A New Approach on Cluster based Call Scheduling for Mobile Networks

P. K. Guha Thakurta
Department of CSE, NIT, Durgapur-713209, India
Email: parag.nitdgp@gmail.com

Saikat Basu[1], Sayan Goswami[2], Subhansu Bandyopadhyay[3]
Department of Computer Science, Louisiana State University, USA[1] ,
Sapient Global Markets, India[2]
Department of CSE, University of Calcutta, Kolkata-700009, India[3]
Email: sbasu8@lsu.edu[1], sayan.nitd@gmail.com[2], subhansu@computer.org[3]

Abstract— **An efficient cluster based approach on call scheduling in mobile networks is proposed in this paper. The dynamic threshold value (τ) is formulated with justification. The clusters are formed on the basis of defined threshold value. The cluster head has been selected with respect to different weight metrics for improving call scheduling. The leader (cluster head) election procedure is also described for link breakage and link emergence respectively. After the formation of clusters, the subsequent call scheduling algorithm has also been outlined in detail in this work.**

Index Terms—**Mobile computing, clustering, call scheduling, Routing, Dynamic thresholding.**

I. INTRODUCTION

The rapid growth of cellular telephony needs efficient resource allocation strategies. Hence, an effective call selection procedure is also required at the same time. During network congestion, Call Admission Control (CAC) strategy is used to give permission to limited number of users as well as deny service for rest of the users [3]. Consequently, Quality of Service (QoS) becomes an important factor for admitted users. It is therefore necessary to consider two near contradictory requirements – allocating resources as well as ensuring Quality of Service (QoS) when all users are trying to make a request at the same time.

Nodes communicate with each other using multi-hop links in mobile cellular networks. Each node in the network has call forwarding capability to other nodes. So, various routing strategies [11] have been designed to address the problem of finding the routing path. The cluster based routing protocol proposed in [9] assumes that the mobile nodes are location-aware. The procedure behind the foundation of location different neighbor's location with respect to a specific node is beyond the scope of this work. One of the most important parameters to be considered for leader election in a cluster is the congestion metric. However, a limitation in the proposed routing protocol in [10] has been observed regarding this issue. An efficient call scheduling procedures known as Priority based Tree Generation for mobile networks (PTGM) in [1] has described a tree based methodology with the foundation of unique path sequence. This tree based call scheduling procedure has been mapped into Cartesian coordinate system, with mobile terminal (MT) placed at the origin (0,0). Hence, other cells are represented as points in the coordinate system with following certain criteria [2]. This coordinate based routing protocol (CSTR) has been formulated with the help of a tree structure and all possible routing paths could also be enumerated in a simple manner. This routing path analysis needs a more efficient methodology to increase throughput and reduce network latency at the same time.

In this paper, a new constraint based spatial clustering algorithm has been designed for call scheduling. This algorithm is based on a dynamic threshold value. This threshold value has been formulated with respect to the Euclidian distances between the cells of the coordinate based system representation in [2]. On the basis of threshold value, the different clusters of cells could be formed and hence, a new clustering algorithm is proposed. The positional identity (x,y) of each cell is broadcast to all other cells in the network. Consequently, it causes congestion known as Broadcast Storm. To prevent it, a new multicast clustering algorithm is also proposed in this paper. Once the clusters have been formed, the leader election in a cluster is done with a weight based algorithm to reduce the searching complexity to a large extent. This weight is quantified with respect to different performance metrics. Due to the emergence of new nodes or disappearance of the existing ones in/from the network, the cluster leader needs to be updated. Hence, two algorithms named as *link_emergence* and *link_breakage* have been proposed. The formation of clusters and call scheduling through the clustered nodes have been discussed in detail. The simulated result of performance analysis for the system is also shown in terms of QoS of the network.

doi:10.4304/jait.3.3.184-190

The rest of this paper is organized as follows. In section 2, a brief description of coordinate based routing protocol (CSTR) is provided for completeness of the work. The different issues of the proposed model are described with various algorithms in section 3. The experimental results are discussed in section 4. The section 5 concludes the advantages of the proposed model with future scope.

