# Neural Networks Based Efficient Multiple Multicast Routing for Mobile Networks

Vijaya Kumar B. P., *Member, IACSIT* and Dilip Kumar S. M., *Member, IACSIT*

***Abstract*—Mobile multimedia networks that handle multicast communication services require a reliable and efficient point-to-multipoint specific group communications. In mobile networks, this leads to many challenges to provide an efficient multicast routing. In this paper, an extension to our previous work is proposed. In the previous work, the construction of multicast distribution tree (MDT) for multicast routing in mobile networks was proposed. However, in this work, the technique is revised and scalability is adapted to construct an efficient *multiple* multicast routing in mobile networks. The technique involves the construction of MDTs for multiple multicast groups (MCG), considering the network traffic and available resources. The technique uses Hopfield Neural Network (HNN) and Kohonens Neural Network (KNN) for the construction of efficient multiple MDTs. The computational power of the proposed technique is demonstrated through simulation. The technique is tested for different group size and network topologies along with host mobility. The proposed work facilitates a possible multicast routing technique for future high speed mobile networks and on-demand multicast applications.**

***Index Terms*—Efficient routing, mobile networks, multiple multicast routing, neural networks.**

## I. Introduction

Multicast is a mechanism for *one-to-many* communication in which the sender transmits a data packet, and the network performs the task of delivering the data packets to multiple destinations. By eliminating multiple transmissions, multicast results in significant savings in host processing and network bandwidth [1], [2]. In $1 \times N$ multicasts, i.e., a single sender to $N$ destinations will probably be the most commonly used form of multicasting as it is used by a variety of applications, such as transmission of news program, distant learning, weather reports, on-demand applications, etc. Some of the features that such a mechanism in mobile networks must have are:

1) To construct a $1 \times N$ MDT from source to $N$ destinations.
2) Establishing the MDT as quickly as possible.
3) Computing the MDT as and when required to hide the mobility of host from applications.
4) An adaptation scheme for addition and deletion of participating hosts in a multicast group (MCG).
5) Ensure end-to-end reliable multicast message delivery.

In the previous work [3], a multicast routing algorithm for constructing a reliable multicast tree that connects the participants of a single MCG by considering reliable nodes in a mobile network was proposed. In this work, the technique is modified and scalability is adapted to construct an efficient *multiple* multicast routing in mobile networks. The technique is aimed at constructing low cost MDTs for the MCGs admitted based on the traffic load on the network and the type of application run by each group. Further, the technique establishes MDTs when there is a change in location of the group members due to mobility. The technique employs two different types of neural networks (KNN and HNN) and their computational power is used as a heuristic approach to tackle the issues in multiple multicast routing. By the massive parallel computation and learning capability of neural networks, we can find a near-optimal multicast route very fast when implemented in hardware [4]. The rest of the paper is organized as follows. Some of the related works are highlighted in Section II. The mobile network model is presented in Section III. The proposed multicast routing technique along with algorithms is described in Section IV. The principle of clustering and construction of MDTs by using KNN and HNN are discussed in Section V. Simulation results of the proposed technique and concluding remarks of this work are presented in Sections VI and VII respectively.

Manuscript received October 20, 2013; revised December 20, 2013.

B. P. V. Kumar is with the Dept. of Information Science and Engineering, M. S. Ramaiah Institue of Technology, Bangalore, India (e-mail: vijaykbp@yahoo.co.in).

S. M. D. Kumar is with the Dept. of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore, India (e-mail: dilipkumarsm@gmail.com).

## II. Related Works

There has been several research works in the area of multicast routing and neural network based routing in mobile networks. Some of them are highlighted in this section. In [5], a method called bi-directional tunnel and rebuilding the tree is proposed, where a mobile sender post the MCG membership information to either its home agent on the home link or to its local gateway router on the foreign link. The above technique may lead to overload of home agents, non optimal routing and long delay times for rerouting. A low-cost multicast routing algorithm in mobile networks is suggested to reduce the overall cost of the multicast tree and the latency for join and leave operations when mobile hosts move from one cell to another [6]. A minimum delay routing algorithm is proposed [7] using neural network to find a set of alternative routes for each commodity to distribute optimally for reducing the time delay a packet encounters. An improved neural heuristics for multicast routing is proposed and the routing is achieved by constructing a minimum cost tree that spans the source and destinations for multi-point communication [8]. In [9] and [10], the authors developed a NN-based shortest-path algorithm by means of which the implementation of the optimal routing problem was discussed. A unified approach for the multiple multicast tree construction and rate allocation problem is addressed in [11].

The objective of this problem is to find a max-min fair rate allocation among the multiple MCGs that co-exist in the network subject to the link-capacity constraints. Bandwidth-aware multiple multicast tree formation and link scheduling for multiple multicast sessions for wireless networks are presented in [12] and [13] respectively.

## III. Mobile Network Model

A typical mobile network consists of cells of different size and shape controlled by Base Stations (BS) or Mobile Switching Stations (MSS). The MSS provides an end-access connection for roaming mobile hosts and is interconnected by either wired or wireless media. Assuming that all MSSs acts as multicast supporting routers and are capable of subscribing to MCGs, a set of reliable BSs are considered to select a path for constructing a MDT. The topology of a mobile network may be treated as an undirected graph $\psi=(V,E)$, where $V$ is the set of nodes, representing the MSSs, $E$ is the set of edges representing the wired/wireless links connecting the MSSs, and a set $\Lambda_g \subseteq V$ representing the MSSs to which participating mobile hosts of the MCG $g$=1,2,…,$G$ are connected, where $G$ = no. of MCGs. Now the problem is to find a MDT that connects the source node and all other MSSs that belongs to set $\Lambda_g, \forall \in [1,G]$, with minimum number of links, reliable MSS and cost effective distribution paths based on the traffic load.

