Mobile Cloud Management: A New Framework

Mohammad Javad Salehi[1] mjsalehi@ee.sharif.edu
Babak H. Khalaj[1] khalaj@sharif.ir
Marcos Katz[2] mkatz@ee.oulu.fi
Ghazal Fazelnia[1] fazelnia@ee.sharif.edu
Parishad Karimi[1] pkarimi@ee.sharif.edu

[1] Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran
[2] Department of Electrical Engineering, University of Oulu, Oulu, Finland

Abstract—**Smartphones can be viewed as resource pools capable of communicating with their outside world. Such communication capabilities provide unique opportunity for sharing resources, which would result in new possibilities for end users. In this paper, we present a new framework designed for managing mobile local networks, also referred to as Mobile Clouds. In this framework, mobility of nodes is managed by selecting one leader for each cloud, which is responsible for all cloud-level decisions, as well as processing all incoming data from other users. We also evaluate and discuss our proposed framework under different scenarios.**

Index Terms—**Mobile Local Networks; Mobile Clouds; Resource Sharing; Cloud Management.**

I. Introduction

It is becoming more and more common to integrate several air interfaces on each cellphone, not only for cellular network access, but also for short range communications such as Wi-Fi and Bluetooth communication technologies. Such short-range communication links on mobile devices provide great cooperation opportunity among these nodes. Throughout this paper, we will use the term **"Mobile Cloud"** for cooperative arrangement of mobile devices (e.g. mobile phones or other communication enabled portable devices). These devices can communicate with each other using short-range links, and furthermore can naturally connect to the overlay cellular networks while maintaining their local links. Such mobile cloud architecture allows formation of Resource Pools, where resources include radio resources as well as physical resources present on board (such as different sensors and actuators, processing power, etc.). The key point is that such resources can be shared, summed up (augmented) or relocated inside the cloud. In order to further clarify the resource sharing ideas in mobile clouds, consider the following few examples:

1) Several cellular links can be used simultaneously in order to increase the data throughput of one or more nodes of the cloud. For example, the well-known Distributed MIMO techniques can be created and used within the cloud to improve performance or increase coverage.
2) Imaging sensors of nearby devices could be shared in order to obtain 3D models of scenes, images with higher resolution, or for improving stabilization (reducing handshake effects) on final images.

In the meantime that the aforementioned examples demonstrate novel opportunities and potential research areas provided by the resource sharing concept, they also bring up a number of key challenges to be solved before such cooperative platforms can work properly in practice. For example, the first step in the resource sharing procedure is forming a local network, which can adapt to high mobility characteristics of devices (or nodes) participating in it. In addition, before starting the cooperation process, different devices should be aware of each other's presence and their capabilities and currently available resources. The final step would then be sharing resources, which will require techniques specific to each type of resource or application being shared. For example, camera sharing could exploit image processing techniques, whereas CPU sharing will more rely on grid or distributed computing methods. In this paper, we develop a novel framework for managing local networks formed with resource sharing in mind. Our proposed framework handles mobility of the nodes by choosing one leader for each cloud which is responsible for cloud-level decision making as well as collecting status data from all nodes present in the cloud.

II. Related Works

The main principles of resource sharing in mobile clouds are presented in [1] and [2]. In [3], a cooperative power saving strategy is proposed and it is shown that all participants benefit by cooperating. In [4] and [5], cooperative methods for multimedia applications are studied. In [6], cellular-controlled short-range communication links are studied and new methods for bandwidth and power allocation are presented. In [7] and [8], cooperative web browsing for mobile phones is considered and it is shown that cooperation can increase webpage download speed.

Computer grids also address resource sharing from another perspective, and self-organized grouping (SOG) protocols, presented in [9], are designed in such platforms, to improve query performance. In [10], space filling curves are added to SOG to further improve the query performance. We should also mention another key centralized resource sharing protocol in grid-type environment, the Matchmaking algorithm [11]. Despite popularity of Matchmaking algorithm for such environments, since it is designed for networks whose topology is not changing rapidly, it is not directly applicable to our mobile cloud scenario.