II. COORDINATE BASED SEARCH TREE GENERATION WITH THE DETERMINATION OF ROUTING PATHS (CSTR) [2]

In this model, MT is denoted by (0,0) and each other cell is having a coordinate of the form (x,y). The cells covered by the radius r (within transmission covering range [3]) of MT are mapped as (x,y+1), (x+1,y+1) and (x+1,y) such that $x+1 \leq r$ and $y+1 \leq r$. For example, in Fig. 1(a), cell numbers C_{13}, C_{12} and C_{11} of r = 1 in [1] are mapped into (1,0), (1,1) and (0,1) respectively. Therefore, cellular structure of mobile networks detected in [1] could be mapped into a coordinate based system as shown in Fig. 1(b).

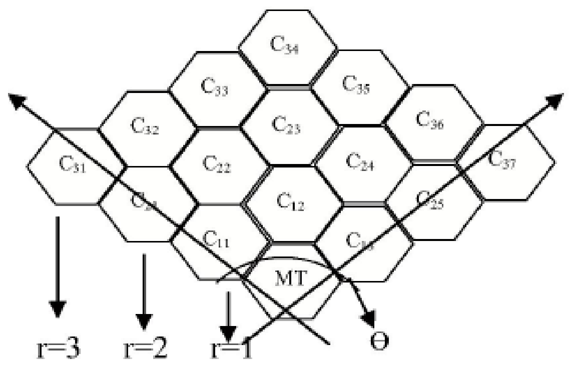

Fig. 1(a): Cellular structure for Mobile Networks for r=3

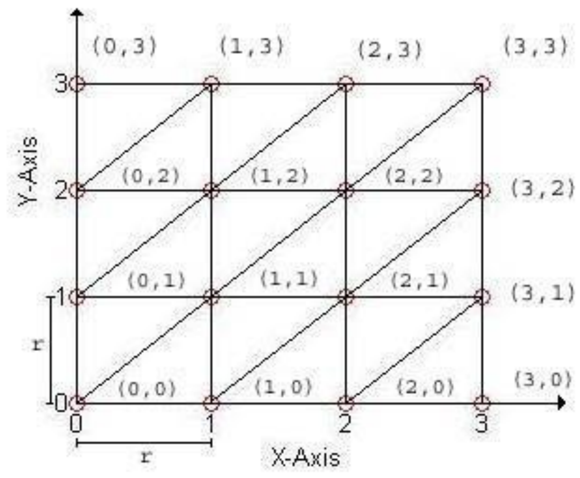

Fig. 1(b): Coordinate based representation of Fig. 1(a) (in [2])

III. THE PROPOSED MODEL

The model proposed in this paper is the collection of several functional events in a sequential manner. These are listed as follows:

(i) Determination of threshold value;
(ii) Formation of Constraint based Spatial Clustering Algorithm and identification of broadcast storm;
(iii) Prevention of broadcast storm by using multicast clustering algorithm;
(iv) Introduction of weight metric and subsequent proposal of a leader election algorithm for the reduction of searching complexity;
(v) The necessity of dynamic link handling using weighted mobility-adaptive leader election algorithm;

A. *Determination of Threshold Value*

The threshold value selection for a cluster based system reflects the number of clusters obtained. If the threshold value is very small, then there would be a high number of clusters and consequently each connected component size becomes quite small. So it results in a low throughput. On the other hand, the reverse situation would occur for the high threshold value. This leads to high congestion and latency. The threshold values could be defined as follows:

$$\text{Threshold } (\tau) = \begin{cases} \frac{D}{N}, \mathit{for\ cells\ belonging\ to\ the\ same\ r} \\ \frac{\min + max}{2} \qquad\qquad , \mathit{otherwise} \end{cases}$$

where, *D* denotes the distance between the two farthest cells in radius *r*,
N denotes the total number of cells in radius *r*,
max and *min* denotes the maximum and minimum Euclidian distances between cells belonging to radii i and j respectively.

Considering Fig. 1(a) and Fig. 1(b), the threshold value would be 0.56 for cells belonging to radius r = 2 and 1.618 for that of r = 1 and r = 2.