## IV. Proposed Multicast Routing

The proposed technique is to construct an efficient MDT for a given set of MCGs in mobile networks. Following objectives are considered to construct an efficient MDT for a given set of MCGs in mobile networks.

- To minimize number of links connecting every node belonging to a MCG.
- To maintain minimum number of hops between each node to every other nodes belonging to a MCG.
- To establish the route tree quickly as and when the mobile hosts change their location.
- To construct MDTs for the admissible MCGs based on the existing load on the network and the type of application to be run by individual MCGs.

The description of the technique involves two phases and is explained with algorithms in the following subsections.

### A. Phase I: Construction of Multicast Distribution Tree

This phase involves three steps. The description of each step and the solution along with the algorithms are given below:

- Step I: Clustering and Center Cluster (CC) for each MCG

This step involves the clustering of nodes and the computation of CC for each MCG whose nodes are almost equidistant from the group members. Clustering is performed by partitioning the mobile network based on the adjacency relation between the nodes with suitable neighborhood distance. The distance is fixed based on the size of the network, its topology and the number of clusters to be formed. The CC for each MCG is computed by considering the shortest distance between the group members. This CC concept will minimize the path length and number of links used for constructing a MDT. The KNN model is used for clustering (described in Section V.A), which has the capability to group similar featured input patterns into clusters. Algorithm 1 describes the computation of the CC for MCG members.

1) Divide the mobile network into clusters of nodes based on the adjacency relation between the nodes;
2) The CC among the set of clusters to which the particular group member belongs is identified for each MCG;
3) Output the nodes that belong to the CC of corresponding MCG;

**Algorithm 1**: Finding the CC for each MCG.

**Nomenclature:**
*G = no. of MCGs, g = 1,2,…,G (MCG members), H = #hops for MSSs*

$v_{ig}$ , $v_{dg}$ = *source and destination node of group g*

$Path_{ig}$ = *output path-set for node-pair* $i$ . *Initially set to* $\phi$

***For each*** *MCG* $g = 1, 2, ..., G$ ***do***

  ***For each*** *node-pair* $(v_{1g}, v_{dg})_i$ ***do***

    $V_{Ig}=V_{1g}$ ; *L=0;* $Path_{ig}=\phi$ ;

    **CALL** Procedure **PATH** ($V_{Ig}=V_{1g}$ , $Path_{ig}$ );

  ***End for***

***End for***

*Procedure* ***PATH*** ($V_{Ig}=V_{1g}$ , $Path_{ig}$ )

***If*** $v_I == v_d$ ***then*** *Output* $Path_{ig}$

  ***If*** $Path_{ig}$ *contains node(s) of Center Cluster* ***then return****;*

  ***Else If*** ($v_I$ *is not in* $Path_i$ ) *and (L < H) and (*$v_I$ *is reliable) then*

    *L++; add* $v_I$ *to path;*

  ***End if***

  ***For*** *each neighboring node* $v_z$ *of node* $v_I$ ***do***

    ***PATH*** ($v_z$ , $v_d$ , *s* $Path_i$ );

  ***End for***

***End if***

*L--;*

***End if***

**Algorithm 2:** To compute the set of node-pairs.

- Step II: Computation of a set of node-pair paths

This step involves the computation of a set of paths from the source node to each destination group member by considering the reliable nodes that passes through the CC of corresponding MCG.

*Notations and Problem Formulation:* Let $\Lambda_g \subseteq V$ be a set containing the nodes that belongs to the MCG $g$=1, 2,…,$G$ where $\Lambda_g=\{v_{1g}, v_{2g}, …, v_{Ng}\}$, $v_{ig}=i^{th}$ node that belongs to the MCG $g$, $N_g$ = no. of participating nodes in MCG $g$ and $G$ = no. of MCGs under the assumption that $G$, $N_g$'s and $\Lambda_g$'s are given are given at the instant when the routing algorithm is initiated. Before considering the actual algorithm, a node-pair set is defined

as $PR_g=\{(v_{1g},v_{2g})_1,(v_{1g},v_{2g})_2,\ldots,(v_{1g},v_{Ng})_{N_g}\}$, for the $N_g$ nodes that belongs to the MCG $g$ with $v_{1g}$ as the source node. The elements of the set $PR_g$ are the node-pairs and the number of elements in the set is $N_g-1$, under the assumption that one or more paths exist between each node-pair and the link between nodes is bidirectional. Similarly the node-pair sets are defined for all the MCGs. The reliable nodes are considered during path selection.

*Finding the set of paths:* Algorithm 2 finds all possible paths between the node-pair that cross no more than H hops (or MSSs) by avoiding unreliable nodes. The paths are selected to pass through the CC of corresponding MCG. This algorithm performs a tree search technique until destination is reached and is performed for all the MCGs. The path set is found for each node-pair of all the MCG passing through the nodes in the CC of corresponding group when the algorithm terminates.

- Step III: Construction of multicast distribution trees

The task involves selection of a single path from the set of paths found in the previous step between each node-pair of all the MCG g to construct MDTs. Here, the construction task is formulated into an optimization problem. Before formulating the problem, some of the notations defined are as follows.

- $(v_{ag},v_{bg})_i$ = $i^{th}$ node-pair for MCG $g$, where $a\neq b$, $v_{ag}$ is the source node, $v_{bg}$ is the destination node and $a,b\in[1,N_g]$.
- $NP_i$ = no. of paths between $i^{th}$ node-pair $(v_{ag},v_{bg})_i$ of MCG $g$, which can be fixed based on the requirement.
- $R_{ij}$ = $j^{th}$ path between $i^{th}$ node-pair $(v_{ag},v_{bg})_i$ of $g$.
- $X_{ij}^{g}=1$ if path $R_{ij}$ is chosen; else 0.

