Reliable multicast routing in mobile networks: a neural-network approach

B.P. Vijay Kumar and P. Venkataram

Abstract: Mobile networks that handle multicast communication services, such as video-on-demand, news-on-demand etc., require a kind of reliable and secure point-to-point, point-to-multipoint specific group communications for sophisticated organisation of multicast communications. A reliable multicast tree is an efficient connectivity between the source node and the multicast group members through the dependable hosts. Because the mobile host changes its access point over time, multicast routes must be updated. This poses several challenges to provide an efficient multicast routing. A neural-network-based multicast routing algorithm is proposed for constructing a reliable multicast tree that connects the participants of a multicast group. The problem is tackled by dividing a mobile network into clusters of nodes based on the adjacency relation between the nodes (mobile support stations), by considering a suitable neighbourhood distance. The centre cluster, whose nodes are almost equidistant from the multicast group members, is computed to construct a shortest multicast tree that passes through the centre cluster and reliable routers among all the group members. A Kohonens self-organising-map neural network has been used for clustering. Hopfield neural networks are used to construct a multicast tree with a minimum number of links which passes through the nodes that belong to centre cluster. In addition, the tree is constructed as and when the member(s) moves. This scheme should construct a reliable multicast tree and minimise recomputation time of the multicast tree as and when the multicast route is updated when the mobile hosts change their access point due to mobility. The computational power of the proposed neural-network-based multicast routing algorithm is demonstrated through simulation. The algorithm is also tested for mobility of the participating mobile hosts. The proposed work facilitates a possible multicast routing algorithm for future high-speed mobile networks.

1 Introduction

The increased demand for multimedia conferencing and for users to enjoy the benefits of mobility while having access to the Internet have motivated research in the area of multicasting in mobile networks, in which the networks must be adequately equipped (with supporting systems) to handle the multicast communication. Performing multicast in a mobile network with mobile hosts introduces additional concerns to the already existing difficulties of multicasting in static networks, such as multicast routing, group management etc. These new problems are caused mainly by the movement of the mobile hosts, but are also due to the quality of the wireless connection between the mobile host and the fixed network and due to resource constraints in the mobile computing environment.

Typically, multicast is a mechanism for efficient one-to-many communication, in which the source transmits a data packet, and the network performs the task of delivering that packet to the multiple destinations. By eliminating multiple transmissions, multicast results in significant savings in source host processing and network bandwidth.

$1 \times N$ multicasts, i.e. a single sender to N receivers, will probably be the most commonly used form of multicasting as it can be used by a variety of applications, such as transmission of news programmes, distance learning, weather reports, on-demand applications etc. Some of the features that such an algorithm for multicasting must have are:

- (a) Constructing a $1 \times N$ multicast tree from a source to N users,
- (b) Establishing the tree as quickly as possible,
- (c) Computing the multicast tree as and when required to hide the mobility of the host from applications,
- (d) An adaptation scheme for addition and deletion of participating hosts in a multicast group, and
- (e) Ensuring the end-to-end reliable multicast message delivery.

A typical topology of a mobile network may be treated as an undirected graph $\Psi = (V, E)$; where V is the set of nodes, representing the mobile support stations (MSSs), E is the set of edges representing the wired/wireless links connecting the MSSs, and a set $A \subseteq V$, representing the MSSs to which participating mobile hosts of the multicast group are connected. Now the problem is to find a multicast tree that connects source and all the other MSSs that belong to set A , with a minimum number of links.

In this paper, we propose a reliable multicast routing algorithm in mobile networks by taking the reliable MSSs. The routing algorithm is aimed at constructing a reliable

multicast tree that connects all the multicast group members. Further, the method is to establish a multicast tree when there is a change in location of the group members due to mobility. We employ a neural-network-(NN-) based multicast routing algorithm which includes three steps:

Step 1: Divide the mobile network into a set of clusters of nodes based on the adjacency relation between the nodes (i.e. MSSs), by considering a suitable neighbourhood distance. The centre cluster, whose nodes are almost equidistant from the multicast group members, is computed.

Step 2: Passing through the centre cluster, a set of paths to each destination node is computed from the source node by considering the reliable nodes.

Step 3: A reliable multicast tree is constructed from these sets of paths using a Hopfield neural network (HNN).