B. *Constraint Based Spatial Clustering Algorithm*

Once the threshold value is determined, the clustering process [12] is initiated. Here, the metric used is the positional identity (x,y) of the cell, where x denotes the position of the cell along the X-axis and y denotes its position along the Y-axis. Each cell advertises its positional identity to adjacent cells in the form of broadcast packets. It is performed in the network as shown in Fig. 1(b). Now, the difference between the positional identity values of the two cells is compared with the predefined threshold values. If it is less than the threshold value, an edge is constructed between the two vertices (cells). Otherwise, no edge exists between these two cells. Hence, a number of connected components (clusters) are obtained, maintaining the tradeoff between throughput and latency. This procedure is described by the following algorithm.

Algorithm:

```
begin
```

```
  Set the threshold value $\tau$
  for each cell i
      positional identity$_i$ $\leftarrow$ ( $x_i$, $y_i$ )
      BROADCAST_PACKET ( positional identity$_i$ ) /* sends a broadcast packet to all of its neighbors */
  end for
  for each cell j
      while RECEIVE_PACKET( ) is not NULL
          ($x_i$, $y_i$) $\leftarrow$ positional identity$_i$
          diff $\leftarrow$ SQRT (($x_i - x_j)^2 + (y_i - y_j)^2$)
          if diff < $\tau$
              construct an edge $e_{ij}$ between $v_i$ and $v_j$
          else
             continue
        end if
     end while
  end for
  end
```

The various functions used in the above algorithm are:

1) BROADCAST_PACKET( ) – Advertises the positional identity of a cell to all the adjacent cells in the network.
2) RECEIVE_PACKET( ) – A cell receives a packet from an adjacent cell using this routine.

A major problem has been identified in the above algorithm known as broadcast storm. Due to the broadcasting of packets, it may happen that there would be the possibility of multiple copies of the same packet reaching any particular cell. Subsequently, it causes network congestion.

C. *Prevention of Broadcast Storm*

A multicast clustering algorithm has been proposed to prevent broadcast storm. The key feature of this algorithm is that – sending multicast packets to the neighbors except the one from which it receives a multicast packet. This prevention procedure is described by the following multicast clustering algorithm.

Algorithm:

```
begin
Set the threshold value $\tau$
for a particular cell i
   positional identity$_i$ $\leftarrow$ ( $x_i$, $y_i$ )
   MULTICAST_PACKET ( positional identity$_i$ ) /* sends a multicast packet to all the neighbors but one cell at a time */
end for
for each cell j $\epsilon$ NEIGHBOR(i)
   while RECEIVE_PACKET( ) is not NULL
       ($x_i$, $y_i$) $\leftarrow$ positional identity$_i$
       diff $\leftarrow$ SQRT (($x_i - x_j)^2 + (y_i - y_j)^2$)     /* SQRT returns the square root */
       if diff < $\tau$
           construct an edge $e_{ij}$ between $v_i$ and $v_j$
       else
           continue
       end if
           MULTICAST_PACKET ( positional identity$_j$ ) /* repeat the process for each neighbor */
   end while
end for
end
```

The functions used:

1) MULTICAST_PACKET( ) - Advertises the positional identity of a cell to all the adjacent cells in the network except the one from which it itself received the multicast packet.
2) RECEIVE_PACKET( ) – A cell receives a packet from an adjacent cell using this routine.

D. *A New Approach towards Leader Election*

Once the clusters have been formed, the next step is to design an efficient leader election algorithm. Leader election is a fundamental control problem in both wired and wireless systems [4]. A leader is required in a group communication system to handle the transmission of messages to the members of the group. Here, it is represented by a cluster-head of a cluster. The dynamic cluster-head selection is required for the constant change of the network topologies. The weights (W) are assigned to each member of the cluster. The maximum weighted member is to be considered as the cluster-head. These weights are dependent on the following parameters.

1) Degree of a vertex (cell): This denotes the number of cells (D) connected to that specific cell. If the degree of the cell is higher, then it has a greater probability of being elected as a cluster-head due to nearest neighborhood principle [5].

2) Mean of the distances: This parameter (δ) represents the mean of the distances of a cell from its neighbors. If the value of δ is small, then its probability of being elected as a cluster-head is higher.