From the set of paths connecting each node-pair, $(v_{ag},v_{bg})_i$ where $i$=1, 2, …, $Ng$—1 select a single path between each node-pair, so that the number of links connecting all group members, and the number of MSSs between the source and every group members is minimized by considering the existing load on the network. The formulation of the objective functions for this optimization problem is as follows.

1) To minimize the no. of links connecting every node belonging to a MCG $g$=1, 2, …, $G$
Minimize

$$O_1=\frac{1}{2}\sum_{g=1}^{G}\sum_{i=1}^{N_g-1}\sum_{k=1,k\neq i}^{N_g-1}\sum_{j=1}^{NP_g(i)}\sum_{l=1}^{NP_g(k)} f(R_{ij}^{g},R_{kl}^{g})X_{ij}^{g},X_{kl}^{g} \quad (1)$$

2) To minimize the no. of hops between the node-pair by considering the existing link load on the network:
Minimize

$$O_2=\frac{1}{2}\sum_{g=1}^{G}\sum_{i=1}^{N_g-1}\sum_{j=1}^{NP_g(i)}\sum_{l=1,l\neq j}^{NP_g(i)} q(R_{ij}^{g})X_{ij}^{g},X_{il}^{g} \quad (2)$$

3) To minimize the no. of common links between MDTs that belongs to different groups (for load balancing).
Minimize

$$O_3=\frac{1}{2}\sum_{g=1}^{G}\sum_{g'=1}^{G}\sum_{i=1}^{N_g-1}\sum_{k=1,k=i}^{N_{g'}-1}\sum_{j=1}^{NP_g(i)}\sum_{i=1}^{NP_{g'}(k)} h(R_{ij}^{g},R_{kl}^{g'})X_{ij}^{g},X_{kl}^{g'} \quad (3)$$

where $f(R_{ij}^{g},R_{kl}^{g})=|R_{ij}^{g}|+|R_{kl}^{g}|-|R_{ij}^{g}\cap R_{ij}^{g}|$ for $g$, $q(R_{ij}^{g})=|R_{ij}^{g}|*L_{ij}^{g}$, $|R_{ij}^{g}|$ the no. of links in $R_{ij}^{g}$ path for the MCG g, $|R_{ij}^{g}\cap R_{kl}^{g'}|$ = the no. of common links shared by paths $R_{ij}^{g}$ and $R_{ij}^{g'}$ for groups $g$ and $g'$ respectively, $L_{ij}^{g}$ = existing load on the path $R_{ij}^{g}$ (value is fixed based on the maximum link load among the links over a path $R_{ij}^{g}$, (normalized w.r.t. its capacity)). Now the objective is to minimize the functions in Eqns. 1-3, subject to the constraints that a single path is chosen for each node-pair (i.e. $X_{ij}^{g}=1$ for exactly one value of $j$ for each value of $i$ belonging to group $g$) and totally $N_g-1$ paths are selected. The objective function corresponds to the no. of links of all selected path from the set of paths to each node-pair and the no. of hops between the node-pair of the selected paths by considering the existing link load for each MCG. Similarly, load balancing is maintained by minimizing the common links between the MDTs that belongs to different groups. This optimization problem is solved by using the HNN model (described in Section V.B). Algorithm 3 summarizes the construction of a MDT for a given set of paths to each node-pair of different group members.

1) Select a single path for each node-pair such that the objective function specified in Eqns. 1-3 is minimized;
2) MDTs are constructed from the set of selected paths from source to destination group members for each group;

**Algorithm 3**: Construction of a MDT.

### *B. Phase II: Admission of MCG*

This phase (Algorithm 4) involves the computation of the admissible MCGs based on the type of application run by each group and existing traffic over the links in the constructed MDTs for the given MCGs.

1) Assign priorities to the application, i.e., high priority to real-time application and low priority to non-real-time applications;
2) For all constructed MDT links, compute the total link load by considering existing load and added load due to multicast applications;
3) if link load exceeds its capacity then reject the low priority application, in turn the MCG assigned with particular application end
4) Output the admitted MCG and corresponding MDTs;

**Algorithm 4**: Admission of MCG.

*C. Example*

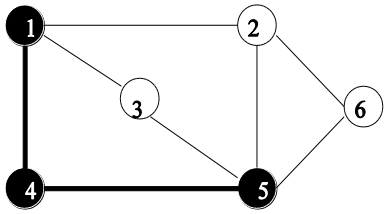


Fig. 1. Example network with 3 MCG nodes (1, 4, 5).

The example given in this section illustrates the construction of a MDT. Consider a 6-node mobile network as shown in Fig. 1 with one MCG i.e. $G$=1 for simplicity, with node-set $\Lambda_g$ ={1, 4, 5} and $N_g$ =3 . The corresponding node-pair set $PR_g$ ={$(1,4)_1$,$(1,5)_2$ $(4,5)_3$} . We consider the hop count $H$=3 , then there exists 3 paths between the node-pairs (1, 4), (4, 5) and 4 paths between node-pair (1, 5) i.e., $NP_g(1) = 3$ , $NP_g(2) = 4$ and $NP_g(3) = 3$. The paths connecting these node-pairs are as follows.