2 Mobile-network model

A typical mobile-network environment consists of cells (see Fig. 1), each of which is serviced by a base station (BS) or MSS. The MSS provides a connection end-point for the roaming mobile hosts. The MSSs are interconnected by either wired or wireless media. It is assumed that all MSSs act as multicast supporting routers and are capable of subscribing to multicast groups. A set of reliable MSSs is considered to select a path for constructing a multicast tree.

For example, if a shortest path is to be selected between the mobile hosts X and Y which are currently attached to the MSSs servicing the cells 1 and 3, respectively (see Fig. 1), the shortest path (in terms of number of hops) between these two MHs is 1–2–3. If the servicing of MSS in the cell 2 is unreliable, the alternative paths, 1–4–5–3 or 1–6–7–3 are considered to be the reliable shortest paths (shown by broken lines in Fig. 1) between the MHs X and Y.

Fig. 1 Mobile-network model

3 Related work

Some of the work on dynamic routing for topological changes in mobile networks, multicast routing algorithms and neural-network-based routing algorithms is briefly explained in this Section.

Distributed algorithms for mobile wireless networks are designed for generating loop-free routes in networks with

frequently changing topology [1], but they exhibit instability if the network becomes partitioned and also in the network partitions that are separated from the desired destinations. The algorithms are distributed, dead-lock-free and maintain routes to each desired destination which are loop-free at all times [2].

A low-cost multicast routing algorithm in wireless mobile networks is suggested to reduce the overall cost of the multicast tree and the latency for join and leave operations when mobile hosts hand-off from one cell to another [3]. In [4], a virtual-connection tree-based scheme is given to deal with the mobile-host mobility. In this scheme, a set of adjacent cells is grouped into a cell cluster. When a call is admitted, the scheme pre-establishes connections between a root switch and each base station in the cell cluster.

A global-state routing scheme is proposed for mobile networks [5], where nodes exchange vectors of link states among their neighbours to maintain a global knowledge of the network topology and optimise their routing decision locally. The impact of node mobility and wireless communication on routing system design are discussed in [6], based on the set of techniques employed for routing in mobile networks.

A number of methods have been proposed for centre location in a multicast tree construction for CBT protocols [7], i.e. a centre specific tree for multiple sources and receivers in a multicast group. Some centre-location algorithms are:

- (i) Maximum centred tree algorithm [8], which picks the node with the lowest maximum distance to any group members as centre node;
- (ii) Average centred tree algorithm, which picks the node with the lowest average distance to all group members; and
- (iii) Diameter-centred tree algorithm [8] which finds the node which is the midpoint of the lowest maximum diameter, defined as the sum of the distances to the two furthest nodes.

The tournament based centre-location protocol [9] runs a tournament between the nodes to determine a centre in the method.

A minimum-delay routing algorithm is proposed [10] using a neural network to find a set of alternative routes for each commodity to distribute optimally and so reduce the time delay a packet encounters. An improved neural heuristic for multicast routing is proposed and the routing is achieved by constructing a minimum-cost tree that spans the source and destinations for multipoint communication [11]. In [12, 13], the authors developed a NN-based shortest-path algorithm, by means of which the implementation of the optimal-routing problem was discussed.

4 Proposed multicast routing algorithm

We propose a routing algorithm for constructing a reliable multicast tree in mobile networks. The description of the algorithm is given in three steps.

Step 1. Finding the centre cluster for multicast group members: This step discusses the computation of a centre cluster, whose nodes are almost equidistant from the group members. The centre cluster for the multicast group members is computed by dividing (partitioning) the mobile network into a set of clusters of nodes based on the adjacency relation between nodes with suitable neighbourhood distance. The distance is fixed based on the size of the network topology and the number of clusters to be formed. The centre cluster for the set of clusters containing the

multicast group members is computed by considering the shortest distance between them. This centre-cluster concept will minimise the number of links used for constructing a multicast tree.

The Kohonen self-organising neural network (KNN) model is used for clustering, which has the capability to group the similar featured input patterns into clusters. The input patterns are the rows of the adjacency matrix of the given network topology, and are used as training patterns to the KNN for clustering. The pattern with similar features have been grouped into clusters. Algorithm 1 describes the computation of the centre cluster for multicast group members. The KNN model used for clustering is described in Section 5.1.

Step 2. Finding a set of node-pair paths: This step involves computation of a set of paths from the source to each destination group member by considering the reliable nodes that passes through the centre cluster.