3) Counter values associated with each cell (C): The concept of counters was introduced in [6]. The counter values are increased with respect to time and provide a measure of the congestion level along a specific path. Naturally, higher counter value of a particular cell means high congestion. Subsequently, it has a lower probability to be elected as the cluster-head.

Once all the parameters have been quantified, the weight (W) can be defined as follows.

$$W \propto \frac{D}{\delta \, . \, C}$$

The leader election algorithm runs recursively and eventually terminates after electing a unique cluster-head for each cluster. This algorithm is comprised of the following procedures.

ELECT_HEAD ( ) – Any cell which has no cluster-head begins a diffusing computation [7] and calls this procedure. This in turn sends the elect_head message to all its neighbors.

RECV_MSG ( ) -- When a cell receives an elect_head message through the RECV_MSG ( ) procedure, it sets the sending cell as its parent.

SEND_NODE ( ) -- It returns the cell (C_{ij}) which sent the last message to the receiving cell.

SEND_ACK ( ) -- If a cell receives an elect_head message from a non-parent cell, then it returns an ACK message to the cell. This message contains the information about the current best valued cell and its identity in the cluster. This is a recursive procedure which continues until a unique leader is chosen for the cluster.

SEND_NAK ( ) -- This procedure is called when a cell has received an elect_head message from another cell that is designated as its parent.

BROADCAST_PACKET ( ) – When the cell initiates the diffusing computations, subsequently it learns about the leader. Then, it sends a broadcast packet to all other cells belonging to the cluster using this procedure.

On the basis of the defined procedures, an algorithm is described for the cluster-head selection problem considering the highest weighted node as follows.

Algorithm:

```
Leader_selection( )
begin
for each cluster $C_i$ do
    for each node $A_i$ in $C_i$ do
        while CLUSTER_HEAD($A_i$) is NULL
            ELECT_HEAD ( $A_i$ )
        end while
    end for
    for any node $A_j$ in $C_i$
        if RECV_MSG ( ) == elect_head /* elect_head message sent by the ELECT_HEAD ( ) procedure */
            if SEND_NODE ( ) == PARENT($A_j$) /* indicates the parent of $A_i$ in the spanning tree */
                SEND_NAK ( PARENT ($A_j$) )
            else
                SEND_ACK (recv_node) /* indicates the node from which the elect_head message was received by the node $A_j$ */
            else if for all nodes $\in$ CHILD($A_j$)
                RECV_ACK ( ) == true
                then SEND_ACK ( PARENT ($A_j$) )
            end if
        end if
    end for
    if for all nodes $A_j \in$ CHILD($A_i$) /* CHILD ($A_i$) returns the children of the node $A_i$ in the spanning tree */
        RECV_ACK ( ) == true
        then BROADCAST_PACKET ( leader) /* leader indicates the most valued node based on the weight */
    end if
end for
end
```

E. *Weighted Mobility Adaptive Leader Election*

The leader election problem defined in the previous section has been described from the static point of view. The weights of each node are to be calculated during link breakage (i.e., when one or more nodes are detached from the cluster) or link emergence (i.e., when new nodes are added to the cluster). Subsequently, the leader election algorithm is updated, considering these two cases.

1) Link Breakage: If a link fails, then the nodes connected through the link is detached from the cluster. In the proposed algorithm, two new message packets namely *ping* and *response* are introduced. All nodes keep on sending *ping* packets to adjacent nodes to check the condition of their links. Whenever a node receives a *ping* packet, it replies with a *response* packet. So, once a node has sent a *ping* packet and has not received a *response* from a particular node, this means the link has failed. Consequently, the nodes participate in a leader election mechanism to choose a new leader. This process is described by the following algorithm.

Algorithm:

```
begin
    for each node $A_i$ in $C_i$
        for every other node $A_j$ in $C_i$
            SEND_PACKET ( ping )
        end for
    end for
    for each node $A_i$ in $C_i$
        if RECEIVE_PACKET ( ) == response
            continue
        else
            Leader_selection ( ) /* calls the main Leader selection algorithm for determining the new leader */
        end if
    end for
end
```

The procedures used in the above algorithm are:

1) SEND_PACKET(ping) – Send a ping packet to adjacent nodes to check the condition of the link.
2) RECEIVE_PACKET ( ) - Receives a packet from adjacent nodes.
3) Leader_selection ( ) – The main leader election algorithm proposed in section III.D.