Leader selection in networks with mobile nodes is also studied in topics related to MANETs, some examples being [12], [13] and [14]. Our work is different as it focuses on the resource sharing concept in these networks rather than selecting a general leader, and hence all parameters are defined such that messaging between nodes is optimized and all events due to node mobility are handled in the best way.

III. Cloud Management Framework

We assume the mobile cloud has been formed, i.e. , nearby nodes have been connected to each other using short-range communication links. We also consider the following assumptions for the communication model between nodes:

1) Nodes connect to each other in the transport layer using connection-oriented protocols such as TCP. Hence, all routing and error-correction operations are hidden to nodes, and nodes face error-free channels when a route from source to destination is available.
2) All nodes in the network can connect to each other. The routing algorithm depends on the network type, i.e. short-range link being used, and ad-hoc or infrastructure mode of the network.

The proposed framework involves a centralized algorithm for managing the resource sharing operations. All nodes present in the cloud cooperate to select a unique leader, and after leader election they send status information to the leader regularly. The leader node is responsible for all cloud-level decisions, and furthermore should process the received data to find potentially suitable devices for resource sharing. There are no constraints on the mobility of the nodes, i.e. all nodes (including the leader node) may join or leave the local network at anytime. In addition, nodes present in the local network can decide whether or not to participate in the resource sharing cloud. This is accomplished simply by switching the resource sharing feature on or off on the device being used.

We will now explain each task in more detail.

A. Leader Election and Communication Management

As stated in section 2, self-organizing groups (SOG) algorithm proposed in [9] provides a robust tool for leader election and group management. However, as SOG is defined primarily for accelerating query operations in computer grids, we should still adapt it to manage mobility of the nodes for the resource sharing problem and address issues such as data management and merging different clouds. In this respect, for each node n, we define:

1) S; a parameter indicating the device resource status. Later, we will explain how to calculate S. Initially, the value of S is undefined.
2) W; set of all nodes in the cloud. Initially, W is equal to a set with just one member, n.
3) S_0; set of latest S values for all nodes in W. Initially, S_0 is equal to a set with just one member, S.
4) R; set of resource status. Initially, R is equal to an empty set, $\emptyset$.
5) L; cloud leader. Initially, L is equal to n.
6) C; total number of nodes in the cloud (the cloud size). Initially, C is equal to 1.

Note that the R set is composed of all resource values which are available in a device, and the S value is a parameter

TABLE I
Algorithm 1: Leader collision algorithm

1	**Get L(m). Suppose P=L(m).**
2	**Get S(P) and C(P) from node P.**
3	**If S<S(P)**
4	Send *JoinGroup* message to P.
5	Send C, W, S_0 and R_0 to P.
6	Get L, C, W and S_0 from P.
7	**Else**
8	Send *DeclareAsLeader* message to P.
9	Get C(P), W(P), S_0(P) and R_0(P) from P.
10	Set C=C+C(P), W=W+W(P), S_0= S_0+ S_0(P), and R_0=R_0+R_0(P).
11	Send L, C, W and S_0 to all nodes in W.
12	**End if**

TABLE II
Algorithm 2: Old Leader operations when receiving DeclareAsLeader message from node n

1	**Send C, W, S_0 and R_0 to n.**
2	**Get L, C, W and S_0 from node n.**

TABLE III
Algorithm 3: Leader operations when receiving JoinGroup message from node n

1	**Get C(n), W(n), S_0(n) and R_0(n) from node n.**
2	**Set C=C+C(n), W=W+W(n), S_0=S_0+S_0(n) and R_0=R_0+R_0(n).**
3	**Send L, C, W and S_0 to all nodes in W.**

averaging these values.We also define the following parameter only for leader nodes:

1) R_0; set of latest R sets for all nodes in W. Initially, R_0 is equal to a set with just one member, R.