Given an undirected graph $\Psi = (V, E)$ for the topology of a mobile network, where V is the set of vertices or nodes representing the MSSs and E is the set of edges representing the wired/wireless links, let $A \subseteq V$ be a set containing the nodes that belongs to the multicast group, where $A = \{v_1, v_2, \dots, v_N\}$, and N is the number of participating nodes. Assume that the set of multicast group nodes (A) and reliability of each node in V are given at the instant when the routing algorithm is initiated.

Here, define a node-pair set $NPS = \{(v_1, v_2)_1, (v_1, v_3)_2, \dots, (v_1, v_N)_{N-1}\}$, for N nodes that belongs to multicast group and v_1 is the source node of the multicast group. The elements of the set NPS are the node-pairs and the number of elements in the set is $N-1$, under the assumption that one or more paths exist between each node-pair through the centre cluster. The reliable nodes are considered during path selection. Algorithm 2 describes the computation of a set of paths for each node-pair that cross no more than H hops (or MSSs) by avoiding unreliable nodes.

Step 3. Construction of a multicast tree: The task involves the selection of a single path from the set of paths between each node-pair. We formulate the task into an optimisation problem, and before formulating the problem, we define the following notations: $(v_1, v_d)_i$ is i th node pair of the multicast group, v_d is destination node, and $d \in [2, N]$, $NP(i)$ is the number of paths of i th node-pair $(v_1, v_d)_i$, which can be fixed based on the requirement,

$$R_{ij} = \text{the } j\text{th path of } i\text{th node-pair } (v_1, v_d)_i,$$

$$X_{ij} = \begin{cases} 1 & \text{if path } R_{ij} \text{ is chosen;} \\ 0 & \text{if path } R_{ij} \text{ is not chosen;} \end{cases}$$

From the set of paths connecting each node-pair $(v_1, v_d)_i$ where $d = 2, 3, \dots, N$ and $i = 1, 2, \dots, (N-1)$, select a single path for 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 minimised. The formulation of the objective function for this optimisation problem is as follows:

$$\text{Minimise } \frac{1}{2} \sum_{i=1}^{N-1} \sum_{k=1}^{N-1} \sum_{j=1}^{NP(i)} \sum_{l=1}^{NP(k)} f(R_{ij}, R_{kl}) X_{ij} X_{kl},$$

$$\text{for } (i, j) \neq (k, l) \quad (1)$$

where $f(R_{ij}, R_{kl}) = |R_{ij}| + |R_{kl}| - |R_{ij} \cap R_{kl}|$

$|R_{ij}|$ = the number of links in R_{ij} path

$|R_{ij} \cap R_{kl}|$ = the number of common links shared by paths R_{ij} and R_{kl} .

The objective is to minimise the function in (1), subject to the constraint that a single path is chosen for each node-pair (i.e. $X_{ij} = 1$ for exactly one value of j for each value of i) and in total $N-1$ paths are selected. Here, the objective function corresponds to the number of links of all selected path from the set of paths to each node pair and the number of links between the node-pair of the selected paths. This optimisation problem is solved by using a HNN approach and is described in Section 5.2. Algorithm 3 summarises the construction of a multicast tree for a given set of paths to each node pair of the group members.

4.1 Algorithms

In this Section we present algorithms 1–3 for the three steps of the multicast routing technique.

4.1.1 Algorithm 1. Finding the centre cluster for multicast group members:

Begin

Step 1. Divide the mobile network into clusters of nodes based on the adjacency relation between the nodes. (rows of the adjacency matrix are used as a training patterns to KNN for clustering);

Step 2. The centre cluster, among the set of clusters to which the multicast group member belongs, is identified;

Step 3. Output the centre nodes that belongs to the centre cluster;

End.

4.1.2 Algorithm 2. Computation of set of paths:

We employ the recursive algorithm 2, to find all possible paths between the node pair that cross no more than a specified number of hops by avoiding unreliable nodes. The paths are selected to pass through the centre cluster. This algorithm performs a tree-search technique until a destination node is reached.

The following symbols are used: H = number of hops (or MSSs), v_1 = source node, v_d = destination node, v_I = intermediate node of the network, and $Path_i$ = output path set generated for node pair i , initially set to be an empty set Φ .