2) *Link Emergence:* When a new cell site appears in the network, then a new base station is added to include the cell site in the cellular network. Then a new link comes up that connects the Base Station to the network. In this situation, two cases may occur. Either the new node(cell) has a weight less than that of the current cluster-head or greater than that. If the weight of currently arrived node is greater than the cluster head, then a packet with its own value and the link information is broadcast to the others in the cluster. Accordingly, the cluster head is updated and subsequently, all nodes of the cluster are informed of the changes. Otherwise, the new node is simply registered in the cluster. This process is described by the following algorithm.

Algorithm:

```
begin
    for each node $A_i$ added to $C_i$
```

```
        if weight_{Ai} < weight_{clusterhead}
            $C_i \leftarrow C_i \cup \{A_i\}$
        else
            BROADCAST_PACKET ( leader, $A_i$ )
    /* broadcasts a packet declaring itself as the leader and passing its control information to other nodes */
        end if
    end for
end
```

The procedure used:
BROADCAST_PACKET ( ) – The node with the highest weight broadcasts a packet declaring itself as the leader and passing its control information to other nodes.

CALL SCHEDULING

Once the clusters are formed and maintained, these can be used to handle incoming call requests. When a mobile node sends a call request, the call is forwarded to the corresponding Base Station which is then forwarded to the cluster-head of the corresponding cluster. This cluster-head then forwards the call to the cluster-head of the cluster to which the call is to be forwarded. The cluster-head then forwards the call to the base station of the call receiving node and subsequently forwarded to the call receiving node.

So, the path of the call request can be represented as follows:

$C_S \rightarrow BS_S \rightarrow Leader_S \rightarrow Leader_R \rightarrow BS_R \rightarrow C_R$

Where, C_S, BS_S and $Leader_S$ denote the sender side and C_R, BS_R and $Leader_R$ denote the receiver side.

CALL ROUTING BETWEEN $LEADER_S$ AND $LEADER_R$

Once the $Leader_S$ and $Leader_R$ cells have been predetermined, the next step is to find the shortest path between $Leader_S$ and $Leader_R$. This is done by using the Kruskal's algorithm. This algorithm finds a minimum spanning tree for a connected weighted graph. So, to map our problem to the Kruskal's algorithm, a connected weighted graph is constructed where the cluster-heads represent the vertices and the edges are represented by weights that denote the Euclidean distances between the nodes. For sake of completeness the Kruskal's algorithm is presented next.

Algorithm:

```
Kruskal(G = <N, A>: graph; length: $A \rightarrow R^+$): set of edges
    Define an elementary cluster $C(v) \leftarrow \{v\}$.
    Initialize a priority queue Q to contain all edges in G, using the weights as keys.
    Define a forest $T \leftarrow \emptyset$    //T will ultimately contain the edges of the MST
    // n is total number of vertices
    while T has fewer than n-1 edges do
        // edge u,v is the minimum weighted route from u to v
        $(u,v) \leftarrow$ Q.removeMin()
        // prevent cycles in T. add u,v only if T does not already contain a path between u and v.
        // the vertices has been added to the tree.
        Let C(v) be the cluster containing v, and let C(u) be the cluster containing u.
        if $C(v) \neq C(u)$ then
            Add edge (v,u) to T.
            Merge C(v) and C(u) into one cluster, that is, union C(v) and C(u).
    return tree T
```

The above algorithm can be shown to run in $O(E \log E)$ time, or equivalently, $O(E \log V)$ time, all with simple data structures. These running times are equivalent because:

- E is at most V^2 and $\log V^2 = 2\log V$ is $O(\log V)$.
- If we ignore isolated vertices, which will each be their own component of the minimum spanning forest, $V \leq E+1$, so $\log V$ is $O(\log E)$.

Next the call scheduling algorithm is presented.

