Clustering-based data in ad-hoc networks

Bakhta Meroufel[1], Ghalem Belalem[2]
[1]bakhtasba@gmail.com , [2]ghalem1dz@univ-oran.dz
Department of computer science
Faculty of Sciences
Oran University (Es Senia)
-Algeria-

Abstract: Clustering is an important mechanism in large wireless sensor networks for obtaining scalability, reducing energy consumption and achieving better network performance. Most of the research in this area has focused on clustering based on physical parameters such as: energy, mobility, connectivity, density.., without taking in the count the data stored in each nodes. The main objective of this paper is to provide a useful fully-distributed algorithm for clustering that maximize the intra-cluster access, so we used a new heuristic parameter that combine between energy, mobility and number of data to select the Cluster-Heads. Our clustering strategy was compared with Lowest-ID Cluster Algorithm (*LID*) and the results show that our algorithm improves system performance and increases its life.

Keywords: Clustering, data, ad-hoc networks, availability, mobility.

1. Introduction

A mobile ad-hoc network (MANET) is a collection of mobile nodes that form a wireless network without the existence of a fixed infrastructure or centralized administration. This type of network can survive without any infrastructure and can work independently. Hosts forming an ad hoc network can take equal responsibility in maintaining the network. Each host provides routing services to other hosts to deliver messages to remote destinations. As such a network requires no fixed infrastructure; it makes them better for deployment in a volatile environment such as battlefield and disaster relief situations [6]. Some of the constraints in MANETs are: limited bandwidth, low battery nodes and link frequent breaks due to node mobility. These constraints must be taken into account, while maintaining the connectivity between nodes. Clustering plays an important role in solving such problems [11]. It hides the dynamic structure of the system by forming a hierarchical topology. There are many researches that offer different strategies for clustering based on different metrics.

Unfortunately, the majority of thus works do not take into account the management of requests in the system [1]. In a network where energy is critical, the management of user requests and the difference between intra-and inter-cluster access can consume a lot of energy, which degrades system performances and reduces its life. In this paper, we propose a new strategy of clustering that takes into account the data of the system in addition to mobility and energy of the nodes. Our goals are minimizing the energy used by managing the user's requests and at the same time improving the response time and the stability of the system. The rest of the paper is structured as follows: in the second section, we present some related works on clustering. The third section presents the environment of our work. In section four we introduce our clustering algorithm based on the data, energy and mobility of each node in the system. We validate the proposed approach by a set of experimental results shown in the fifth section, and we finish this work with a conclusion and perspectives.

2. Related works

The algorithms differ on the criterion of selection of cluster-head. Among these algorithms we have: the Lowest-ID Cluster Algorithm (*LID*) [6], in this algorithm, each mobile host in the network must have a unique identifier *id*. The node that has the smallest *id* among all its neighbors is elected as the cluster-head. The cluster is formed by the cluster-head and all its neighbors. *LID* has the advantage of being simple and rapid but also generates a large number of clusters and can be adjustable to changes in the topology. The Last cluster-head Change algorithm (*LLC*)[4], is designed to minimize the change of cluster-head and provides better stability in the composition of system. In High-Connectivity Clustering (*HCC*) [9], [11], the cluster-head election is based on the degree of connectivity (number of neighbors of the node) instead of the identities of the nodes. A node is elected as a cluster-head if it has the highest connectivity among all its neighbors. This algorithm suffers from frequent changes of cluster-head. In [3], two clustering algorithms are proposed, providing a new approach. The first, Distributed Clustering Algorithm (*DCA*) which is targeted to "quasi-static" in which the movement of nodes must be "slow". A weight is defined by the speed of each node. The criterion for the election of the cluster-head is the maximum weight in its neighborhood. The node whose weight is the greatest among all its neighbors is elected cluster-head. The second algorithm is designed for networks and mobility called Mobility-Adaptive Clustering algorithm (*DMAC*). Each node reacts locally to all changes depending on its status: member node or cluster-head. In the two algorithms, it is assigned different weights to the nodes and it is assumed that each node has knowledge of its weight. A node is selected as cluster-head if its weight is greater than among all its neighbors. There are other works that take into account data management in ad hoc networks, such as work [13] which proposes a strategy with two steps: first create the cluster and then replicate the data requested in each cluster. The works [12] and [7] also proposed methods of replication to improve availability of data or to facilitate the update and allocation of data. All works cited precisely separate between the clustering and data management and use the replication to improve their approaches. In this paper we propose a strategy of clustering that includes data management and the creation of clusters in one step by taking advantage of existing data without creating other replicas in the system.

