Kabat MR, Patel MK, Tripathy CR. A heuristic algorithm for core selection in multicast routing. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 26(6): 954–961 Nov. 2011. DOI 10.1007/s11390-011-1192-x

# A Heuristic Algorithm for Core Selection in Multicast Routing

Manas Ranjan Kabat, Manoj Kumar Patel, and Chita Ranjan Tripathy

*Department of Computer Science and Engineering, Veer Surendra Sai University of Technology, Burla 768018, India*

E-mail: manas_kabat@yahoo.com; patel.mkp@gmail.com; crt.vssut@yahoo.com

Received September 23, 2010; revised May 19, 2011.

**Abstract** With the development of network multimedia technology, more and more real-time multimedia applications need to transmit information using multicast. The basis of multicast data transmission is to construct a multicast tree. The main problem concerning the construction of a shared multicast tree is selection of a root of the shared tree or the core point. In this paper, we propose a heuristic algorithm for core selection in multicast routing. The proposed algorithm selects core point by considering both delay and inter-destination delay variation. The simulation results show that the proposed algorithm performs better than the existing algorithms in terms of delay variation subject to the end-to-end delay bound. The mathematical time complexity and the execution time of the proposed algorithm are comparable to those of the existing algorithms.

**Keywords** QoS routing, multicast routing, delay-variation, end-to-end delay

## 1 Introduction

The rapid development of network technology and multimedia technology gradually enables network multimedia such as video conferencing, distance education and coordinative work to become the mainstream Internet activities. Multicasting is very convenient for these services, which usually have strict requirements of quality of service (QoS) parameters such as end-to-end delay, delay variation, loss, cost, throughput and so on. Therefore, it is an important and urgent research problem to set up multicast routing with high QoS. The central problem of QoS multicast routing is to set up a multicast root tree that can satisfy certain QoS parameters. The prime problem of constructing a shared multicast tree is to determine the position of the root of the shared tree. This is referred to as the center selection problem. The center is called corepoint in core-based tree (CBT)[1] or Rendezvous Point (RP) in PIM-SM[2]. This problem was first proposed by Wall[3]. The selection of core directly affects the performance of multicast. A poor selection may lead to many performance problems such as high cost, high delay and high congestion. Therefore, it is very important to select a good core to have effective multicast. However, the core selection is an NP-complete problem[1,3-4], which needs to be solved using heuristic algorithm. Researchers have already proposed several solutions to this problem[5-15]. However, the problem of finding a better or the best core node has not yet been completely solved.

There are many well-known approaches to construct core-based multicast tree satisfying delay and delay-variation constraints. These are delay variation multicast algorithm (DVMA)[12], delay and delay variation constrained algorithm (DDVCA)[13], AKBC (Ahn, Kim, Bang, Choo) algorithm[14] and AKC (Ahn, Kim, Choo) algorithm[15]. The issues of minimizing multicast delay variation problem under the multicast end-to-end delay constraints are defined and discussed in [12]. This problem is referred to as the delay and delay variation bounded multicast tree (DVBMT) problem. In DVMA[12], it is assumed that the complete topology is available at each node. The algorithm starts with a spanning tree satisfying the delay constraint only, which may not include some destination nodes. Then the algorithm searches through the candidate paths satisfying the delay and delay-variation constraints from a non-tree member node to any of the tree nodes. It works on the principle of $k$-shortest paths to the group of destination nodes concerned. If these paths do not satisfy a delay constraint, then it may find a longer path, which is a shortfall of DVMA. The spanning tree built by DVMA satisfies both delay and delay-variation constraints. However, due to the very high time complexity, it does not fit in modern high-speed computer network environment.

The other approach to the DVBMT problem is DDVCA[13]. It first calculates the delay of the least

Regular Paper

delay paths from each destination node to all the nodes. The node that has the minimum delay-variation is selected as the core node. The source node sends a single copy of the message to the core node. Then the core node forwards the message to all the receivers through the minimum delay path. In comparison to DVMA, DDVCA possesses a significant lower time complexity, i.e., $O(mn^2)$ where $m$ represents the number of destination nodes and $n$ the total number of nodes in the computer network.