Begin

For each node-pair $(v_1, v_d)_i$, $i = 1$ to $(N-1)$ do /* $d = 2, 3, \dots, N$, */

$v_I = v_1$; $L = 0$; $Path_i = \Phi$;

CALL Procedure PATH(input: v_I, v_d , Output: $Path_i$)

End

Procedure PATH(input: v_I, v_d , Output: $Path_i$)

Begin

if ($v_I = v_d$)

then output $Path_i$ **if** $Path_i$ **contains node(s) of centre cluster; return;**

else

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

begin

$L = L + 1$;

add v_I to $Path_i$;

for each neighbouring node v_z of node v_I **do**

PATH ($v_z, v_d, Path_i$)

$L = L - 1$;

end;

End;

The specific path set is found for each node pair passing through the centre cluster, when the algorithm terminates.

4.1.3 Algorithm 3. Construction of a reliable multicast tree:

Begin

Step 1. Select a single path for each node pair such that the objective function specified in (1) is minimised, (this is an optimisation problem solved by using the HNN described in Section 5.2).

Step 2. A reliable multicast tree is constructed from the set of selected paths from source to destination group members.

End.

4.2 Example

Consider a mobile network with five MSSs represented by a five-node network shown in Fig. 2. Given a multicast group, $A = \{1, 4, 5\}$ where the number of group members $N = 3$, the corresponding node-pair set $NPS = \{(1, 4)_1; (1, 5)_2\}$ and the number of paths for each node-pair, $NP(i) = 3$; $i = 1, 2$, with hop count $H = 3$. Considering all the nodes to be reliable and the nodes of the centre cluster 2, 3 and 4, then the path set for each node-pair is 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: $\{R_{21} = (1, 3, 5), R_{22} = (1, 4, 5), R_{23} = (1, 2, 5)\}$

Now the routing problem is the selection of one path from the set of paths for each node-pair that gives a minimum value for the objective function given in (1). The path selected for node-pair $(1, 4)$ is $R_{11} = (1, 4)$; and for node pair $(1, 5)$ is $R_{22} = (1, 4, 5)$, whose minimum objective-function value is 2. The constructed multicast tree is shown in Fig. 2 by thick lines.

Fig. 2 Mobile network with three multicast group nodes 1, 4, 5

5 Neural-network model for clustering mobile network and multicast-tree construction

In this Section, the principle of construction of a reliable multicast tree using a neural-network model is described. We present a KNN model and its learning algorithm for clustering based on the adjacency relation between the nodes of the network topology and a Hopfield neural-network model for the construction of a reliable multicast

tree. After a brief description about the HNN, we present the basic energy function, derivation of the connection weight matrix developed for the HNN model and the equation of motion to satisfy the objectives and constraints of the problem. Finally, an algorithm to construct a reliable multicast tree is described.

5.1 Kohonen self-organising neural network

A Kohonen self-organising neural network is a competitive, feed-forward type and unsupervised training neural network, where there is no external expected output or the target output is presented for the learning process. In this process of learning, the network is able to discover statistical regularities in its input space and automatically develops different group or cluster behaviour to represent different classes of inputs [15].

5.1.1 Selection of neurons and KNN architecture:

Kohonen self-organising neural networks have two layers, an input layer and an output layer known as the competitive layer. Each input neuron is connected to every neuron on the output layer, the latter being organised as a two-dimensional grid. Figure 3 shows a typical example of a Kohonen neural network with three input and 16 output neurons.

Fig. 3 Typical Kohonen neural network model with three input and 16 output neurons

The input layer in a Kohonen network 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 vertices, then the number of input neurons is $|V|$.

However, the neurons on the output layer find the organisation of relationships among input patterns, which are classified by the competitive neurons and can organise 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. We have chosen 50 output neurons for a 25-node network topology with a suitable neighborhood value.

5.1.2 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: $I = (I_1, I_2, \dots, I_{|V|})$, where there are $|V|$ input neurons in the input layer. Now suppose there are M neurons in the output (competitive) layer, and let O_j be the j th output neuron. Then the whole of the output layer will be:

$$O = (O_1, O_2, \dots, O_M)$$

For each output neuron O_j there are $|V|$ incoming connections from the $|V|$ input neurons. Each connection has a weight value W . So, for any neuron O_j on the output layer, the set of incoming connection weights is

$$W_j = (W_{j1}, W_{j2}, \dots, W_{j|V|})$$

The Euclidean-distance value D_j of a neuron O_j in the output layer, whenever an input pattern I is presented at the input layer, is

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

The competitive output neuron with the lowest distance at this stage is the closest to the current input pattern, called the winning neuron O_c . After the winning neuron is identified, the next step is to identify the neighbourhood around it, which is simply the set of output neurons which are close to that winning neuron. Figure 4 shows an example of a neighbourhood around a winning neuron, which is just the set of neurons that are within the square that is centred on the winning neuron O_c .