- Path set for the node-pair $(1, 4)_1$ is: $R_{11}$ = (1, 4), $R_{12}$ = (1, 3, 5, 4), $R_{13}$ = (1, 2, 5, 4),
- Path set for the node-pair $(1, 5)_2$ is: $R_{21}$ = (1, 3, 5), $R_{22}$ = (1, 4, 5), $R_{23}$ = (1, 2, 5), $R_{24}$ = (1, 2, 6, 5),
- Path set for the node-pair $(4, 5)_3$ is: $R_{31}$ = (4, 5), $R_{32}$ = (4, 1, 3, 5), $R_{33}$ = (4, 1, 2, 5).

## V. Neural Network Model for Clustering and Multicast Tree Construction

In this section, the principle of constructing MDTs using KNN and HNN models are described.

*A. Kohonen Neural Network*

KNN is a competitive, feed-forward type and unsupervised training neural network [14]. In the process of learning, the network is able to discover statistical regularities in its input space and automatically develops different groups or clusters to represent different classes of inputs.

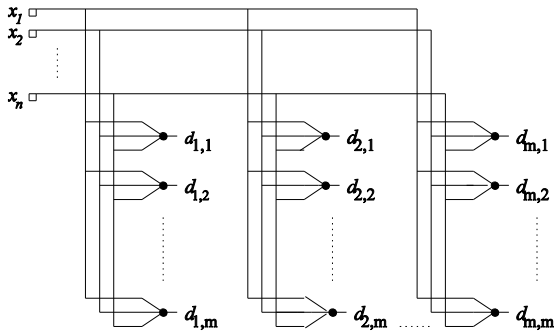


Fig. 2. A typical Kohonen neural network model with n input and m*m output neurons.

*Selection of Neurons and KNN architecture:* Kohonen neural networks have two layers, an input layer and an output layer known as competitive layer. Each input neuron is connected to every neuron on the output layer which are organized as a two dimensional grid. Fig. 2 shows a structure of a typical KNN. It has n input nodes and m × m output nodes. The input layer in the KNN has the same number of neurons as the size of each input pattern. In the present case, the input patterns are the rows of the adjacency matrix of the network topology. For example, if the size of the adjacency matrix for the given network topology is $|V| \times |V|$, where $V$ is a set of vertexes, hence the number of input neuron is $|V|$. However, the neurons on the output layer find the organization of relationships among input patterns which are classified by the competitive neurons and can organize a topological map from a random starting point and the resulting map shows the natural relationships among the patterns that are given to them. The number of output layer neurons (competitive layer neurons) is selected based on the number of distinct patterns (classes) present in the input patterns. 50 output neurons for a 25 node network topology with suitable neighborhood value are chosen in the present case.

*Learning algorithm:* The network is presented with a set of training input patterns I, i.e., rows of the adjacency matrix of the network topology. Let the input pattern be: $I=(I_1,I_2,\ldots,I_{|V|})$ where there are $|V|$ input neurons in the input layer. Now suppose there are $Q$ neurons in the output (Competitive) layer let $o_j$ be the $j^{th}$ output neuron. So the whole of the output layer will be: $o=(o_1,o_2,\ldots,o_Q)$ . For each $o_j$ there are $|V|$ incoming connections from $|V|$ input neurons. Each connection has a weight value W. Therefore, for any $o_j$ on the output layer the set of incoming connection weights are: $W_j=(W_{j1},W_{j2},\ldots,W_{j|V|})$ . The Euclidean distance value $D_j$ of neuron $o_j$ in the output layer, whenever $I$ is presented at the input layer is:

$$D_j=\sqrt{\sum_{i=1}^{|V|}(I_i-W_{ji})^2} .$$

The competitive output neuron with the lowest Euclidean distance at this stage is the closest to the current input pattern, called as winning neuron, $o_c$ . Once $o_c$ is identified, the next step is to identify the neighborhood around it (set of output neurons). The overall weights update equation is: $W_j^{new}=W_j^{old}+\alpha(I-W_j^{old})$ if $o_j$ is in the neighborhood of winning unit $o_c$ . Here, $\alpha$ is the learning rate parameter, typical choices are in the range $[0.2\ldots0.5]$ . The learning algorithm for clustering is summarized and is presented below.

1) Assign random values to connection weights W in range $[0,1]$;
2) Select an input pattern from the adjacency matrix of the network topology (row of an adjacency matrix);
3) Determine the winner output neuron $o_c$ ;
4) Perform a learning step affecting all neurons in the neighborhood of $o_c$ by updating the connection weights;
5) Update $h_t$ and continue with step 2 until no noticeable changes in the connection weights are observed for all the input patterns'

**Algorithm 5**: Learning algorithm for clustering.

*B. Mapping to HNN Model*

In this subsection, the HNN model for the construction of MDT for the MCGs is presented.

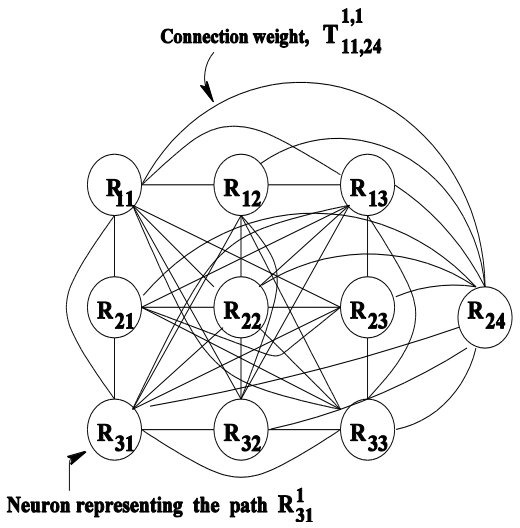


Fig. 3. Hopfield neural network model representing the neurons for each path of the example network shown in Fig. 1.