Another efficient core selection algorithm has been proposed by KIM *et al.*[6] to produce a core-based multicast tree under delay and delay-variation constraints. First, this algorithm finds a set of candidate core nodes that have the same associated multicast delay-variation for each destination node. Then, the final core node is selected from the set of candidate core nodes that has the minimum potential delay-variation. A variation of this algorithm is proposed in [14], which uses the MODE function to find the exact location of the core. Then the potential delay-variation associated with each candidate core node is found and the node that has the minimum potential delay-variation is considered as the best core node. However, all these algorithms are applied only in the symmetric network environment that has no direction.

Ahn, Kim and Choo[15] proposed an algorithm that constructs a multicast tree with low delay-variation in a realistic network environment that has two-way directions. This algorithm works efficiently in the asymmetric network.

However, these algorithms[6,14-15] select the best core node out of a set of candidate core nodes that have the same associated delay-variation. Therefore, these algorithms are restricted only to selecting the best core node, which may not generate an optimal delay-variation-based multicast tree in many cases.

In our proposed algorithm, we introduce a parameter known as delay variation parameter ($\rho$) to find the set of candidate core nodes. Our algorithm finds the best core node out of a set of candidate core nodes and constructs the multicast tree by connecting the destinations from the core node through the shortest paths and the source to core point through the best path out of $k$-shortest paths. The performance of the proposed algorithm is evaluated through simulations. It is observed the algorithm outperforms the existing algorithms.

The rest of the paper is organized as follows. The mathematical model developed to model a computer network is presented in Section 2. The proposed algorithm and its principle of working followed by the case studies and complexity analysis are presented in Section 3. The simulation results of our algorithm are presented in Section 4. Finally, we present the conclusion in Section 5.

## 2 Mathematical Model

A computer network is modeled as a directed, connected graph $G = (V, E, v_s, M)$, where $V$ is a finite set of vertices (network nodes) and $E$ is the set of edges (network links) representing connection of those vertices. Let $|V|$ be the number of network nodes and $|E|$ the number of network links. The symbol $v_s$ represents the source node and $M$ is the set of destination nodes. The link $e = (v_i, v_j)$ from node $v_i \in V$ to node $v_j \in V$ implies the existence of a link and $e' = (v_j, v_i)$ from $v_j$ to $v_i$. Each link is associated with a positive real value: delay $D(e)$: $E \rightarrow R^+$. The link delay function $D(e)$ is considered to be the sum of queuing delay, transmission delay and propagation delays.

A multicast tree $T(v_s, M)$ is a sub-graph of $G$ spanning the source node $v_s \in V$ and the set of destination nodes $M$. Let $m = |M|$ be the number of multicast destination nodes. Let $P_T(v_s, v_j)$ be a unique path in the tree $T$ from the source node $v_s$ to a destination node $v_j \in M$. The total delay of the path $P_T(v_s, v_j)$ is simply the sum of delay of all links $D(P_T(v_s, v_j)) = \Sigma e \in P_T(v_s, v_j) D(e)$.

The other parameter multicast delay-variation $dv$, is the maximum difference between the end-to-end delays along the paths from the source to any two destination nodes and is defined as follows:

$$dv = \max\Big\{\Big|\sum_{e\in P_T(v_s,v_j)} d(e) - \sum_{e\in P_T(v_s,v_k)} d(e)\Big|, \quad \forall v_j, v_k \in D\Big\}.$$

Thus, based upon the above definition we can now mathematically formulate the multicast routing problem as follows. For a given weighted graph $G = (V, E)$, the source node $v_s \in V$, a destination node set $M \subseteq V - \{v_s\}$, a link-delay function $D(e)$: $E \rightarrow \mathbb{R}^+$, $e \in E$ and a constant $\Delta$, the objective is to determine an optimal multicast tree $T$ such that

$$D(v_s, M) = \max_{v_j \in M} \sum_{e\in P_T(v_s,v_j)} d(e) \leqslant \Delta,$$

$$dv(v_s, M) = \max\Big\{\Big|\sum_{e\in P_T(v_s,v_i)} d(e) - \sum_{e\in P_T(v_s,v_j)} d(e)\Big|, \quad \forall v_i, v_j \in M\Big\} \leqslant \delta.$$

## 3 Proposed Algorithm

In this section, we present our core selection

algorithm to construct a multicast tree that is superior to the existing algorithms[6,13-15]. We discuss the working principle of the proposed algorithm followed by the case studies and complexity analysis.