Fig. 4 A 4×4 Kohonen grid showing the size of neighbourhood influence around neuron O_c

The size of the neighbourhood h starts with a big enough size and decreases with respect to learning iterations such that $h_t = h_0(1-t/T)$ where, h_t denotes the actual neighbourhood size, h_0 denotes the initial neighbourhood size, t denotes the current learning epoch and T denotes the total number of epochs to be done, where an epoch is the time taken for network to sweep through all the rows of the adjacency matrix as an input data pattern once. During the learning stage, the weights of the incoming connections of each competitive neuron in the neighborhood of the winning one are updated, including the winning neuron.

The overall weights update equation is:

$$W_j^{new} = W_j^{old} + \alpha(I - W_j^{old})$$

if unit O_j is in the neighbourhood of winning unit O_c , where α is the learning rate parameter.

Typical choices are in the range $[0.2 \dots 0.5]$.

A learning algorithm for clustering is summarised as follows.

Begin

Step 1. Assign random values to the connection weights W in the range $(0, 1)$;

Step 2. Select an input pattern from the adjacency matrix of the network topology, (row of an adjacency matrix);

Step 3. Determine the winner output neuron O_c using (2);

Step 4. Perform a learning step affecting all neurons in the neighborhood of O_c by updating the connection weights;

$$W_j^{new} = W_j^{old} + \alpha(I - W_j^{old})$$

Update α , h_t and continue with step 2 until no noticeable changes in the connection weights are observed for all the input patterns.

End

5.2 Hopfield neural network model

A Hopfield neural network (HNN) is a single-layer network with feedback consisting of set of neurons which are fully interconnected by weighted connections (see Fig. 5).

Fig. 5 Neural-network model representing the path neurons for the network in Fig. 2

5.2.1 Selection of neurons and HNN architecture: For the HNN model, we define one neuron for each path between every node pair. Hence, the number of neurons η for N nodes in a multicast group is

$$\eta = \sum_{i=1}^{N-1} NP(i)$$

For example, Fig. 5 shows the HNN model for the network shown in Fig. 2. In total there are six neurons, which are connected by connection-weight matrix T , (see Fig. 5), where $T_{ij,kl}$ is a connection weight between the neurons representing the path R_{ij} and R_{kl} , for $i, j, k, l = 1, 2, 3$.

The connection-weight matrix T is $\eta \times \eta$ and symmetric. The elements of T are connection weights $T_{ij,kl} = T_{kl,ij}$ and $T_{ij,ij} = 0$.

The strengths of these connections are chosen to encourage the sharing of links by many paths of different node-pairs and the correct number of path activations (i.e. one neuron is turned on per node-pair). Here we represent each neuron by double indices (i.e. neuron ij represents the j th path between the node-pair i).

5.2.2 Dynamics of HNN model: The dynamics of HNN can be described by a general equation of motion (energy function for HNN) and may be written in terms of the connection weights, bias input and neuron output as follows [16].

$$E_{total} = -\frac{1}{2} \sum_{i=1}^{N-1} \sum_{k=1}^{N-1} \sum_{j=1}^{NP(i)} \sum_{l=1}^{NP(k)} T_{ij,kl} X_{ij} X_{kl} - \sum_{i=1}^{N-1} \sum_{j=1}^{NP(i)} X_{ij} B_{ij} \quad (3)$$

for $(i, j) \neq (k, l)$

where:

$T_{ij,kl}$ = strength of the connection weight between neurons ij and kl ,

$NP(i)$ = number of paths between the i th node-pair $(v_1, v_d)_i$, for $i = 1, 2, \dots, (N-1)$,

B_{ij} = bias value applied to neuron ij ,

X_{ij} = ij th neuron output,

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 output and bias value input to each neurons (bias-values are set to zero in our case).