*Selection of neurons and HNN architecture:* In the proposed HNN model, we define one neuron for each path between a node-pair. Hence, the total number of neurons $\eta$, for $N_g$ number of nodes in each MCG, $g=1,2,\ldots,G$ is $\eta=\sum_{g=1}^{n}\sum_{i=1}^{N_g-1} NP_g-1(i)$. In Fig. 3, there are 10 neurons, one neuron for each path of every node-pair, and are connected by connection weight matrix *T*, where $T^{g}_{ij}, T^{g'}_{ij}$ is a connection weight between the neurons representing the path $R^{g}_{ij}$ and $R^{g'}_{ij}$ for MCGs $g$ and $g'=(1,2,\ldots,G)$ respectively. The connection weight matrix T is $\eta\times\eta$ and symmetric. The elements of T are connection weights $T^{g}_{ij}, T^{g}_{kl}=T^{g}_{kl}, T^{g}_{il}$ between the neurons of same MCG $g$ and $g'$, and $T^{g}_{ij}, T^{g}_{ij}=0$ for MCG $g=1, 2,\ldots, G$. The weight values of these connections are chosen to encourage:

- The sharing of links by many paths of different node-pairs of group $g$,
- The selection of paths to minimize the number of links and with links having less traffic load,
- The correct number of path activations, (i.e., exactly one neuron is turned ON per node-pair of group $g$) and
- To minimize the number of common links between the MDT that belongs to different groups to have load balancing on the network.

Here each neuron is represented with triple indices (i.e., neuron $R^{g}_{ij}$ represents the $j^{th}$ path between the node pair $i$ of group $g$).

*Dynamics of HNN model:* The dynamics of HNN is described by a general equation of motion and may be written in terms of connection weights, bias input and neuron output relating to the present problem [11] as follows.

$$E_{total}=-\frac{1}{2}\sum_{g'=1}^{G}\sum_{g=1}^{G}\sum_{i=1}^{N_g-1}\sum_{k=1}^{N_{g'}-1}\sum_{j=1}^{NP_g(i)}\sum_{l=1}^{NP_{g'}(i)} T^{g,g'}_{ij,kl} X^{g}_{ij} X^{g'}_{kl} - \sum_{g=1}^{G}\sum_{i=1}^{N_g-1}\sum_{j=1}^{NP_g(i)} X^{g}_{ij} I^{g}_{ij} \text{ for } (i,j)\neq(k,l) \text{ if } g=g' \quad (4)$$

where $T^{g}_{ij}, T^{g'}_{kl}$ = strength of the connection weight between neurons representing the path $R^{g}_{ij}$ and $R^{g'}_{ij}$, for MCGs $g$ and $g'=(1,2,\ldots,G)$; $NP_g(i)$ = no. of paths between node-pair $i^{th}$ of MCG $g$; $I^{g}_{ij}$ = bias value applied to neuron $ij$ of MCG $g$ (bias values are set to zero in the present case) and $X^{g}_{ij}=ij^{th}$ neuron output of MCG $g$. The neuron input-output relation is a sigmoid activation function: $X_{ij}=\phi(u_{ij})=(1+e^{-u_{ij}})^{-1}$ where $u_{ij}$ is the input value to neuron $ij$ from all the other neuron's output.

*Problem constraints and energy function:* Here the goal is to minimize the total energy function value involving objective function in Eqns. 1-3 and constraints $C_1-C_3$ given below which equates to zero when the constraints are satisfied.

To activate no more than one path per node-pair of each MCG:

$$C_1=\frac{1}{2}\sum_{g=1}^{G}\sum_{i-1}^{N_g-1}\sum_{j=1}^{NP_g(i)}\sum_{l=1,l\neq j}^{NP_g(i)} X^{g}_{ij} X^{g}_{il}=0 \quad (5)$$

1) To activate no more than one path per node-pair of each MCG:

$$C_2=\frac{1}{2}\sum_{i=1}^{G}\sum_{i=1}^{N_g-1}\left(\sum_{j=1}^{NP_g(i)} X^{g}_{ij}-1\right)^2=0 \quad (6)$$

2) To activate a total of exactly $\sum_{i=1}^{G} N_g-1$ paths in the network for G MCGs:

$$C_3=\frac{1}{2}\left(\sum_{g=1}^{G}\sum_{i=1}^{N_g-1}\sum_{j=1}^{NP_g(i)} X^{g}_{ij}-\sum_{g=1}^{G} N_g-1\right)^2=0 \quad (7)$$

The total energy function $E_{\text{total}}$ that satisfies the objective function 1-3 and constraints 5-7 for the problem is as follows: Let

$$M_g=N_g-1 .\ E_{total}=O_1+O_2+O_3+C_1+C_2+C_3$$

$$E_{total}=\frac{A}{2}\sum_{g=1}^{M_G}\sum_{i=1}^{M_g}\sum_{k=1,k\neq i}^{M_g}\sum_{j=1}^{NP_g(i)}\sum_{l=1}^{NP_g(k)} f(R^{g}_{ij},R^{g}_{kl})X^{g}_{ij},X^{g}_{kl} + \frac{B}{2}\sum_{g=1}^{G}\sum_{i=1}^{M_g}\sum_{j=1}^{NP_g(i)}\sum_{l=1,l\neq j}^{NP_g(i)} q(R^{g}_{ij})X^{g}_{ij},X^{g}_{il} + \frac{C}{2}\sum_{g=1}^{G}\sum_{g'=1}^{G}\sum_{i=1}^{M_g}\sum_{k=1,k\neq i}^{M_g}\sum_{j=1}^{NP_g(i)}\sum_{i=1}^{NP_{g'}(k)} h(R^{g}_{ij},R^{g'}_{kl})X^{g}_{ij},X^{g'}_{kl}$$

