Performance Evaluation of Party Protocol

Ghazi AL SUKKAR, Mehdi SABEUR, Hossam AFIFI,
Badii JOUABER, and Djamal ZEGHLACHE
Institut National des Telecommunications
Evry, France

Sidi-Mohammed Senouci
France Telecom R&D
Lannion,France

***Abstract*—In this paper we study a self organizing network architecture, Party. Party is a new routing protocol intended to be applied in environments with large number of nodes where the scalability of the routing protocol plays an important role. Party's routing is unique and only depends on the current node's neighborhood. Routing tables are created on the basis of the first hop neighborhood only. We will show the protocol performance with a large number of nodes in the network, and compare it to the legacy ad hoc routing protocols. Results show a large improvement in terms of overhead and throughput.**

Keywords: Multi-hop Routing, Distributed Hash tables, Self Organizing Networks, Dynamic Address Allocation.

I. INTRODUCTION:

Wireless Self-Organizing Networks (SONs) are expected to play an important role in future communications. Such self-organizing networks are not supposed to be underlying on an IP infrastructure nor they depend on IP-like hierarchical addressing.

The routing protocol is a key component in the network layer, the current ad hoc routing protocols 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 [1] [2] [3]. 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 [4], which is also a common assumption among cluster based routing protocols.

In this paper, we study the performance of Party [22], a new network layer in which the integrated routing protocol is very simple and depends only on node's neighbors, each node has its own universal identifier (we can use as an identifier, the node's IP address or its MAC address) and is assigned a temporary address relative to its location in the network. With dynamic addressing, nodes change addresses as they move, so their addresses have a topological meaning.

The rest of this paper is organized as follows. In section II we describe the related work; Overview of Party basic operation is discussed in section III, Performance analysis is reported in section IV. Section V shows the simulation results. Finally we conclude with section VI.

II. RELATED WORK:

Several routing protocols have been proposed for wireless SONs especially for Ad hoc networks, most of them are IP-based [6] [7] [8], where addresses are static and used to identify the nodes. The design of a routing protocol 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) [9] and Fisheye State Routing (FSR) [10], nodes are treated differently depending on their distance from the destination.

In multilevel-clustering approaches such as Landmark [11], LANMAR [12], L+ [13], MMWN [1] and Hierarchical State Routing (HSR) [14], certain nodes are elected as cluster heads (also called Landmarks). 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.

Another way to achieve scalability is to use geographic location information to assist in the routing, in these protocols [15] [16] 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, example of these protocols NoGeo [17].

In Area Routing [18], nodes that are close to each other in the network topology have similar addresses, without any explicit nodes hierarchy. PeerNet[1] [19], Tribe [20], and Party protocol exploit this idea, i.e. nodes that are neighbors in the topology 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. 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

[1] Which is now called DART protocol.

of entries in the node's routing table is *O(q)* where *q* is the number of immediate neighbors of that node.

Party, resembles PeerNet in the way of address allocation and building routing tables that have the same number of entries as in Tribe, where a small amount of information is sufficient to implement routing tables, (in contrary to PeerNet where the routing information passes through the whole network). Here each node stores information about itself and its immediate neighbors.

III. Overview of Party protocol:

Each node has a static globally known and unique identifier ID, Party assigns each node a unique address that changes with node movement to reflect its location in the network (temporary address). This address is used to simplify routing in the network. Since the address of the node changes with its movement, we need an additional lookup service providing the temporary address for 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.

In a typical data exchange, the sender node only needs to know the *identifier* of the receiver. The sender looks up the current address of the destination node using the lookup service. The forwarding process is done in a way similar to the one done in Pastry [5], one hop at a time, where each node forwards the message to its immediate neighbor who forwards 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.

Party basic operation includes the following mechanisms:

A. Address Allocation:

Party enables nodes to allocate addresses in a local way i.e. without the need to contact far away nodes in the network, at any given time; 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, $a_{k-1},\dots,a_0$, the first node to exist in the network takes the all zeroes address 00. . .0, call it the *root node*, as nodes arrive in the neighborhood of this node (i.e. they are in the transmission range of it), they contact it to obtain an address (call these nodes level 1 nodes). The root node controls the first digit (leftmost digit) of the address, where it gives 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 asks for an address, they fix the first digit as their address and change the second digit according to node arriving sequence.