5.2.3 Problem constraints and energy function: The goal is to minimise the total energy function involving the objective function in (1) and constraints. The problem constraints C_1 and C_2 are to activate no more than one path for each node-pair and C_3 is to activate a total of exactly $(N-1)$ paths in the network, and are given in (4), (5) and (6), respectively (which must be equal to zero when the constraints are satisfied).

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

$$C_1 = \frac{1}{2} \sum_{i=1}^{N-1} \sum_{j=1}^{NP(i)} \sum_{l=1, l \neq j}^{NP(i)} X_{ij} X_{il} = 0 \quad (4)$$

To activate exactly one path per node-pair:

$$C_2 = \frac{1}{2} \sum_{i=1}^{N-1} \left(\sum_{j=1}^{NP(i)} X_{ij} - 1 \right)^2 = 0 \quad (5)$$

To activate a total of exactly $(N-1)$ paths in the network:

$$C_3 = \frac{1}{2} \left(\sum_{i=1}^{N-1} \sum_{j=1}^{NP(i)} X_{ij} - (N-1) \right)^2 = 0 \quad (6)$$

The total energy function E_{total} that satisfies the objective function (1) and constraints for the problem is the sum of

(1), (4), (5) and (6) as follows, i.e.

$$E_{total} = \frac{A}{2} \sum_{i=1}^{N-1} \sum_{k=1}^{N-1} \sum_{j=1}^{NP(i)} \sum_{l=1}^{NP(k)} f(R_{ij}, R_{kl}) X_{ij} X_{kl} + \frac{\gamma_1}{2} \sum_{i=1}^{N-1} \sum_{j=1}^{NP(i)} \sum_{l=1, l \neq j}^{NP(i)} X_{ij} X_{il} + \frac{\gamma_2}{2} \sum_{i=1}^{N-1} \left(\sum_{j=1}^{NP(i)} X_{ij} - 1 \right)^2 + \frac{\gamma_3}{2} \left(\sum_{i=1}^{N-1} \sum_{j=1}^{NP(i)} X_{ij} - (N-1) \right)^2 \quad (7)$$

In (7), the term A minimises the total number of uncommon links of the paths in every node-pair of multicast group. Terms γ_1 and γ_2 are zero, if no more than one path exists between the node-pair. Term γ_3 is zero if exactly $(N-1)$ paths exists in the solution. The constant parameters A , γ_1 , γ_2 and γ_3 are chosen to best fit the problem during simulation. The values chosen for these parameters are assumed to be positive constant in our case.

Looking at the problem constraints and corresponding terms in the energy equation which must be equal to zero when constraints are satisfied, this will force the energy function to one of the corner of a hypercube, which will have minimum value and is taken as a feasible solution for the above problem with proper mapping from the neural-network output.

5.2.4 Determination of connection weights: The strength of the connection weights is derived by comparing the objective function and constraints in (7), with the general energy-function equation (3) for HNN. The resulting connection weight of neuron ij and kl is

$$T_{ij,kl} = A f(R_{ij}, R_{kl}) (1 - \kappa_{ik}) - \gamma_1 \kappa_{ik} (1 - \kappa_{jl}) - \gamma_2 \kappa_{ik} - \gamma_3$$

$$\text{where } \kappa_{xy} = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \quad \text{Kronecker delta function}$$

(8)

5.2.5 Equation of motion: As the HNN evolves from an initial state, it follows a trajectory over which the energy function decreases monotonically. The evolution of the inputs at each neuron is characterised by the equation [16].

$$\frac{du_{ij}}{dt} = -\frac{u_{ij}}{\tau} - \frac{\partial E_{total}}{\partial X_{ij}} \quad (9)$$

The equation of motion for an HNN with symmetric connection weights (i.e. $T_{ij,kl} = T_{kl,ij}$) will always guarantee convergence to a stable state [16]. As the system evolves from an initial state, the energy function decreases monotonically until equilibrium at a (local) minimum is reached. Here the final state depends on the initial state at which the system evolution has started, the objective function and constraints.