Suppose we have one mobile node which turns the resource sharing feature on. This node forms a single-node cloud initially, and declares itself as the leader. After that, it periodically searches for nearby nodes which are taking part in some resource sharing cloud. When such node m is found, n runs the leader collision algorithm as shown in Table I to join the group. Note that algorithm can be used to solve collisions between two groups of any given size, as will be explained in more detail later.

In the case of a single node n aiming at joining an existing cloud, if node n decides to become the new cloud leader, it sends the *DeclareAsLeader* message to the old leader, otherwise it sends the *JoinGroup* message. In both cases, all nodes in the final cloud would be informed of the new cloud structure, i.e. the new leader, list of all nodes in the cloud, and summary of all participating nodes status. The leader performs algorithms 2 and 3 with respect to *DeclareAsLeader* and *JoinGroup* messages, which are shown in Tables II and III respectively.

Again, both algorithms 2 and 3 might be applied when two groups of any size merge. After node n joins an existing cloud, we have a new cloud with size C and leader L. All

TABLE IV
ALGORITHM 4: REGULAR OPERATIONS OF ALL NODES IN THE CLOUD

1	**Update R and S using device resource information.**
2	**If this is the leader node**
3	**If there exists any s in S_0 such that S<s-T**
4	Send ChangeLeader message to node n having greatest S value.
5	Send R_0 to node n.
6	Get L, C, W and S_0 from node n.
7	**End if.**
8	**If set C and W has been changed recently**
9	Send new C and W to all nodes in W.
10	**End if**
11	Send S_0 to all nodes in W.
12	**Else**
13	Send new information to L (will be further explained in part B).
14	**End if.**

TABLE V
ALGORITHM 5: REGULAR NODE N OPERATIONS WHEN RECEIVING CHANGELEADER MESSAGE

1	**Get R_0 from old leader.**
2	**Set L=n.**
3	**Send L, C, W and S_0 to all nodes in W.**

cloud-level decisions such as leader replacement and merging the current cloud with another one are made by leader till leader is connected to the network. Other nodes in the cloud just send their resource information and status parameter S to the leader regularly. Algorithm 4, shown in Table IV, shows regular operations performed by all nodes in the cloud. Parameter T in algorithm 4 is a threshold used to increase group stability. Algorithm 4 states that all nodes only send their status information to the leader on a regular basis and all other operations are performed by the leader. These additional operations performed by the leader node include monitoring the entire cloud to determine whether there is any better node to become the new leader or not. The monitoring is performed by comparing the current leader's status value S_L with all other nodes status values. If some node n is found with the S_n value greater than S_L+T, n should become the new leader by using the *ChangeLeader* message. As n is currently submitted to the cloud, it already has W and S_0 sets and hence the old leader just needs to send the R_0 set to n. Node n performs algorithm 5 after receiving the *ChangeLeader* message to inform all nodes of its new leadership in the cloud, as shown in Table V.

The other important task which is regularly performed by the leader node through algorithm 4, is informing all cloud nodes of recent changes in the cloud structure. Changes in the cloud structure occur, for instance, when some nodes turn the resource sharing feature off, or get disconnected from the network (changes also occur when new nodes join the cloud). In such cases, C and W will change, and the leader should inform all remaining nodes of these changes in the network

TABLE VI
ALGORITHM 6: NODE N OPERATIONS WHEN L NODE IS MISSING AND THERE IS NO NEW LEADER NODE SELECTED.

1	**Search S_0 set to find its greatest value S_g corresponding to node G in W set.**
2	**If n=G**
3	Set L=n and C=1.
4	Set W, S_0 and R_0 to empty sets.
5	**Else**
6	Set W, S_0 and R_0 to empty sets.
7	Send JoinGroup message to node G.
8	**End if**

structure. However, it is important to note what happens if the leader node itself gets disconnected from the cloud? This event would happen in the following two cases:

1) Leader node L turns the resource sharing feature off. In this case, L selects the new leader which has the greatest S value among S_0 set and sends *ChangeLeader* message to it. L performs the required operations before getting disconnected from the cloud.
2) The leader node L gets disconnected from the network. In this case, there is no time for node L to select the new leader, and the new leader selection should be handled by other nodes in the cloud themselves. One approach to address this issue is to start with each node declaring itself as a one-member cloud and searching for nearby clouds. However, generally, this is not a wise approach since all nodes in the cloud currently have W and S_0 sets which are valuable information and can help reduce the time and messaging overhead needed. We propose algorithm 6 to address this problem, as shown in Table VI.