Algorithm:

```
Call_Schedule (Call $t_i$)
begin:
forward call from calling node $C_{Si}$ to base station $BS_{Si}$
forward call from base station $BS_{Si}$ to cluster-head $H_{Si}$
G $\leftarrow$ FORM_GRAPH(network N)
min_span_tree T $\leftarrow$ Kruskal(G)
Edge set {E} $\leftarrow$ SELECT_EDGE_SET($H_{Si}$, $H_{Ri}$)
forward call from cluster-head $H_{Si}$ to cluster-head $H_{Ri}$
forward call from cluster-head $H_{Ri}$ to base station $BS_{Ri}$
forward call from base station $BS_{Ri}$ to receiver node $C_{Ri}$
end
```

The functions used are presented below:

1) FORM_GRAPH(network N) – Forms a connected weighted graph from network N with the mobile nodes as vertices and connecting links as edges having weights equal to the Euclidean distances between the nodes.
2) SELECT_EDGE_SET(H_I, H_J) – Selects the edge set joining the cells H_I and H_J in the minimum spanning tree T.

The time required for cluster formation is already evaluated in [13, 14]. So, the methodology proposed here reduces the computation time to a large extent with respect to previous approaches.

IV. EXPERIMENTAL RESULTS

The proposed model is simulated with Matlab 7.5.0. Here, the result shows the effect of clustering on call scheduling. The Fig. 2 shows the congestion in the network with respect to various cells with and without

clustering. The weight metric used is the counter values that were introduced in [6] and discussed in Section III.D. The counter values are an effective measure for the network congestion with respect to the traffic requests being handled by a given Base Station(cell) at a given point of time. The x-axis gives the id's of the various cells in the network. Thus, e.g., 250 here refers to cell C_{250}. Random calls were initiated with a pseudo-random number generator. The algorithm was executed for 100 iterations and the average
values of the Weight metric(C) were plotted as the graph shown in Fig.2.

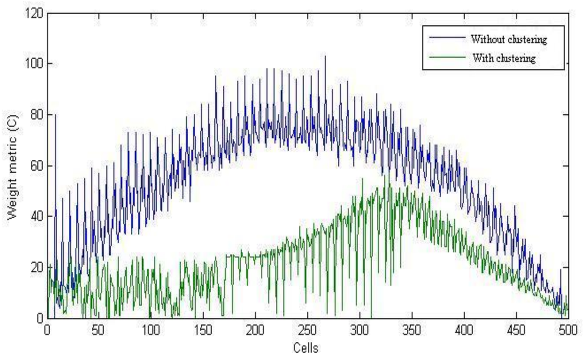

Fig. 2. Congestion in the network for various cells

Quality of Service(QoS)

The Quality of Service(QoS) is inversely proportional to the congestion in the network. So, the QoS metric(Q) increases when the congestion metric(C) decreases and vice versa.
This can be mathematically represented as :

$$\mathrm{QoS(Q)}\ \alpha\ \frac{1}{Congestion\ metric(C)}$$

The Quality of Service metric Q can be parametrically represented as the following mathematical formulation:

$$QoS(Q) = \phi + \log\left(\frac{1}{Congestion\ metric\ (C)}\right)$$

Where, $\phi = -\mu + \text{minimum value of} \log\left(\frac{1}{C}\right)$

μ is a predefined constant having value 0.5. The value of μ is chosen in such a way in order to make the parameter Q to assume positive values only.
Hence we get the following graph for QoS versus the cells.

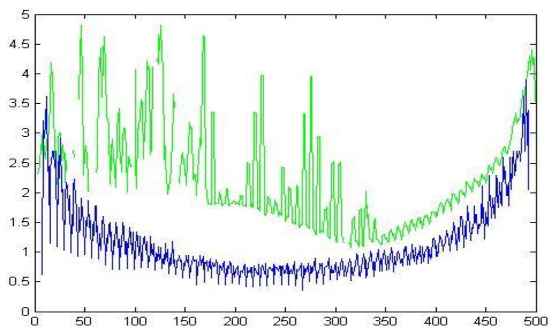

Fig 3. Quality of Service(QoS) for various cells

V. CONCLUSION

The procedure for cluster based call scheduling methodology has been described in this work. The searching cost is reduced for using the cluster head in routing. At the same time, the leader (cluster head) election algorithm has been enhanced with the inclusion of *link_breakage* and *link_emergence* procedures. So, it increases the flexibility of dynamic call scheduling. The use of Kruskal's algorithm for finding the shortest path and the subsequent call scheduling algorithm proposed in this paper are reactive call scheduling strategies that, though efficient in terms of storage costs require higher computational resources because the Shortest paths have to found out each time a call is initiated. This problem can be solved by maintaining routing tables. Further study on extending this model for construction of routing tables is in progress.