The equation of motion can be expressed in iterative form as follows:

$$\begin{aligned}
u_{ij}(t + \Delta t) = & u_{ij}(t) - \Delta t \frac{u_{ij}}{\tau} - \Delta t A \sum_{k=1, k \neq j}^{N-1} \sum_{l=1}^{NP(k)} f(R_{ij}, R_{kl}) X_{kl} \\
& - \Delta t \gamma_1 \sum_{l=1, l \neq j}^{NP(i)} X_{il} - \Delta t \gamma_2 \left(\sum_{j=1}^{NP(i)} X_{ij} - 1 \right) \\
& - \Delta t \gamma_3 \sum_{i=1}^{N-1} \sum_{j=1}^{NP(i)} X_{ij} - (N-1)
\end{aligned} \quad (10)$$

Some of the issues that arise when these equations are simulated in software involve the choice of the coefficients A , γ_1 , γ_2 , γ_3 and Δt . Usually these values are chosen to ensure that an energy function in (7), which is characterised by the presence of valleys in the energy surface, has only one point with a graceful descent along the energy surface [16, 14] so that the network can converge to a valid solution of high quality. Here the step size Δt is chosen in 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 a large value of Δt or extremely slow convergence due to a small value of Δt . Algorithm 4 describe the steps to find a reliable multicast tree.

5.2.6 Algorithm 4. HNN-based reliable multicast tree:

Begin

Step 1. Find the centre nodes for the multicast group members using clustering based on the adjacency relationship between the nodes using KNN, *algorithm 1*;

Step 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 centre node(s); *algorithm 2*;

Step 3. Design an HNN model for the path set;

(a) Randomise the initial input to neurons, Count = 0;

(b) Use equation of motion (10) and sigmoid function to update the neuron output;

(c) Iterate step 3b until (HNN converges to a stable state or count = 5000);

Step 4. Construct a reliable multicast tree from the selected reliable path for each node-pair;

End

6 Simulation

Simulation is carried out to evaluate the performance of the proposed multicast routing in mobile-networks. In the simulation, we consider several graphs to represent the mobile network topology with edges representing the wired or wireless links connecting the MSSs representing the vertices of the graph. We consider a partially connected mobile-network topology with some percentage among the total nodes as participating nodes of a multicast group.

The proposed multicast routing algorithm is used for computing the multicast trees for the generated graphs. The experiment is carried out by considering 16-, 25- and 30-node networks with a change in topology. Simulation is carried out by considering different number of participating

nodes (a minimum of three nodes up to a maximum of 50% of the total number of nodes in a given network topology).

Given the initial neuron input values u_{ij} at time $t = 0$; the time evolution of the state of the HNN is simulated by solving (10) using the HNN model. When an HNN converges to a stable state, i.e. no more updating of neuron output with further iterations, the output X_{ij} ($= 1$) of the neuron denotes that the j th path of the i th node pair (v_1, v_d) is chosen. Similarly for all the other node-pairs.

The values chosen for the energy-function coefficients of the HNN model in the simulation are as follows. $A = 18.5$, $\gamma_1 = 6.5$; $\gamma_2 = 3.5$, $\tau = 1.0$ and $\Delta t = 0.005$; and, for the KNN model, the neighbourhood size $h_0 = 6$ and the learning rate $\alpha = 0.3$ are chosen. Simulation experiments well carried out on a Pentium III PC with the Linux operating system, using C++ programming language.

The performance of the algorithm in constructing the multicast tree is taken with respect to the number of multicast group nodes, the number of iterations used for constructing a reliable multicast tree. Also, the mobility of multicast group members is considered in constructing the tree as and when the member(s) moves. Here we consider the mobility in terms of the number of group members changing their location. During simulation, the output of the HNN-used for constructing a multicast tree is maintained and used as the initial value for constructing a multicast tree next time, when the member(s) moves. This method of using initial value for the HNN from the previous-case results has taken a smaller number of iterations to converge. Hence, once the tree is constructed at the initial stage, the reconstruction of the tree is fast, when certain percentage of participating group member(s) change their locations.

6.1 Results

Simulation results, to evaluate the performance of the proposed algorithm, are shown in Figs. 6 to 8. The average number of iterations required for converging to an efficient route with respect to the number of multicast nodes is shown in Fig. 6. The results are taken for different numbers of alternative paths between each node pair of a multicast group. The x-axis is the number multicast-group nodes normalised with respect to the total number of nodes in the network topology. From the graph, it is clear that, as the number of multicast nodes increases, the number of the iterations required for convergence to an efficient route

Fig. 6 Average iterations for constructing a multicast route with respect to number of alternative paths

Fig. 7 Time required for the algorithm to converge to feasible solution

Fig. 8 Percentage mobility with respect to the iterations for convergence

increases. However, the number of iterations for convergence decreases with a decrease in the number of alternative paths for each node-pair.