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

The second level nodes take control of the third digit and so on. Fig. 1 shows 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* Є {0, 1,…, 9}, nodes in level *l* subtree are the children of the node in level *l-1*. The leaves do not take control of addresses since address space reaches its limit. Fig. 2 shows an example of a network topology with Party protocol in use.

B. The Routing Procedure:

Address allocation algorithm in Party simplifies the routing procedure. Routing is performed on a hop by hop basis. Having obtained its temporary address R_i from one of its neighbors (we call this neighbor the *parent neighbor* P_i) according to the parent selection mechanism as explained in [22], the new node *i* also learns the temporary addresses of its immediate neighbors through the periodically exchanged hello messages.

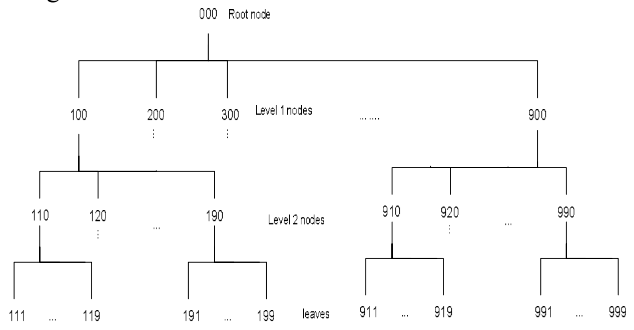

Figure 1. Address tree with three digits decimal address space.

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 its target.

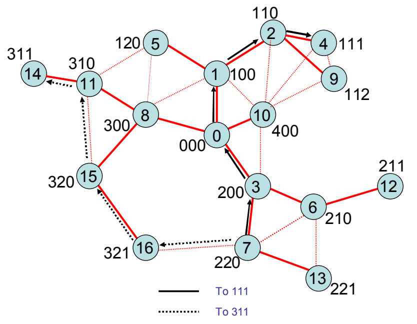

Figure 2. Network with three digits address space. Numbers in the circles are nodes identifiers; numbers beside the circles are nodes addresses.

If the node cannot find in its routing table such a node with a longer shared prefix matching, it simply forwards the message to its parent and so on until the message reaches its destination. Fig. 2 shows an example of how the routing algorithm works, here node 7 with R_7 = 220 wants to send for the destination 14 with R_{14} = 311. Node 7 finds in its 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.

C. *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 function like SHA-1 [21], 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. 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, P_i). This mapping information will be refreshed periodically, as long as node *i* maintains 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. Also to avoid node failure each node could selects more than one node to be its rendez-vous node, where it will store its mapping information simultaneously in all of these nodes.

D. *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 applies the globally known hash function on the destination node ID_d, so it will 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 B, each time the message reaches a new node, this node will check its cache for a fresh mapping information, if it finds 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.

IV. PERFORMANCE ANALYSIS:

A. *Scalability:*

This scalability comes from the following new features:

- Size of the routing table, where each node has a routing table of size *O(q)*, where *q* is the number of immediate neighbors of the node. Recall that in a classical proactive protocol the table size depends on the network size n; i.e. routing table size O(n).
- Signaling traffic needed to implement and maintain the routing table, routing table entries are the immediate neighbors and the only signaling traffic needed is the hello signals between neighbors that used to inform the neighborhood nodes that the node is still alive and still in its position. Classical ad hoc protocols, both proactive and reactive, require much more signaling messages.
- Cost of new joining node, 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.

B. *Number of nodes:*

Referring to the address tree illustrated in Fig.1, the upper bound number of addresses that could be allocated in Party when using base *B* numbers with *k* digits for the temporary address is given by the following equation:

$$1 + m + m^2 + m^3 + ... + m^k = \sum_{n=0}^{k} m^n = \frac{m^{k+1} - 1}{m - 1} \quad (1)$$

Where m = B – 1.

C. *Loop-free:*

At any particular routing step, the packet is never routed to a node whose temporary address is further from the destination's temporary address than the current address. The forwarding process as describe in section III will only forward to a node whose temporary address is closer to the destination's address than the current node's address. When it fails to find such a node, it routes to its parent.