3. Contributions

Proper management of queries in an ah-hoc network improves the reliability and usefulness of the system but increases energy and reduces the lifetime of the nodes. The majority of the works cited above focus on the physical characteristics of networks such as: energy, connectivity, mobility, density, without taking into account the usefulness of the network itself, that is to say the goal of building the networks, the type of services provided and the types of treatments that can be achieved. In this paper, we propose a new algorithm for non-overlapping clustering (see definition 2) to minimize power consumption and response time in the system. Minimizing the access time of queries depend on the location and distribution of data in the system [8]. The main idea in this work is to maximize intra-cluster access by maximizing the number of different data in the same cluster and remove the replica of same data. In the example of Figure 1, the system contains two clusters, each cluster has two different data (two colors) and in this case, the cluster can satisfy only queries that research these two data. As against the second system in Figure 2, each cluster contains three data which increases the number of requests satisfied at the intra-cluster.

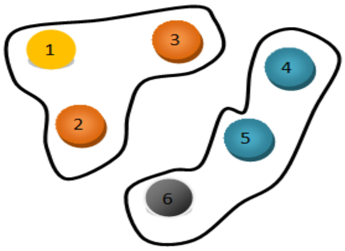

Fig1: First system: two colors in each cluster

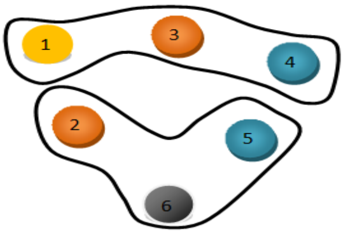

Fig2: Second system: three colors in each cluster

Model

The system is modeled by an undirected graph $G = (V, E)$ where V is the set of network nodes and E models all the connections between these nodes. An edge $(u, v) \in E$ if and only if nodes u and v can mutually receive transmissions of each other. This means that all links between nodes are bidirectional. In this case, we say that u and v are neighbors. The set of neighbors of a node $v \in V$ is denoted $Neigh_v$. Each node u of the network is associated to a unique *id* and can communicate with its neighbors $Neigh \subseteq V$. Each node is characterized by its mobility *Mob* and energy *Energ* remaining in the battery, it also can store zero or more data D_i. Users can initiate read requests to access data in the system.

For the sake of clarity, we will use later in this paper the following notations:

- $Neigh_u$: The neighbors of node u.
- $Energ_u$: The remaining energy of the battery of node u.
- Mob_u: The mobility of the node u.
- $ListL_u$: List local data stored in the node u.
- $ListT_u$: Total list of data stored in the node u and its neighbors (see formula 2).

Before describing our clustering algorithm in detail, we make the following assumptions, which are common in the design of clustering algorithms for MANETs [1], [3], [6], [11], [13]:

- The network topology is static during the execution of the clustering algorithm.
- A packet transmitted by a node can be received correctly by all its neighbors in a finite time.
- Each node has a unique *id* and knows its neighbors and vice versa.
- The inter-cluster access is expensive compared to intra-cluster access in terms of: bandwidth and energy.

One more of these assumptions, we also assume that all requests made by users are of the reading type [10].

4. Clustering

The clustering algorithm consists of two steps: the selection of cluster-head and then the construction of clusters.

4.1 Metric of clustering

Given the interest in the concept of clustering and its undeniable contributions to improve the performance of an ad hoc network, the choice of the clustering mechanism is important. Thus, a clustering algorithm must first be able to select the appropriate nodes to ensure the functionality of the cluster-head. In our algorithm, the cluster-head selection is based on a new metric γ:

$$\gamma_n = \frac{Energ_u}{Mob_u} * \| \, ListT_u \, \| \quad \textbf{(1)}$$

- $\tilde{a}_u$: Metric of clustering.
- $\|ListT_u\|$: The cardinality of the data list of the node u and its neighbors.

$$ListT_u = \bigcup_{i=1}^{i=m} ListL_m \; \bigcup ListL_u \quad \textbf{(2)}$$

- m: The number of neighbors of node u.