Figure 7 shows the time required for the algorithm to converge to a feasible solution with respect to the number of multicast nodes. From the graph it is clear that, as the number of multicast nodes increases, the convergence time increases. Figure 8 shows a performance evaluation of the algorithm for mobility, where the participating multicast nodes change their position. The graph is plotted for the percentage of mobile hosts changing their location and establishing new multicast trees with respect to the number of iterations. Results show that, as the percentage of mobility increases, the number of iterations required to construct a multicast tree is high, in turn requiring more time for convergence.

7 Conclusions

In this paper we have proposed a neural-network- (NN-) based multicast routing algorithm for constructing an efficient multicast tree that connects the participants of a multicast group. The method computes a reliable multicast route (tree), and also selects a reliable route whenever there is a change in location of group members due to mobility.

This work presents the capability of neural networks as a computational tool for solving a constrained optimisation 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 a reliable multicast tree is low for small numbers of multicast-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, since a hardware-implemented neural network can achieve an extremely high response speed.

8 References

- Corson, M.S., and Ephremides, A.: 'A distributed routing algorithm for mobile radio networks'. Proc. IEEE Military Communications Conf., Boston, MA, 1989
- Corson, M.S., and Ephremides, A.: 'A distributed routing algorithm for mobile wireless networks', *Wirel. Netw.*, 1995, **1**, (1), pp. 61–81
- Alrabiah, T., and Aljadhari, A.: 'Low-cost multicast routing in wireless mobile networks'. Proc. IEEE Wireless Communications and Networking Conf., Chicago, IL, USA, Sept. 2000, pp. 1467–1471
- Acampora, A., and Naghshineh, M.: 'Control and quality-of-service provisioning in high speed microcellular networks', *IEEE Pers. Commun.*, 1994, **1**, (2), pp. 36–42
- Chen, T.W., and Gerla, M.: 'Global state routing: A new routing scheme for ad-hoc wireless networks'. Proc. IEEE Int. Conf. on Communications, ICC'98, Atlanta, GA, June 1998, pp. 171–175
- Ramanathan, S., and Steenstrup, M.: 'A survey of routing techniques for mobile communications networks', *Mobile Netw. Appl.*, 1996, **1**, (2), pp. 89–104
- David, G.T., and Ravishanker, C.V.: 'Distributed center-location algorithms: proposals and comparisons'. Proc. IEEE INFOCOM, 1996, pp. 75–84
- Wall, D.W.: 'Mechanisms for broadcast and selective broadcast'. PhD thesis, Stanford University, 1982
- Shukla, S.B., Boyer, E.B., and Klinker, J.E.: 'Multicast tree construction in network topologies with asymmetric link loads'. Technical report NPS-EC-94-012, Naval Post Graduate School, Sept. 1994
- Feng, G., and Douligeris, C.: 'A neural network method for minimum delay routing in packet switched networks', *Comput. Commun.*, 2001, **24**, pp. 933–941
- Gelenbe, E., Ghanwani, A., and Srinivasan, V.: 'Improved neural heuristics for multicast routing', *IEEE J. Select. Areas Commun.*, 1997, **15**, (2), pp. 147–155
- Mustafa, K., Ali, M., and Kamoun, F.: 'Neural networks for shortest path computation and routing in computer networks', *IEEE Trans. Neural Netw.*, 1993, **4**, (6), pp. 941–954
- Venkataram, P., Ghosal, S., and Vijay Kumar, B.P.: 'Neural network based optimal routing algorithm for communication networks', *Neural Netw.*, 2002, **15**, pp. 1289–1298
- Feng, G., and Douligeris, C.: 'On the convergence and parameter relation of discrete-time continuous-state Hopfield networks with self-interaction neurons', *IEICE Trans. Fundam. Electron. Commun. Comput.*, 2001, **E84-A**, (12), pp. 3162–3173
- Haykin, S.: 'Neural networks: A comprehensive foundation' (Macmillan College Publishing Co., New York, USA, 1995, 2nd edn.)
- Hopfield, J.J., and Tank, D.W.: 'Neural computation of decisions in optimisation problems', *Biol. Cybern.*, 1985, **52**, (1), pp. 141–152