$$+\frac{\gamma_1}{2}\sum_{g=1}^{G}\sum_{i-1}^{M_g}\sum_{j=1}^{NP_g(i)}\sum_{l=1,l\neq j}^{NP_g(i)}X_{ij}^{g}X_{il}^{g}+\frac{\gamma_2}{2}\sum_{g=1}^{G}\sum_{i-1}^{M_g}(\sum_{j=1}^{NP_g(i)}X_{ij}^{g}-1)^2$$

$$+\frac{\gamma_3}{2}(\sum_{g=1}^{G}\sum_{i=1}^{M_g}\sum_{j=1}^{NP_g(i)}X_{ij}^{g}-\sum_{g=1}^{G}N_g-1)^2 \quad (8)$$

In Eqn. 8, term A minimizes the total number of uncommon links of the paths in every node-pair of each MCG. The term B minimizes the number of hops from source node to every other node in the group by considering the link load in the network. The term C minimizes the total number of common links between the MDTs of different MCGs. $\gamma_1$ and $\gamma_2$ terms are zero, if no more than one path is activated between the node-pair and for all the MCGs. The $\gamma_3$ term is zero, if exactly $\sum_{g=1}^{G}(N_g-1)$ paths are activated in the solution. Parameters A, B, C, $\gamma_1$, $\gamma_2$ and $\gamma_3$ are to be chosen to best fit the problem during simulation. Values chosen for these parameters are assumed to be positive constant in the present case. Looking at the problem constraints and corresponding terms in the energy equation forced to be equal to nearly zero when constraints are satisfied. This forces the energy function to one of the corner of a hypercube, will have the minimum value and may be taken as an efficient solution for the above problem with proper mapping derived from the neural network output $X_{ij}$.

*Determination of connection weights:* The strength of the connection weights are derived by comparing the objective function and constraints in Eqns. 1-3 and Eqns. 5-7, with general energy function Eqn. 4 for HNN. The resulting connection weight between the neurons $ij$ and $kl$ is:

$$T_{ij,kl}^{g,g'}=-Af(R_{ij}^{g},R_{kl}^{g'})(1-\kappa_{ik})-Bq(R_{ij}^{g})\kappa_{ik}(1-\kappa_{jl})-\gamma_1\kappa_{ik}(1-\kappa_{il})-\gamma_2\kappa_{ik}-\gamma_3$$

where $\kappa_{ab}=1$ if $a=b$ ; 0 otherwise ; Kronecker delta function.

*Equation of motion:* As the HNN evolves from an initial state, it follows a trajectory over which energy function decreases monotonically until equilibrium at a (local) minimum is reached [11]. The evolution of the inputs at each neuron is characterized by the equation: $\frac{du_{ij}}{dt}=-\frac{u_{ij}}{\tau}-\frac{\partial E_{total}}{\partial X_{ij}}$. The equation of motion for an HNN with symmetric connection weights, i.e. $T_{ij,kl}=T_{kl,ij}$ will always guarantees convergence to a stable state [11]. Here the final state depends on the initial state at which system evolution has started, objective function and constraints. The equation of motion can be expressed in iterative form as follows.

$$u_{ij}(t+\Delta t)=\frac{u_{ij}}{\tau}-\Delta t\cdot\sum_{g=1}^{G}(A\sum_{k=1,i\neq k}^{M_g}\sum_{i=1}^{NP_g(k)}f(R_{ij}^{g},R_{kl}^{g})X_{ij}^{g}X_{kl}^{g}$$

$$-B\sum_{i=1,i\neq j}^{NP_g(i)}q(R_{ij}^{g})X_{ij}^{g}-C\sum_{g'=1,g'\neq g}^{G}\sum_{k=1,i=k}^{M_{g'}}\sum_{i=1}^{NP_g(k)}h(R_{ij}^{g},R_{kl}^{g})X_{ij}^{g}X_{kl}^{g}$$

$$-\gamma_1\sum_{i=1,i=j}^{NP_g(i)}X_{ij}^{g}-\gamma_2(\sum_{l=1}^{NP_g(i)}X_{il}^{g}-1)-\gamma_3\sum_{k=1}^{M_g}\sum_{i=1}^{NP_g(k)}X_{il}^{g}-M_g \quad (9)$$

Some of the issues that arise when these equations are simulated in software are, choosing the coefficients A, B, C, $\gamma_1$, $\gamma_2$, $\gamma_3$ and $\Delta t$. Usually these values are chosen to ensure that the energy function in Eqn. 8, which is characterized by the presence of valleys in the energy surface, and has only one point with a graceful descent along the energy surface so that the network can converge to valid solution of high quality [9], [11]. The step size $\Delta t$ is chosen such a way that there should not be any oscillation of the neuron output values (i.e., an increase in energy function $E_{total}$) due to large value for $\Delta t$ or extremely slow convergence due to small value for $\Delta t$. Algorithm 6 describes the steps to find an efficient MDT.