The node with the maximum γ in the neighborhood can be selected as a cluster-head. The rapport between energy and mobility helps to select a cluster-head which has a relatively high energy capacity and low mobility witch increase the stability of the system. Held that the parameter $\|ListT_u\|$ can elect a cluster-head which has a great opportunity to satisfy read requests in a single degree at the neighborhood.

Maximize the number of different data per cluster can be achieved by choosing as cluster-head node, which in its first neighborhood has many different data including its own list of data (Maximum $\| \, ListT_u \, \|$). For example in Figure 3, assuming that the report $\frac{Energ_i}{Mob_i}$ $/i \in [0,3]$ is the same for all nodes i (just to explain) then:

- For node 0, the list $ListT_0$ = {A} ∪ {A, Z} ∪ {A, H} = {A, H}.
- For node 1, the list $ListT_1$ = {A, H} ∪ {C, B, F} ∪ {A} ∪ {A, Z} = {A,B,H,F,C,Z}.
- For node 2, the list $ListT_2$ = {A, Z} ∪ {A, H} ∪ {A} = {A, E, H}.
- For node 3, the list $ListT_3$ = {C, B, F} ∪ {A, H} = {A, H, C, B, F}}

We note that the node1 in its neighborhood contains a lot of data compared to its neighbors: $| \, |ListT_1 \, \|>\|ListT_3 \, \|>\|ListT_2 \, \|>\|ListT_0 \, | \, |$, so node 1 is capable of meet a lot of requests (read requests) by at most one jump (zero jump if data is stored in the node itself). In this case, the node 1 will be selected as a cluster-head and broadcasts its status to its neighbors.

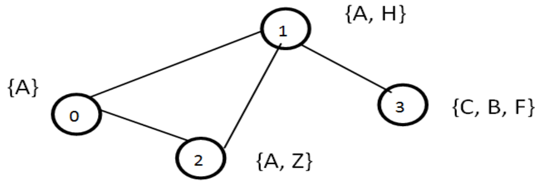

Fig3. A system with 4 nodes

4.2 Clustering algorithm

The construction of the clusters is through periodic exchange of messages which we will call *hello* messages. Each network node exchanges with its neighbors the *hello* messages.

Each *hello* message sent by a node u contains four values are: id_u, $Statut_u$, ($Energ_u$ / Mob_u) and $ListL_u$.

- id_u: the identifier of the node u.
- $Statut_u$: The role of the node u in the system (See definition 3).

This message is also used for each node to verify the presence of its neighbors. Thus, if a node no longer receives *hello* message from a neighbor at the end of a period, it considers that this neighbor has disappeared. So each node waits a specified period in advance and it is assumed that during this period, all nodes have sent their *hello* message. In our algorithm we use the following definitions:

Definition 1 (Cluster): We define a cluster C_i by a connected sub-graph of the network, with a diameter less than or equal to 2. The cluster has an identifier corresponding to the identity of the node with the higher γ in the cluster, that is to say that if the cluster is the cluster C_i then the cluster-head of this cluster has the identifier $id = i$.

Definition 2 (Non-overlapping): a non-overlapping clustering ensures that there is no node that belongs to two different clusters at the same time witch minimizes the conflict.

Definition 3 (node status): Each node u has a status $Statut_u$, the $Statut_u$ can be one of the following: *CH*: cluster-head node, *MN*: member node, or *GN*: gateway node. These three roles are defined in definitions 3.1, 3.2 and 3.3:

Definition 3.1 (Cluster-head): A node u is called cluster-head of cluster C_i iff: $id_u = i \wedge \forall v \in Neigh_u, \gamma_u > \gamma_v$.

Definition 3.2 (member node): A node u is said member node if it is not a cluster-head and it does not have in its neighborhood a node associated to a different cluster. $u \in C_i$ Then $id_u \neq i \wedge [\nexists v \in (C_j / C_j \neq C_i) \wedge v \in Neigh_u]$

Definition 3.3 (Gateway node) A node u is a Gateway node iff: $u \in C_i$ Then $\exists v \in Neigh_u / v \in C_j \wedge C_j \neq C_i$. Gateway node has a special role. It provides access to one or more neighboring clusters.