Another issue of concern is what happens if one node sends the *JoinGroup* message to another node which is not a cloud leader yet? In this case, the receiver node forwards the message to its own leader who is responsible for cloud-level decisions, and the leader node runs algorithm 3 for joining this new node to the cloud.

Using algorithms 1-6, new nodes can join existing clouds and all events caused by node mobility would be handled.We study one more event which happens when two cloudswith more than just one member intend to merge. In this case, the joining process will be successful if just one of the cloud leaders runs algorithm 1. Through this algorithm, collision between two leaders is resolved and one of them would become the new leader.

After introducing our cloud management framework, we will explain how to calculate the S value in more detail.

B. Averaging Resource Values

As mentioned earlier, we use the S parameter for averaging all resource values in each device. Now the question is how to calculate this average value? In other words, what principle is used for calculating the overall status of each device? In order to answer this question, we should take the following considerations into account:

1) S represents an overall summary of all resources in the device. So, at least one resource value should be used in calculating S.
2) S parameter is mainly used for leader selection in the cloud. Therefore, resources which are more important for leader operations should have more effect on the S value.
3) Leader replacement in the cloud involves time and messaging overhead which decreases overall network performance. One parameter which is directly related to cloud stability is *Node Connectivity*, defined as the numerical average of achievable throughputs between one node and all other nodes in the cloud. However, as all short-range networks do not provide such information, we use Node Connectivity just when we can get necessary information from the underlying short-range network.

We propose the following linear equation for calculating S. A linear relationship is used because of its simplicity:

$$S = \alpha_1 r_1 + \alpha_2 r_2 + ... + \alpha_n r_n \tag{1}$$

$$\sum_{n=0}^{n} \alpha_i = 1$$

In (1), the values of α_is are non-negative coefficients and r_is are resource values normalized with respect to some predefined values. Exact coefficient and predefined values, as well as resources being utilized are known to all cloud devices according to protocol version used for communication. Using α_is, we can assign different priorities for each resource. We can also take Node Connectivity into account while calculating S with (1). This is accomplished easily by setting one of r_is equal to Node Connectivity.

IV. Performance Analysis

After describing the proposed framework for mobile cloud management, we will evaluate its performance. The analysis is carried out in the following two ways:

1) We derive analytical equations under specific conditions and investigate the effect of different parameters on system performance.
2) We use simulation results for investigating system behavior under more general conditions.

All simulations are performed using our own developed simulation platform for mobile clouds, which uses Nokia Qt framework as its programming environment. Using Qt framework enables us to deploy our final algorithm on Nokia smartphones, as well as providing us with unique features, such as perfect Signal-Slot mechanism for event handling (even in multi-threaded applications), platform-independent coding, and easy user interface development.

A. Analytical Results

First, we analyze the number of messages needed to be transmitted over the local network in order to reach just one perfect leader. As we mentioned earlier, each node turning on the resource sharing feature forms a new cloud with just one member, and tries to join it with nearby clouds immediately. We assume that the cloud remains unchanged during operation, that is, no node leaves or joins the cloud (the cloud size remains constant). We also assume that some route exists between each pair of nodes. At the first step, lets suppose that two clouds with M and N users aim at combining with each other. Node m from cloud M finds node n which is a member of cloud N. Assuming that neither m nor n are cloud leaders (which is the worst case), the following messages are transmitted till these two clouds join together:

1) Node m asks n about its leader and n responds: 2 Messages
2) Node m asks its own leader to run leader collision algorithm: 1 Message
3) Lets show ms leader as L(m) and ns leader as L(n). L(m) asks L(n) to provide him with its own S value, and L(n) responds: 2 Messages
4) L(m) runs leader collision algorithm:
   a) If L(m) must become the new leader, it sends L(n) *DeclareAsLeader* message, then gets C, W, S_0 and R_0 parameters from L(n) in one message, and finally sends new L, C, W and S_0 parameters to all M+N-1 other nodes in the new cloud: 2+M+N-1 messages.
   b) If L(n) must become the new leader, L(m) sends him the *JoinGroup* message, followed by one message containing C, W, S_0 and R_0 parameters. Then, L(n) sends new L, C, W and S_0 parameters to all M+N-1 other nodes in the new cloud: 2+M+N-1 messages.

