.
- Signaling traffic needed to implement and maintain the routing table, in Party routing table entries are the immediate neighbors and the only signaling traffic needed is the hello signals between neighbors that used to inform that the node is still alive and still in its position. In PeerNet the information needed to implement and maintain the routing

[1] We assume the existence of a mechanism that allows a node to determine when it is leaving its location.

[2] There is a probability that some of these node will note be reached, in this case we said that the network suffer from topology separation. In this paper we are not going to treat this situation.

table have to pass the whole network, since PeerNet uses a modified version of distance vector routing algorithm.

Comparing our protocol with Tribe [26] we notice that the address allocation in ours is more easy and consequently the routing algorithm. When node moves from its location, the process of reassignment is also simpler than the one in Tribe.

The arrival of a new node and node movement affects only a limited number of existing nodes (nodes that are in its direct transmission region) i.e. its immediate neighbors. Thus the signaling overhead resulting from this action will be small and local.

The cost of connection in Party is *O(1)*, since the only thing that a source needs to establish a connection with a destination is that's destination temporary address. This cost is the same we found in geographical routing protocols where nodes need to know its location coordinate using some technologies (e.g. GPS), which is not always available.

The cost of node lookup is also *O(1)*, since the source needs only to route the lookup message to the temporary address resulting from applying the destination's identity to the globally known hash function. The need to contact a faraway rendez-vous node in order to lookup a node located just near the source is alleviated by the use of cashes.

## X. Conclusion:

Party is a network layer designed for wireless self-organizing networks, it is a decentralized, scalable, and independent of IP-like addressing limitations. Party proposes an addressing structure and allocation that ease routing in such networks, the routing strategy provided by Party is a distributed one, where the forwarding process is done hop by hop in a way resemble the forwarding process in Pastry Peer-to Peer protocol [7]. A tree-like logical topology is created, which describes the relative location of the nodes according to their neighborhood in the physical network.

A small amount of information suffices to implement Party routing, *i.e.*, low signaling overhead is generated (only local neighborhood communication), Thus the routing table size is *O(k)*, where *k* is the number of immediate neighbors of the node. Moreover, a node movement does not affect Party's address tree organization, *i.e.*, it does not require the assignment of new addresses to other nodes already in the network.

We believe, however, that Party is an innovative and promising approach for spontaneous networks with low dynamics, like wireless mesh networks where routers, once they have configured themselves, do not move for a long time.Party support dynamic networks, where during node mobility, Party keeps the correct execution of the routing procedure and ensures that the former neighbors of a mobile node remain reachable through some valid path.

Our future study will include sudden node failures and a treatment of certain network issues, like network separation, networks merging. And a study of this protocol performance through simulation. We are now in the state of implementing this protocol in NS2.

## REFERENCES:


[1] Jakob Eriksson, Michalis Faloutsos, and Srikanth Krishnamurthy, "Scalable ad hoc routing: The case for dynamic addressing," in *IEEE InfoCom*, 2004.

[2] Ram Ramanathan and Martha Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *Mobile Networks and Applications*, vol. 3, no. 1, pp. 101–119, 1998.

[3] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang, "A wireless hierarchical routing protocol with group mobility," in *WCNC*,1999.

[4] G. Pei, M. Gerla, and X. Hong, "Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility," in *ACM MobiHOC'00*, 2000.

[5] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A group mobility model for ad hoc wireless networks," 1999.

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

[7] Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *in Proceedings of the Middleware*, 2001.

[8] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker. "A Scalable Content-Addressable Network," In *Proceedings of the ACM SIGCOMM*, 2001.

[9] Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A Scalable Peer-topeer Lookup Service for Internet Applications," *ACM SIGCOMM 2001*, San Diego, CA, August 2001.

[10] Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in communications*, vol. 22, no. 1, pp. 41–53, January 2004.

[11] Perkins, "Ad hoc on demand distance vector routing," 1997.

[12] Charles Perkins and Pravin Bhagwat, "Highly dynamic destination sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM'94*, 1994.

[13] David B Johnson and David A Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, vol. 353. Kluwer Academic Publishers, 1996.

[14] Z. Haas, "A new routing protocol for the reconfigurable wireless networks," 1997.

[15] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen, "Fisheye state routing: A routing scheme for ad hoc wireless networks," in *ICC (1)*, 2000, pp. 70–74.

[16] Paul F. Tsuchiya, "The landmark hierarchy : A new hierarchy for routing in very large networks," in *SIGCOMM*. 1988, ACM.

[17] G. Pei, M. Gerla, and X. Hong, "Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility," in *ACM MobiHOC'00*, 2000.

[18] Benjie Chen and Robert Morris, "L+: Scalable landmark routing and address lookup for multi-hop wireless networks," 2002.

[19] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang, "A wireless hierarchical routing protocol with group mobility," in *WCNC*, 1999.

[20] Y.-B. Ko and N.H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," in *ACM/IEEE MobiCom*, 1998.

[21] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A distance routing effect algorithm for mobility (DREAM)," in *ACM/IEEE MobiCom*, 1998.

[22] Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *ACM MobiCom*, 2003.

[23] James Newsome and Dawn Song, "Gem: graph embedding for routing and data-centric storage in sensor networks without geographic information," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, New York, NY, USA, 2003, pp. 76–88, ACM Press.

[24] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks: Performance evaluation and optimization,," *Computer Networks*, vol. 1, 1977.

[25] J. Eriksson, M. Faloutsos, and S. Krishnamurthy, "PeerNet Pushing Peer- to- Peer down the stack," *Proceedings of International Workshopon Peer to Peer systems*, IPTPS'03, 2003.

[26] Aline C. Viana, Marcelo D. de Amorim, Serge Fdida, and Jos F. de Rezende, "Indirect routing using distributed location information," *ACM Mobile Networks Applications, Special Issue on Mobile and Pervasive Computing*, 2003.

[27] D. R. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web," in *Proceedings of ACM Symposium on Theory of Computing*, El Paso, TX, May 1997.

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