Because the prefix of the parent's temporary address is also a prefix of the current node's address, it is no further from the destination address than the current node.

V. SIMULATION RESULTS:

We evaluated Party's performance using ns-2. Party's results were compared to the results of both a popular reactive routing protocol –AODV [6] – and a popular proactive routing protocol –DSDV [7]. For all simulations, we used the standard values for the Lucent WaveLAN physical layer, and the IEEE

802.11 MAC layer standard with a transmission range of 250m. The duration of all the ns-2 simulations was set to 200 seconds, wherein the first 20 seconds are free of data traffic, allowing the initial address allocation to take place and for the network to organize itself. In order to maintain a mostly connected topology the size of the simulation area was chosen to keep average node degree close to 28. For the traffic we use UDP/CBR flows, where we varied the rate and number of flows and kept the total offered load constant at 250 Kbit/s. In Party protocol we use decimal digits for the temporary addresses. The other Party's simulation parameters used in all simulations are show in table I. We restrict the study to a static topology.

A. Address space utilization:

We set up a series of experiments with network size varying between 100 to 500 nodes. In these experiments the topologies were randomly formed i.e. the position of each node in the simulation area was randomly chosen. The simulation scenario begins with the root node (the node with ID =0), this node preserves the 00…0 address, and begins to transmit HELLO message.

TABLE I: SIMULATIONS PARAMETERS

Parameter	Value
Hello message rate	1s
Mapping registration refreshing rate	3s
Party Signaling Packet size	48 bytes
Address request waiting time	200 ms

When a node in its transmission area hears the HELLO message, it sends ADDR_ALLOCATION_REQ asking this node for an address, and wait 200ms for a response from this node. During this period if it receives the ADDR_ALLOCATION_REP message, it will accept this address and reply with ADD_ALLOCATION_ACCEPT message, registering itself as a child of that node. In its turn it begins to transmit HELLO messages periodically with a period equal to 1s. If it does not receive ADD_ALLOCATION_REP after the expiration of the waiting time, then it will listen again for a new HELLO message.

In these experiments we study the performance of the address allocation procedure used in Party. Fig. 3 shows the convergence time of address allocation as the network size increases; we mean by this, the time that will pass until last node take an address. As expected this time will increase with the network size, it also depends on the number of digits K used for the temporary address, where it decreases with increasing the number of used digits. Although the simulation results show that after 300 nodes this is not the case for K=6 and K=8, this is due to the number of nodes that could not take a temporary address (see Figure 4), as illustrated by the *address tree*, *leaf* nodes could not allocate addresses for other nodes. And since we register the time at which the last node take an address as the convergence time, this time will reach a saturation point, where no more nodes could take a temporary address.

Figure 4 shows that the percentage of node that did not take a temporary address increases as the network size and as expected decreases with increasing the number of digits in the temporary address. Comparing this result with the maximum number of addresses that could be allocated in eqution (1), we can refer the high percentage of nodes not allocated a temporary address, to the random topology formation, where the root node could be located at the edge of the simulation area, and to the simulation scenario where a node try to take an address from the first node transmit a HELLO message.

B. Routing Overhead:

In the following experiments we use 10 decimal digits for Party temporary address i.e. K=10, and in each simulation run we make sure that each node in the network takes a temporary address. Here we compare Party routing overhead with that of AODV, and DSDV, we normalized the packet size for each protocol to that of AODV i.e. 44 bytes. For Figure 5, we examine routing protocol overhead with respect to the number of flows maintaining the network size at 200 nodes.

The results show that our protocol has a lower overhead compared to the other routing protocols. AODV overhead has an approximately linear relationship with flow count, whereas the overhead of DSDV is unaffected by this parameter, due to its proactive route establishment. Party overhead does not affected by the flow count; this is due to the route establishment procedure where no flooding is needed.

In the following experiment, we compare routing protocol scalability with respect to network size. Where Fig. 6 shows that Party maintains a relatively low overhead compared to the other protocols. We can observe also that its overhead grows linearly by a low rate with the network size. This result emphasizes the scalability of Party protocol.