### 3.1 Description

The pseudo code of the algorithm is shown in Fig.1. The algorithm starts with calculating the least delay path from the source node to each node in graph $G$. Then the proposed algorithm calculates the minimum delay paths from each destination node to all nodes in $G'$ by using Dijkstra's algorithm, where $G'$ is the transpose of the adjacency matrix of $G$. If source is found in the path of a destination node $v_d$ to a node $v_k$ then the candidature of $v_k$ for the core node is cancelled.

The delay-variation parameter introduced in our algorithm is $\rho = dv_{\min} + \lceil stddev(dv)/2.0 \rceil$, where $dv_{\min}$ is the minimum delay-variation and $stddev(dv)$ is the standard deviation of the delay variations associated with the nodes. Then, it selects a set of candidate core nodes that satisfies both end-to-end delay and delay-variation parameter. If there is no candidate node then the algorithm terminates without generating a multicast tree. Otherwise, the algorithm maintains a data structure *pass* for each destination node. If the destination node $v_d$ is visited in the path from the source $v_s$ to a node $v_k$, then the $pass(v_s, v_d, v_k)$ is the difference between the delays of the least delay paths from $v_s$ to $v_k$ and $v_s$ to $v_d$. If the destination node $v_d$ is not in the path from source $v_s$ to a node $v_k$ then $pass(v_s, v_d, v_k)$ is 0. Next, the *pass* value associated with each node $v_i$, i.e., $pass(v_s, v_i)$ is calculated as the maximum of the pass values calculated for each destination. The *compare* values of the candidate core nodes are calculated as equal to their respective *pass* values. The node that has the minimum *compare* value is considered as the best core node. If there are two or more nodes that have the same *compare* value, then the node with minimum delay variation is considered as the best core node. If there is still a tie then the best core node is selected in the first-come-first-serve basis.

To construct a better multicast tree than the existing algorithms, we calculate $k$-least delay paths from source node $v_s$ to the core node $v_c$ subject to delay constraint $\Delta - \max_{v_j \in M} P_{ld}(v_c, v_j)$. Then the multicast tree is constructed by connecting each destination node from the core node through the least delay paths and source to core node through the best path out of $k$-least delay paths. The delay-variation of the multicast tree is calculated after pruning the cycles. The final multicast tree generated is one of the multicast trees having the minimum delay-variation.

```
   Algorithm (G, delay)
01. Begin
02.   T = ∅, candidate = ∅
03. for ∀v_i ∈ V P_ld(s, v_i) = the minimun delay path from source to all nodes in V
04. G' = Transpose(G)
05.   for ∀v_j ∈ M in G'
06.     P_ld(v_j, v_i) is the least delay path from v_j in M to all v_i ∈ V,
07.     if s ∈ P_ld(v_j, v_i), then candidature(v_i) = False
08.   for all v_i ∈ V
09.     max_i = max_{v_j ∈ M}{delay(P_ld(v_j, v_i))}
10.     min_i = min_{v_j ∈ M}{delay(P_ld(v_j, v_i))}
11.     dv(v_i) = max_i − min_i
12.     for all v_i ∈ V
13.       if ((dv_i ⩽ ρ) and delay(s, v_i) + max_i ⩽ Δ)
            &&(candidature(v_i)! = False)
14.           candidate = candidate ∪ {v_i}
15.     for {∀ l ∈ P_ld(s, v_i)|v_i ∈ candidate}
16.       if l = m_k, m_k ∈ M then
17.         pass(s, v_i, m_k) = delay(P_ld(s, v_i))−
            delay(P_ld(s, m_k))
18.       else
19.         pass(s, v_i, m_k) = 0
20.       pass(s, v_i) = max_{m_k ∈ M} pass(s, v_i, m_k)
21.       if candidate = ∅ then print "Tree Construction failed"
22.       for ∀c_i ∈ candidate
23.           compare_i = pass(s, c_i)
24.       c = i, where index i for min{compare_i}
25. if there are two or more candidates with same compare value
    then we consider the candidates with minimum dv value
26. find k paths using k-Bellman Ford Algorithm
27. such that p_k(s, v_c) ⩽ Δ − max_{v_j ∈ M}{P_ld(v_c, v_j)}
28. calculate pass_k(s, v_c) = pass(s, v_c)
29. T = T ∪ {l|l ∈ least delay path from v_c to m_k, m_k ∈ M}
30. for i = 1 to k do {
31.   tempT = T ∪ {l|l ∈ least delay path from s to v_c}
32.   if (pass_k(s, v_c) == 0)
33.       dv = max_i − min_i
34.   else
36.   {
37.     t_max = max{delay(P_ld(v_j, v_i))}
38.     v_j ∈ M and v_j ∉ P_ld(s, v_c)
39.     dv = t_max + pass_k
40.     dv = max_i + pass_k
41.     if (dv < min))
42.       {
43.         min = dv
44.         p = k
45.       }
46.   }
47. }
48. T = T ∪ {l|l ∈ the p-th min delay path from s to v_c}
49. prune links for cycles
50. return
51. End
```