Consequently, if we define J(M,N) as the total number of messages required for joining two groups of sizes M and N, we have:

$$J(M, N) = 2 + 1 + 2 + 2 + M + N - 1 = 6 + M + N \tag{2}$$

Now, if we suppose there are 2^u total nodes, which form 1-node clouds in the registration step, 2-node clouds in the first step, 2^2-node clouds in the second step, etc.; and define R(i) as the total messages needed for forming one 2^i-node cloud in the i-th step, we have:

$$R(i) = R(i-1)+R(i-1)+J(2^{i-1}, 2^{i-1}) = 2R(i-1)+6+2^i \tag{3}$$

and as:

$$R(0) = 0 \tag{4}$$

we will have:

$$R(i) = 6(1 + 2(1 + 2^{i-2})) + \sum_{k=2}^{i} 2^{k-2} \times 2^k \tag{5}$$

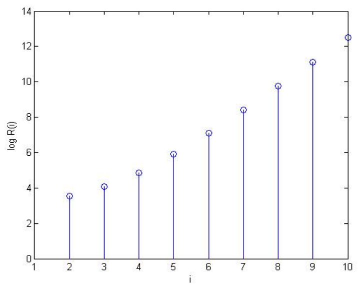

Fig. 1. Total messages needed to form 2^i node cloud

Although (5) is only valid under perfect assumptions, it can provide some insight on how many messages are needed for forming mobile clouds. In Figure 1, we have plotted (5) as a function of i for i=1 to 10. As can be seen from Figure 1, number of messages needed for leader election grows exponentially as the group size increases. However, as we have developed our algorithms mainly for clouds for a small number of nodes, this issue will not be of major concern. For example, i=10 refers to a network with 1024 nodes, which is not a typical value for local mobile networks based on short-range links such as Bluetooth.

Subsequently, we take a different approach and try to calculate how many messages are needed to be transmitted over the local network when the leader node leaves the cloud. As mentioned earlier, there are two possible scenarios for this event (Assume that there are a total of N nodes in the cloud):

1) The leader node turns the resource sharing feature off. In this scenario, leader node finds the node featuring the greatest S value in the cloud, and sends one message asking it to become the new leader. The target node sets itself as the new cloud leader and transmits its new leadership information to all N-2 other nodes. Therefore, if we K_1(N) as the total number of messages needed for leader election in scenario 1, we will have:

$$K_1(n) = 1 + N - 2 = N - 1 \qquad (6)$$

2) The leader node gets disconnected from the network. In this scenario, all nodes in the cloud search for node G with the greatest S value in their S_0 set, and send *JoinGroup* message to it. Assuming the perfect condition in which all nodes find the same node G, we would have N-2 nodes sending *JoinGroup* message to node G, and node G would inform all currently-joined nodes of the new cloud structure after receiving each message; and hence K_2(N), the total number of messages needed for leader election in scenario 2, would be equal to:

$$\begin{aligned} K_2(N) &= N - 1 + (1 + 2 + \cdots + N - 2) \\ &= N - 1 + \frac{(N-2)(N-3)}{2} \end{aligned} \qquad (7)$$

In Figure 2, we have plotted K_1(N) and K_2(N) as a function of cloud size, N. As can be verified in Figure 2, the number of