Definition 4 (Degree of no-similarity): the degree of non-similarity $\bar{\beta}$ between two nodes u and v is the size of the list of data that exists in a node and does not exist in the other (formula 2). Two nodes with a high $\bar{\beta}$ need each other more then two nodes with a lower $\bar{\beta}$ (disjointness of contents).

$$\bar{\beta}(u,v) = \|(ListL_u \cup ListL_v) - (ListL_u \cap ListL_v)\| \qquad \textbf{(3)}$$

Where:

- u,v: Nodes.
- $ListL_u$: List of local data in the node u.

For example, if local lists of data in nodes u, v, h are $ListL_u = \{A, B, C\}$, $ListL_v = \{A, E, F, G, H\}$, $ListL_h = \{A, B, C, D\}$ respectively then:

- $\bar{\beta}$ (u, v)= || {B, C, E, F, G, H} || = 6.
- $\bar{\beta}$ (u,h)= || {D} || =1.

Clustering steps are:

1. Each node calculates the parameter γ and disseminates information on its neighbors by the *Hello* packet.

2. If the node has the maximum γ among its neighbors then it becomes the cluster-head (Status = CH) and sends requests to join the cluster to the other. If two nodes *u* and *v* have $\gamma_u = \gamma_v$, then the cluster-head is the node that has the lowest *id*.
3. The node that receives a request to join a cluster, then it became with a Status = MN.
4. If the node receives several requests from different cluster-head to join their clusters, then it becomes a gateway node: Status = GN. But as a cluster selects the one with the maximum γ. In case of a tie, the node selects the cluster-head that has the maximum no-similarity degree $\bar{a}$ with it.
5. If two neighboring nodes have both a Status = CH, then the first node that detects the conflict through *hello* messages received compared γ to that of its cluster-head neighbor. If the γ is smaller, it abandoned his status as cluster-head and becomes a member node and sends a hello message to its neighbors announcing its new status. Otherwise, it retains the role of cluster-head.
6. The algorithm terminates when each node in the system specifies its status.

For example in Figure 4: $\gamma_0 = 20$, means the node will be a cluster-head. The node 2 will be a node member in the cluster of node 0. Node 7 is a getaway node. The result of clustering is shown in Figure 4.

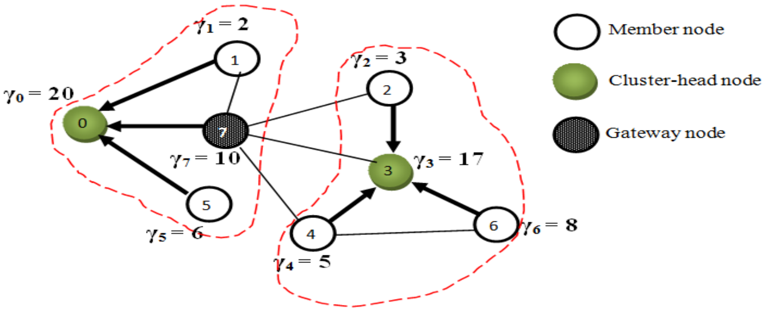

Fig 4. Example of clustering of 8 nodes

4.3 Maintenance of the topology

In ad hoc networks, topology changes frequently due to node mobility and energy. So we have to manage:1) nodes that move, 2) nodes that disappear and nodes that appear. Our algorithm can also manage these cases.

- The appearance of a new node. When a new node enters the network, it broadcasts at a regular interval of time a message of type *hello*. At the end of the timeout, if it has not received any type *hello* message from a cluster-head, the node becomes cluster-head. If the node received a *hello* message from a cluster-head, the node becomes a member node. In the event that it receives from more than one cluster-head, the node becomes a gateway node (Execute step 4 of the clustering algorithm)
- Disappearance of a node. In the same way as for the appearance by exchanging *hello* messages the neighbors are aware of the disappearance.
- Moving a node. Case equivalent to the emergence of a new node. In this case the *hello* messages will be received by the other new neighbors.

5 Implementation and validation

To validate our approach, we used our simulator that allows us to run and measure the performance of our strategy of clustering *CD* (***D****ata based* ***C****lustering*) and compare it with the *LID* strategy [6]. As it is already mentioned in the section of related works, The *LID* is a one hop clustering that selects as cluster-head the node with the minimum *id* among its neighbors without taking into account other characteristics of system such as energy, mobility. The parameters of simulations are presented in Table 1.