Fig.1. Pseudo code for the proposed algorithm.

### 3.2 Case Studies

For ease of understanding, we present some case studies.

*Case I* ($k = 1$). In this case, we consider the example network shown in Fig.2. The source node is $v_1$ and the set of destination nodes are $\{v_5, v_6\}$, where the number along each edge represents the delay for that edge. The delay bound $\Delta$ is assumed 11.

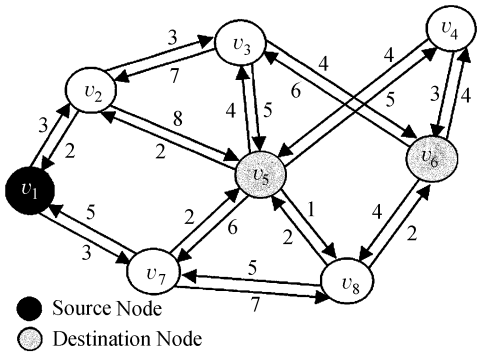


Fig.2. Example network ($\Delta = 11$).

Table 1 shows delay-variation associated with each node. The delay-variation parameter is calculated as 1. The candidate core nodes found are $v_3$ and $v_8$. The parameter $compare_i$ is calculated for each candidate node and the node $v_3$ is found to be the best core node. The multicast tree is generated by connecting the destination nodes $v_5$ and $v_6$ from the core node $v_3$ through the least delay paths and source node $v_1$ to core node $v_3$ through the least delay path. The multicast trees generated by DDVCA and AKC algorithms are shown in Fig.3 and Fig.4 respectively. The final multicast tree generated by our proposed algorithm is shown in Fig.5.

**Table 1.** Selection of Core Node by Our Algorithm for Case I

| | | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
|---|---|---|---|---|---|---|---|---|---|
| Source | $v_1$ | 3 | 3 | 6 | 10 | 5 | 8 | 3 | 6 |
| Pass | $v_5$ | 0 | 0 | 0 | 10 | 5 | 8 | 0 | 0 |
| | $v_6$ | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 |
| Destination | $v_5$ | 5 | 7 | 5 | 4 | 0 | 6 | 2 | 2 |
| | $v_6$ | 8 | 7 | 4 | 3 | 3 | 0 | 5 | 2 |
| $max_i$ | | 8 | 7 | 5 | 4 | 3 | 6 | 5 | 2 |
| $min_i$ | | 5 | 7 | 4 | 3 | 0 | 0 | 2 | 2 |
| $diff_i$ | | 3 | 0 | 1 | 1 | 3 | 6 | 3 | 0 |
| $compare_i$ | | | | 0 | | | | | 1 |

*Case II* ($k = h$). In this case, we consider the example network shown in Fig.6. The source node is $v_1$ and the set of destinations is $\{v_5, v_6\}$, and the number along each edge represents the delay for that edge. The delay bound $\Delta$ is assumed as 11.