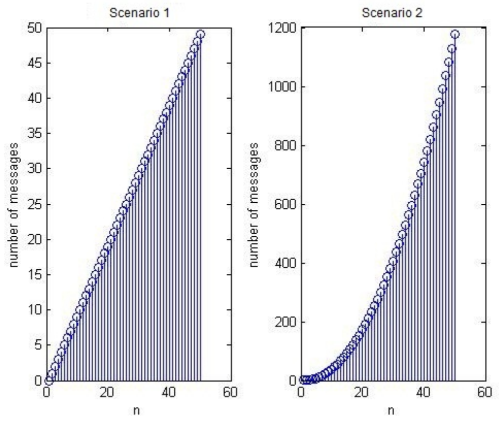

Fig. 2. Total number of messages needed for leader replacement

TABLE VII
SUMMARY OF FORMULAS

$J(M, N)$	**Total number of messages needed for joining two groups of size M and N.**	$6 + M + N$
$R(i)$	**Total number of messages needed for leader selection in the clouds of size 2^i**	$R(i) = 6 \times 1 + 2 \times (1 + 2^{i-2})) + \sum_{k=2}^{i} 2^{2k-2}$
$K_1(N)$	**Total number of message needed for leader reselection when leader node turn the resource sharing feature off**	$N - 1$
$K_2(N)$	**Total number of messages needed for leader reselection when leader node gets disconnected from network**	$\frac{1}{2}N^2 - \frac{3}{2}N$ $+2$

messaging needed in scenario 2 is more than what is needed in scenario 1. Scenario 2 can be further optimized if we have some kind of delay in node G; i.e. if node G sends cloud structure update messages at equal time intervals rather than only after receiving each *JoinGroup* message. In any case, since messaging overhead is not significant even for large values of N (such as N=50) corresponding to very large mobile local networks, we assume current amount of optimization would be enough.

We have summarized all equations in Table VII.

B. Simulation Results

As stated earlier, we have used our own simulation environment for evaluating the performance of the proposed framework. This simulation environment is written based on the Nokia's Qt framework, and uses novel features and tools provided by Qt. The developed environment is still under development and we intend to use it as a tool for evaluating the group-management algorithms in mobile environments. We make use of our simulation environment for the following two evaluations:

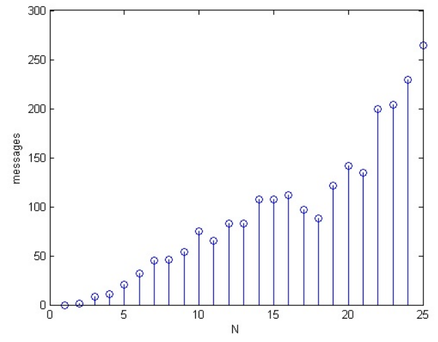

Fig. 3. Total number of leader collision messages needed

1. Does our leader election algorithm converge?
2. How much messaging is needed for leader election?

For such evaluations, we assume N nodes, where N varies between 1 and 100, and run the simulation for all $m \times m$ areas, where m is equal to 10, 20, 30, ... and up to 100 meters. We assume nodes have random movement, and there is a short-range link between two nodes as long as their distance is smaller than 20 meters. Also, nodes can send data to each other using multi-hop routing. The S value for each node is chosen randomly and changes slowly in time. In all our simulations, our algorithm converged in reasonable number of steps, and the total steps needed for convergence were mostly less than 20 steps. In addition, we measured the number of times the leader collision algorithm was needed, for N=1 to 25 and in the 40×40 m^2 area. The results are plotted in Figure 3, which shows an almost linear relationship between the two parameters.

V. Conclusion and Future Work

In this paper, we introduced our framework for cloud management in mobile networks. We proposed novel algorithms for leader election, cloud merging, and handling unexpected conditions. We evaluated the framework with respect to different parameter values, and demonstrated its feasibility for small networks. We also used our own simulation environment to evaluate the convergence behavior of the proposed algorithm. A number of ideas are still under consideration to improve the performance of the proposed algorithm. One important optimization step is to calculate S such that the best performance is achieved. Other optimization steps are related to issues such as determining the best timing for data transmission and processing, i.e. how to send status information to the leader node regularly such that all data is sent with minimum amount of overhead. Also, proper algorithms should be defined for leader node to process the received data and find potentially suitable devices for resource sharing.