C. Throughput:

We start by studying the throughput achieved by these protocols, using a varying number of UDP/CBR flows. Figure 7 shows that DSDV and Party remain largely unaffected as the number of flows increases. As the number of flows increases, AODV's overhead eats up its initial performance advantage. A slight decrease in throughput is expected, as inter-flow interference will increase with increasing number of flows.

In order to get a good idea of protocol scalability with respect to network size. We keep the number of connections fixed at 100 flows. When connection end-points are chosen randomly and uniformly, it is natural for any protocol to see reduced throughput with increasing network size, due to increasing average path length, and increasing routing protocol overhead. As observed by Figure 8, Party throughput is higher than that for the other protocols, due to its small overhead, thus it seems to be suitable for large networks.

VI. CONCLUSION:

We show how Party is a network layer designed for wireless self-organizing networks, decentralized, scalable, and independent of IP-like addressing limitations.

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(q)*, where *q* is the number of immediate neighbors of the node. The simulation results show that Party performs better than the current ad hoc legacy protocols.

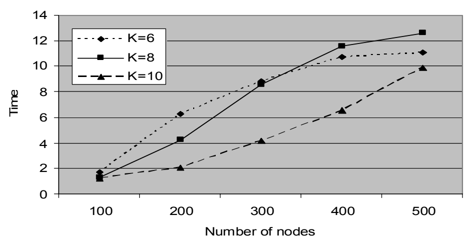

Figure 3. Time needed to allocate addresses for nodes in Party.

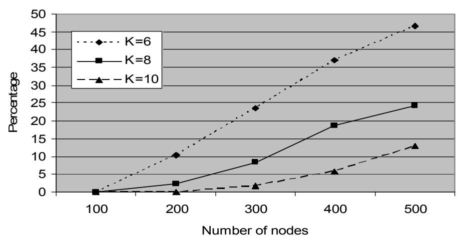

Figure 4. Percentage of nodes that did not take a temporary address.

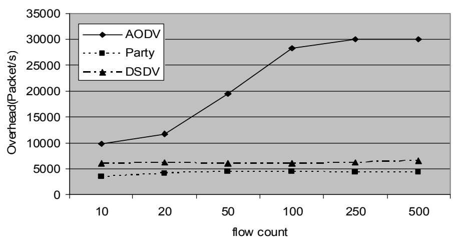

Figure 5. Overhead vs. Flow Count: UDP/CBR flows, 200 Nodes.

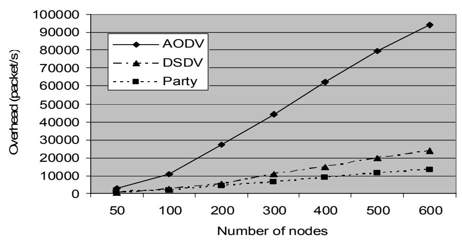

Figure 6. Overhead vs. Network Size: 100 UDP/CBR flows.

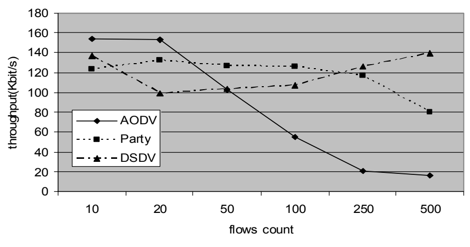

Figure 7. Throughput vs. Flow Count: UDP/CBR flows, 200 Nodes.

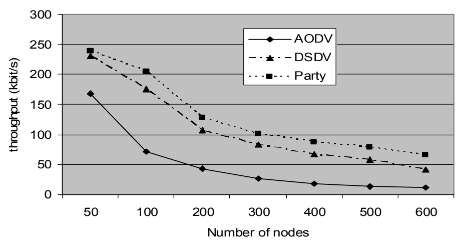

Figure 8. Throughput vs. Network Size: 100 UDP/CBR flows.

REFERENCES

[1] 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.

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

[3] 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.

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

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

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

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

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

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

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

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

[12] 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.

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

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

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

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

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

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

[19] 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.

[20] 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.

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

[22] G. Al sukkar, H. Afifi, and S.-M. Senouci. Party: Pastry-Like Multi-hop Routing Protocol for Wireless Self-Organizing Networks, in proceeding of MCWC, September 2006.