1) Find the center nodes for each of the MCGs using clustering based on adjacency relationship between the nodes using KNN (Algorithm 1)
2) Find for every node-pair a set of paths that cross no more than H hops by avoiding unreliable nodes and passes through the center node(s) of each MCG (Algorithm 2);
3) Design an HNN model for the path set of each MCG;
   - Randomize the initial input to neurons, count=0;
   - Use equation of motion and sigmoid function to update the neuron output;
   - GO TO step 3.b until HNN converges to stable state or count = 5000;
4) Select the admissible multicast group based on the network load and the type of applications (Algorithm 4);
5) Construct MDTs for the admitted MCGs from the selected reliable path for each node-pair of the corresponding group;

**Algorithm 6**: To compute efficient multiple MDT.

## VI. Simulation

In our simulation, the mobile nodes, network environment and parameters are chosen to best fit the simulation platform. The proposed multicast routing algorithm is used for computing the MDT for each MCG for the generated graphs. The experiment is carried out by considering 16, 25, and 30 node networks with change in topology. The time evolution of the state of the HNN is simulated by solving Eqn. 9 using HNN model. When the HNN converges to a stable state, i.e., no more updates of neuron output with further iterations, the output $X_{ij}^{g}(=1)$ MCG $g$ is chosen, similarly for all other node-pairs and MCGs. During simulation, the output of HNN used for constructing a MDT is maintained and used as the initial values for constructing a MDT next time, when the member(s) moves. This method of using initial values for the HNN has taken less number of iterations to converge. Hence, once the tree is constructed at the initial stage, the re-construction of the tree is fast, when certain percentage of participating group member(s) changes their location. The values chosen for the energy function coefficients of HNN model in the simulation: A = 18.5, $\gamma_1$ = 6.5, $\gamma_2$ = 3.5, $\tau$ = 1.0 and $\Delta t$ = 0.005 and for KNN model, the neighborhood size $h_0$ = 6 and learning rate $\alpha$ = 0.3 are chosen. Simulation results to evaluate the performance of the proposed algorithm are shown in Fig. 4-Fig. 7. The average number of iterations required for converging to efficient route with respect to the

number of MCGs is shown in Fig. 4. The results are taken for different number of MCGs. From the graph it is clear that as the number of MCGs increases the iterations required for converging to efficient route increases. Fig. 5 shows the time required for the algorithm to converge to a feasible solution with respect to the number of MCG nodes. The graph shows that as the number of MCG nodes increases the convergence time increases with respect to the network size. Fig. 6 shows the performance evaluation of the algorithm for mobility, where the participating multicast nodes change their position. Result shows that as the percentage of mobility increases the number of iterations required to construct a MDT is high (or more time for convergence). Figure 6 shows the efficiency of the MDT by having common links with respect to the number of multicast groups and their current distribution of nodes (i.e. topology). From the results in the graph, we found that as the density of the nodes between the multicast group nodes increases, the common links in the MDT's increases.

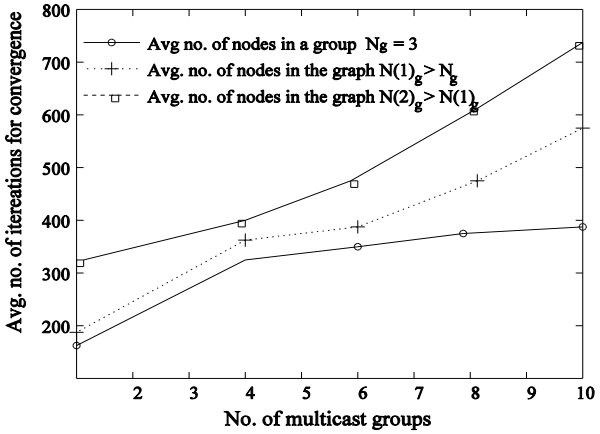


Fig. 4. Avg. iterations for constructing the MDTs w.r.t avg. # nodes in MCG.

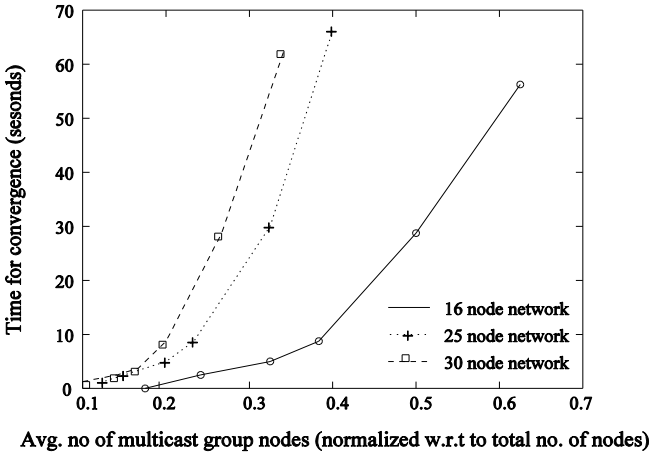


Fig. 5. Time required for the algorithm to converge to feasible solution.

## VII. Conclusion

A multiple multicast routing technique for constructing an efficient MDT for MCGs is proposed. The technique is aimed at constructing MDTs for the groups which are admitted based on the traffic load on the network and the type of application run by each group, that connects the MCG members. This work presents the capability of the neural networks as a computational tool for solving constrained optimization problem. The computational power of the proposed neural network based multicast routing algorithm is demonstrated through simulation. The results verify that the computational time for constructing MDT is low for lesser number of MCGs and group members. The algorithm is also tested for mobility by computing the multicast route for change in location of the participating mobile hosts. The proposed work facilitates a possible multicast routing algorithm for future high speed mobile networks due to the fact that hardware implemented neural network can achieve an extremely high response speed.