Inter-cloud resource sharing is another area which can provide lead to exciting new capabilities on cellphones. For example, in scenarios where some people watch one soccer match in the stadium and use their cellphone cameras to record the scenes, such persons can form one local cloud and share the recorded material to obtain one high-resolution 3D movie, and then share this movie with other people outside-the stadium using long-range connection links such as 3G. Naturally, sharing resources between different clouds requires new protocols to be defined, as well as new problems to be solved. Incentive algorithms would encourage mobile users to share their resources over the cloud, as well as better tools for leader selection. For example, if the leader gains *Points* for good leadership and loses points when its performance is not acceptable, such reward-penalty history could be used for better leader detection and malware protection.

Acknowledgment

This work was supported in part by Iran Telecommunications Research Center (ITRC).

References

[1] Qi Zhang, H.P. Fitzek, Marcos Katz, *Evolution of Heterogeneous Wireless Networks: Towards Cooperative Networks*, 3rd International Conference of the Center for Information and Communication Technologies (CICT) - Mobile and wireless content, services and networks - Short-term and long-term development trends, 2006

[2] H.P. Fitzek, Marton V. Pederson, Marcos Katz, *A Scalable Cooperative Wireless Grid Architecture and Associated Services for*, Future Communications, European Wireless 2007, 2007

[3] Federico Albiero, H.P. Fitzek, Marcos Katz, *Cooperative Power Saving Strategies in Wireless Networks: an Agent-based Model*, Wireless Communication Systems, 2007. ISWCS 2007. 4th International Symposium on, 2007

[4] Federico Albiero, Marcos Katz, H.P. Fitzek, *Energy-Efficient Cooperative Techniques for Multimedia Services over Future Wireless Networks*, Communications, 2008. ICC '08. IEEE International Conference on, 2008

[5] Qi Zhang, H.P. Fitzek, Marcos Katz, *Cooperative Power Saving Strategies for IP-Services Supported over DVB-H Networks*, Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE, 2007

[6] H.P. Fitzek, Marcos Katz, Qi Zhang, *Cellular Controlled Short-Range Communication for Cooperative P2P Networking*, Wireless Personal Communications: An International Journal, Volume 48 Issue 1, January 2009

[7] Gian Paolo Perrucci, H.P. Fitzek, *Cooperative Web Browsing for Mobile Phones*, International Symposium on Wireless Personal, Multimedia Communications (WPMC'07), 2007

[8] G. P. Perrucci, F. H. P. Fitzek, Q. Zhang, Marcos Katz, *Cooperative Mobile Web Browsing*, EURASIP Journal on Wireless, Communications and Networking, Volume 2009, January 2009, 2009

[9] Anand Padmanabhan, Shaowen Wang, Sukumar Ghosh, Ransom Briggs, *A Self-Organized Grouping (SOG) Method for Efficient Grid Resource Discovery*, GRID '05 Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, 2005

[10] Anand Padmanabhan, Sukumar Ghosh, Shaowen Wang, *A Self-Organized Grouping (SOG) Framework for Efficient Grid Resource Discovery*, Journal of Grid Computing, Volume 8, Number 3, September 2010

[11] Rajesh Raman, Miron Livny, Marvin Solomon, *Matchmaking: Distributed Resource Management for High Throughput Computing*, HPDC '98 Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing, 1998

[12] M. M. Rahman, M. Abdullah-Al-Wadud, O. Chae, *Performance analysis of Leader Election Algorithms in Mobile Ad hoc Networks*, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.2, February 2008

[13] N. Malpani, J.L. Welch, N. Vaidya, *Leader Election Algorithms for Mobile Ad-hoc Networks*, Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications, DIALM 2000

[14] O. Dagdeviren, K. Erciyes, *A Hierarchical Leader Election Protocol for Mobile Ad hoc Networks*, IEEE Conference on Computational Science, ICCS 2008