Table 1: Parameters of simulations

Variable	Value
Total number of nodes	1-1000
Node mobility (%)	1-100
Node energy (%)	10-100
bandwidth (Mb/s)	20-100
range (m)	10-100
Total number of data	10-300
Total number of data per node	0-10
Data size (Mo)	1-10

In the first experiment, we studied the impact of the range on the number of clusters in both clustering approaches *LID* and *CD.* The range specifies the number of possible neighbors for each node. We studied the impact of this parameter on the number of clusters in the system. The results are shown in Figure 5. Increasing the range minimizes the number of clusters in the both approaches because it increases the number of neighbors per node. But our approach optimizes the number of clusters with a gain of 25% compared to the *LID* approach.

The number of nodes also affects the number of clusters. According to the results obtained in the second experiment (see Figure 6). Increasing the number of nodes increases the number of clusters, but not on the same frequency for both approaches. Our approach *CD* is better than *LID* approach with a gain estimated by 22.8%.

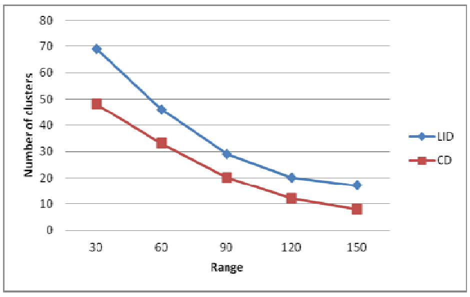

Fig5: Range vs Number of clusters.

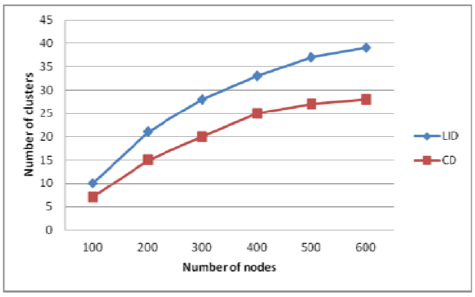

Fig6: Number of nodes vs Number of clusters.

The number of transitions is a very important parameter; it reflects the stabilization of the topology constructed by the clustering. The transition is the passage from configuration *i* to configuration *i* +1. The configuration is the result of the execution of at least one step of the clustering algorithm [2]. The results of our third experiment (see Figure 7) show that despite the increased number of nodes in random graphs we used, the number of transitions for stabilization in our algorithm *CD* varies very slightly, while still better with 30 % than the clustering approach *LID*, because *CD* is purely local and does not require that information obtained through the neighborhood *hello* messages. This ensures the scaling.

Minimizing response time increases system reliability. The results of the fourth experiment shown in Figure 8 prove that in the *LID* approach, the response time increases with increasing number of nodes and the time is greater compared to our approach because the data in cluster formed by *LID* are located randomly. The *CD* approach works better because the CH maximizes the number of data in its neighborhood which increases intra cluster access from access inter-cluster. The *CD* minimizes 43% of response time compared to *LID* approach.

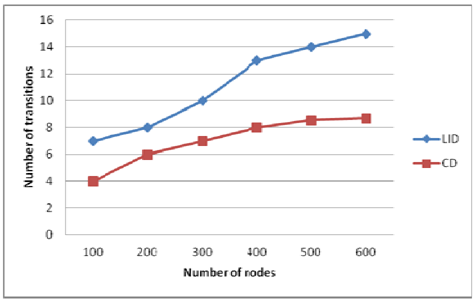

Fig7 : Number of nodes vs Number of transitions

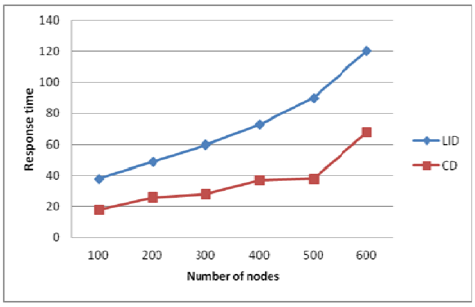

Fig8: Number of nodes vs response time

In the last experiment, we studied the impact of number of read requests to the energy consumed in the system. We note that the energy increases with increasing number of applications for both approaches because each query must be redirected to other nodes in different clusters to reach the answer. But our approach *CD* decreases remarkably the energy consumption (a gain of 25%) because it maximizes the number of different data in each cluster taking in the account the mobility and the energy of each node, which improves the intra-cluster access and stabilize the topology (see Figure 9).