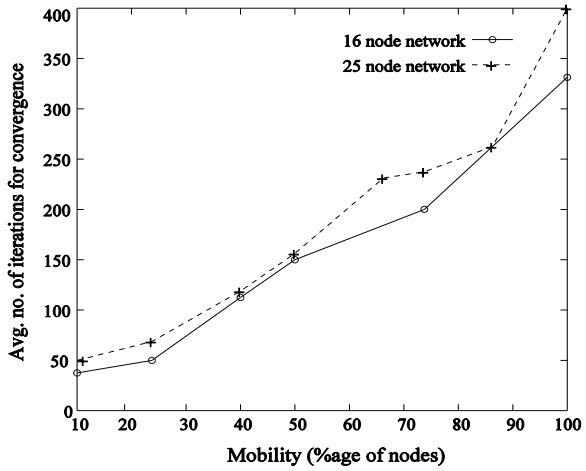


Fig. 6. Percentage mobility w.r.t the iterations for convergence.

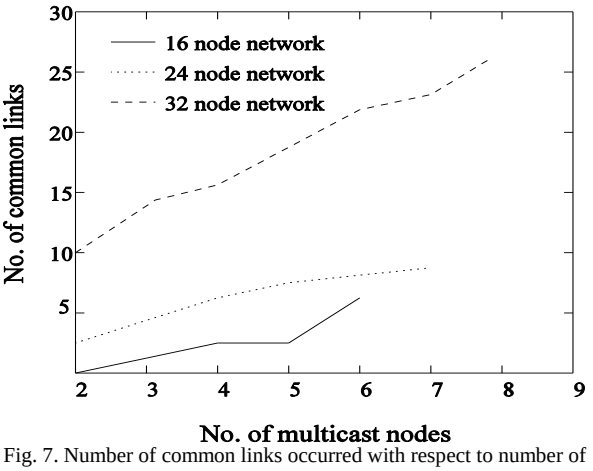


Fig. 7. Number of common links occurred with respect to number of multicast groups for different network sizes.

## References

[1] H. Fang, D. Liang, W. Wang, and Y. Liu, "A time-division multicast transmission scheme using multiple multicast groups," in *Proc. 2nd Intl. Conf. on Computer and Automation Engineering (ICCAE)*, vol. 4, pp. 314-318, Singapore, Feb. 2010.

[2] T.-P Low, Y.-W. P. Hong, and C.-C. J. Kuo, "Opportunistic multicast scheduling with multiple multicast groups," in *Proc. the Global Telecommunications Conf. (GLOBECOM)*, 2011, pp. 1-5.

[3] B. P. V. kumar and P. Venkataram, "A Reliable Multicast Routing in Mobile Networks: A Neural Network Approach," *IEE Communications*, vol. 150, no. 5, pp. 377-384, Oct. 2003.

[4] J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "A neural network approach to routing in multihop radio networks," in *Proc. IEEE INFOCOM'91*, 1991, pp. 1074-1083.

[5] R. S. Chang and Y. S. yen, "A multicast routing protocol with dynamic tree adjustment for mobile IPv6," *Jr. of Information Science and Engineering*, vol. 20, pp. 1109-1124, 2004.

[6] T. Alrabiah and A. Aljadhai, "Low-cost multicast routing in wireless mobile networks," in *Proc. IEEE Wireless Communications and Networking Conf.*, Chicago, USA, Sep. 2000, pp. 1467-1471.

[7] E. Gelenbe, A. Ghanwani and V. Srinivasan, "Improved neural heuristics for multicast routing," *IEEE JSAC*, vol. 15, no. 2, Feb. 1997.

[8] K. Mustafa, M. Ali, and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," *IEEE Tr. on Neural Networks*, vol. 4, no. 6, pp. 941-954, Nov., 1993.
[9] S. Ghosal, P. Venkataram and B. P. Vijay kumar, "Neural network based optimal routing algorithm for communication networks," *Intl. Jl. Of Neural Networks*, vol. 15, pp. 1289-1298, 2002.
[10] S. Haykin, "Neural networks: A comprehensive foundation," presented at Macmillan college publishing company, NY, USA, 1995.
[11] R. Maniyar, P. Ghosh, and A. Sen, "A unified approach for multiple multicast tree construction and max-min fair rate allocation," in *Proc. Intl. Conf. on High Performance Switching and Routing*, 2009, pp. 1-8.
[12] A. Argyriou, "Link scheduling for multiple multicast sessions in distributed wireless networks," *IEEE Wireless Communications Letters*, vol. 2, no. 3, Jun. 2013.
[13] F. Sayit, M. Izmir, E. T. Tunali, A. M. Tekalp, "Bandwidth-aware multiple multicast tree formation for p2p scalable video streaming using hierarchical clusters," in *Proc. 16th IEEE Intl. Conf. on Image Processing (ICIP)*, 2009, pp. 945-048.
[14] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological cybernetics*, vol. 52, pp. 161-152, 1985.

**Vijaya Kumar B. P.** received the Ph.D degree from Electrical Communication Engg. Department, Indian Institute of Science, Bangalore in 2003 and M.Tech degree in Computer Science and Technology from the University of Roorkee with honors in 1992. He is currently a professor in the Information Science & Engineering, M. S Ramaiah Institue of Technology, Bangalore, India where he is involved in research and teaching UG and PG students. He has published more than 60 papers in International Journals and Conferences. His major areas of research are Computational Intelligence applications in Mobile networks.

**Dilip Kumar S. M.** received the B. E degree in 1996 and the M. Tech degree in 2001 from Kuvempu University and Vishweswaraiah Technological University respectively. He obtained the Ph.D degree from Kuvempu University in April 2010. All the three degrees are in Computer Science and Engineering (CS&E) discipline. Currently working as Associate Professor in the Dept. of CS&E, University Visvesvaraya College of Engineering, Bangalore University, Bangalore, India. He is involved in research and teaching and has more than 16 years of teaching experience. He has published 18 papers in International Journals and Conferences. His current research lies in the areas of sensor networks and grid/cloud computing