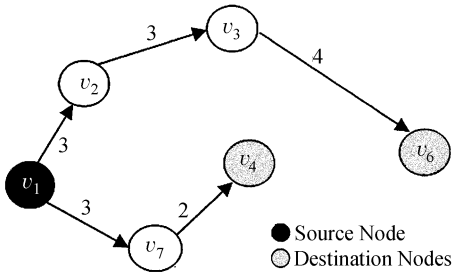


Fig.3. Tree generated by DDVCA, $dv = 5$.

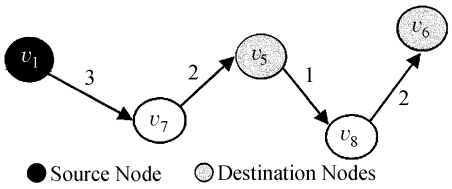


Fig.4. Tree generated by AKC, $dv = 3$.

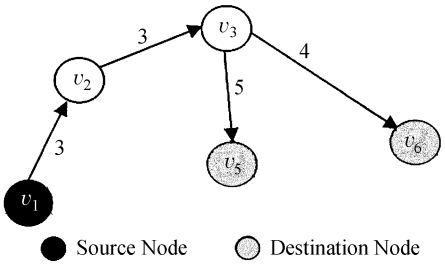


Fig.5. Tree generated by our proposed algorithm, $dv = 1$.

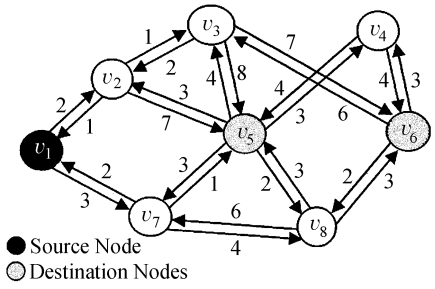


Fig.6. Example network ($\Delta = 11$).

**Table 2.** Selection of Core Node by Our Algorithm for Case II

| | | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ | $v_7$ | $v_8$ |
|---|---|---|---|---|---|---|---|---|---|
| Source | $v_1$ | 0 | 2 | 3 | 7 | 4 | 9 | 3 | 6 |
| Pass | $v_5$ | 0 | 0 | 0 | 3 | 0 | 5 | 0 | 2 |
| | $v_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Destination | $v_5$ | 4 | 5 | 7 | 4 | 0 | 5 | 1 | 3 |
| | $v_6$ | 9 | 8 | 7 | 4 | 5 | 0 | 7 | 3 |
| $max_i$ | | 9 | 8 | 7 | 4 | 5 | 5 | 7 | 3 |
| $min_i$ | | 4 | 5 | 7 | 4 | 0 | 0 | 1 | 3 |
| $diff_i$ | | 5 | 3 | 0 | 0 | 5 | 5 | 6 | 0 |
| $compare_i$ | | | | | 3 | | | | 2 |

Table 2 shows delay-variation associated with each node. The delay-variation parameter is calculated as 2. The candidate core nodes are $v_4$ and $v_8$. The parameter $compare_i$ is calculated for each candidate node and it is seen that the best core node is $v_8$. The multicast tree is generated by connecting the destination nodes $v_5$ and $v_6$ with core node $v_8$ through the least delay path and the best of the $k$-least delay paths from source node to core node. The multicast trees generated by DDVCA and AKC algorithms are shown in Fig.7 and Fig.8 respectively. The final multicast tree generated by our algorithm is shown in Fig.9.

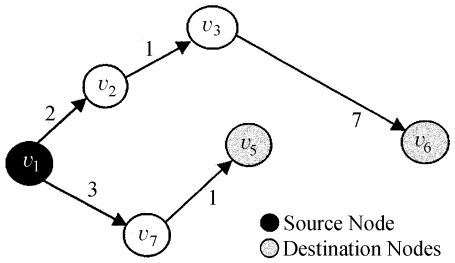


Fig.7. Tree generated by DDVCA, $dv = 7$.