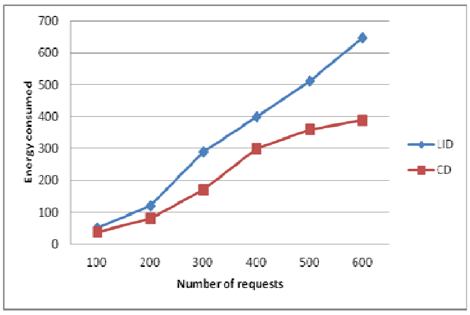

Fig9: Number of requests vs Consumption of energy.

6 Conclusion

In this report, we proposed a clustering strategy that maximizes intra-cluster access and minimizes energy consumption. This strategy uses energy, mobility and the types of data stored in the neighborhood to elect the cluster-head. The experimental results demonstrate the effectiveness of our proposal. Our strategy can be used to create clusters in unstructured P2P system as *FastTrack* and *Gnutella* [5], where the number of message and consumption of bandwidth are large. To avoid thus problems and increase the quality of services in requests management, we use our clustering algorithm where for each node u: $\frac{Energ_u}{Mob_u}=1$, so $\gamma=\|ListT_u\|$. Perspective, we offer extension work by taking into account the availability of links between the different neighboring data to improve system reliability.

References

1. A. Abbasi, M. Younis, *A survey on clustering algorithms for wireless sensor networks, Journal Computer Communications*. 30(14): 2826–2841, 2007.
2. D.J. Baker, A. Ephremides and J. A. Flynn. *The Design and Simulation of a Mobile Radio Network with Distributed Control*. IEEE Journal on Selected Areas in Communications, 2(1): 226–237, 1984.
3. S. Basagni, *Distributed Clustering Ad Hoc Networks*, In Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), pages 310-315, august 1999.
4. C-C. Chiang, H-K. Wu, W. Liu, M. Gerla, *Routing in Clustered Multi hop, Mobile Wireless Networks With Fading ChannelL*. IEEE Singapore international conference on networks, pages 197-211, 1997.
5. K. Eger, T. Ho_feld, A. Binzenhofer, and G. Kunzmann, *Efficient simulation of Large-Scale P2P Networks: Packet-level vs. Flow-level Simulations*, in Proc. UPGRADE-CN'07, HPDC Workshops, pages 1-16, USA, 2007.
6. J. W. A. Ephremides and D. J. Baker, *A design concept for reliable mobile radio networks with frequency hopping signaling*, Proceeding of IEEE, pages 53-76, January 1987.
7. T. Hara, *Replica Allocation Methods in Ad Hoc Networks with Data Update*, Journal Mobile Networks and Applications, 8(4): 342-354, 2003.
8. B-J. Ko and D. Rubenstein. *Distributed self-stabilizing placement of replicated resources in emerging networks*. IEEE/ACM Trans. Netw., 13(3): 476–487, 2005.
9. S-J. Lee, W. Su, J. Hsu, M. Gerla and R. Bagrodia, *A Performance Comparison Study of Ad hoc Wireless Multicast Protocols*. Proceedings of the IEEE International Conference on Computers and Communications, pages 565 – 574, March 2000.
10. B. Meroufel, G. Belalem: *Availability Management in Data Grid,* Lecture Notes in Electrical Engineering, 1, Volume 107, IT Convergence and Services, Part 1, Pages 43-53.
11. H. Taniguchi, M. Inoue, T. Masuzawa, H. fujiwara, *Clustering Algorithms in Ad Hoc Networks*, Electronics and Communications in Japan, Part2, 88(1): 51-59, 2005.
12. L. Yin and G. Cao, *Balancing the tradeoffs between data accessibility and query delay in ad hoc networks*, Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems, pages 289 – 298, 2004.
13. J. Zheng, J. Su, and X. Lu. *A Clustering based Data Replication Algorithm in Mobile Ad hoc Networks for Improving Data Availability*. In Proc 2^{nd} International Symposium on Parallel and Distributed Processing and Applications (ISPA 2004), Pages 399–409, 2004.