REFERENCES

[1] P.K.Guha Thakurta and Subhansu Bandyopadhyay, "A New Dynamic Pricing Scheme with Priority based Tree Generation and Scheduling for Mobile Networks", IEEE Advanced Computing Conference, March 2009.

[2] P.K.Guha Thakurta, Rajarshi Poddar and Subhansu Bandyopadhyay, "A New Approach on Co-ordinate based Routing Protocol for Mobile Networks", IEEE Advanced Computing Conference, February 2010 .

[3] Wen-Hwa Liao, Jang-Ping Sheu and Yu-Chee Tseng, "GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks", Journal on Telecommunication Systems, Springer Netherlands, Vol 18, No. 1-3, September 2001.

[4] Sudarshan Vasudevan, Jim Kurose and Don Towsley, "Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks", IEEE ICNP, 2004, page(s): 350-360.

[5] C. Bohm and F. Krebs, "The k-nearest neighbour join: Turbo charging the kdd process", Journal on Knowledge and Information Systems, Vol. 6, No. 6, 2004.

[6] P.K.Guha Thakurta, Subhansu Bandyopadhyay, S. Basu and S. Goswami, "A new approach on Congestion Control with Delay Reduction in Mobile Networks", Second International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom), October, 2010.

[7] E. J. Dijkstra and C.S. Scholten, "Termination detection for diffusing computations", In Information Processing Letters, Vol. 11, No. 1, pp. 1-4, August 1980.

[8] Joseph. B. Kruskal, “On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem”, In proceedings of the American Mathematical Society, Vol 7, No. 1 (Feb, 1956), pp. 48–50.

[9] Liliana M. Arboleda C. and Nidal Nasser, “Cluster-based routing protocol for mobile sensor networks”. In Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks (QShine '06). ACM 2006, New York, NY, USA, Article 24.

[10] M. Rezaee and M. Yaghmaee, “Cluster based Routing Protocol for Mobile Ad Hoc Networks”, InfoComp, Vol 8, No 1, March 2009, pp 30-36.

[11] Rana E. Ahmed, “A Fault-Tolerant Routing Protocol for Mobile Ad Hoc Networks”, Journal of Advances in Information Technology, Volume 2, Number 2, May 2011, Page (s): 128 – 132.

[12] Subhash K. Shinde, Uday V. Kulkarni, “Hybrid Personalized Recommender System Using Fast K-medoids Clustering Algorithm”, Journal of Advances in Information Technology, Volume 2, Number 3, August 2011, Page(s): 152 – 158.

[13] J. Usha , Ajay Kumar and A.D. Shaligram, "Clustering Approach for Congestion in Mobile Networks", IJCSNS International Journal of Computer Science and Network Security, Volume 10 Number 2, February 2010, Page(s) 113-118.

[14] Mary Inaba, Naoki Katoh, Hiroshi Imai, "Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering", In proceedings of the 10th annual Symposium on Computational Geometry, New York, USA, 1994, ACM Digital Library, ISBN:0-89791-648-4.

P. K. Guha Thakurta: He passed B.Tech and M.Tech in computer science & engineering from Kalyani University and Calcutta University in 2002, 2004 respectively. He is currently working as an Assistant Professor of CSE dept. in National Institute of Technology, Durgapur, India. His research area is Mobile Computing.

Saikat Basu is a PhD student at the Computer Science Department at Louisiana State University, USA. He received his Btech degree in Computer Science from National Institute of Technology Durgapur, India in 2011. His research interests lie in the field of Mobile Computing, Security and Video Surveillence.

Sayan Goswami is a junior associate at Sapient Global Markets. He received his Btech degree in Computer Science from National Institute of Technology Durgapur, India in 2011. His research interests lie in the field of Mobile Computing and Embedded Systems.

Subhansu Bandyopadhyay is an eminent professor in the dept. of Computer Science & Engineering, University of Calcutta.