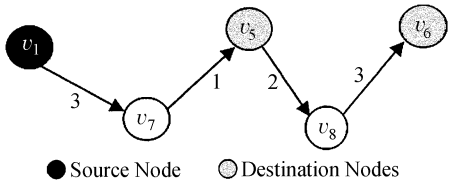


Fig.8. Tree generated by AKC algorithm, $dv = 5$.

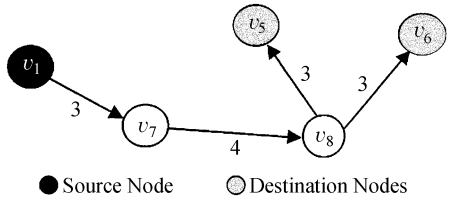


Fig.9. Tree generated by our proposed algorithm, $dv = 0$.

### 3.3 Complexity Analysis

Line 03 is computed in $O(n^2)$ using Dijkstra's algorithm. The lines 05∼06 can also be computed in $O(mn^2)$ by Dijkstra's algorithm. In line 04, $O(n^2)$ is required to transpose adjacency matrix $G$ to its $G'$. Lines 08∼12 and 13∼14 are computed in $O(mn)$ and $O(n)$ respectively. The worst case complexity of lines 22∼25 is $O(n)$. The $k$-shortest paths from the source to the best core node are computed by breaking each link constituting the original shortest path from source to the best core node[16]. If there are $h$ links in the original shortest path, then it computes $k$ shortest paths by using Dijkstra's algorithm in $O(hn^2)$. In the worst case, the path consists of $n$ nodes with $n-1$ links. So the worst case complexity of lines 26∼27 is $O(n^3)$. Line 29 is computed in $O(mn)$. Line 30∼46 require $O(kn)$ to find the best $k$-shortest paths. Line 48 is computed in $O(n)$ and line 49 is computed in $O(n^2)$ using Dijkstra's algorithm. Consequently, the total time complexity of our proposed algorithm is $O(mn^2)$, if $m \geqslant h$, which is comparable to the existing algorithms[6,13-15]. However, the worst case complexity of our proposed algorithm is $O(n^3)$.

## 4 Simulation Results

We have implemented our proposed algorithm in Visual C$^{++}$. The experiments are performed on an Intel Core i3 @2.27 GHz and 2 GB RAM-based platform.

The positions of the nodes are fixed randomly in a rectangle of size 4000 km × 2400 km. The Euclidean metric is then used to determine the distance between each pair of nodes. Edges are introduced between the pairs of nodes $u$, $v$ with a probability that depends on the distance between them. The edge probability is given by $P(u,v) = \beta \exp(-l(u,v)/\alpha L)$, where $l(u,v)$ is the distance from node $u$ to $v$ and $L$ is the maximum distance between any two points in the graph. The value of $\alpha$ controls the number of short links in the randomly generated network topology. The smaller the value of $\alpha$, the higher number of shorter links, $\beta$ controls the number of links in the randomly generated network topology. The lower the value of $\beta$, the larger the number of links, where $\alpha$ and $\beta$ are set to 0.8 and 0.7 respectively.

The link delay function $D(e)$ is defined as the propagation delay of the link, which is calculated by the equation $\frac{l(u,v)}{L} \times SCALE$. The $SCALE$ is assumed to be 20 ms. The source node is selected randomly and destination nodes are picked up uniformly from the set of nodes chosen in the network topology. The delay bound $\Delta$ is set to be 1.5 times the minimum delay between the source and the farthest destination nodes. We also implement DDVCA[13], KIM[6] and AKC[15] algorithms in the same environment to study and compare the performance of our proposed algorithm with the existing algorithms. The simulation is run for 200 times for each case and the average is taken as the output.

### 4.1 Comparison of Multicast Delay-Variations

Fig.10 shows the simulation results of multicast delay-variations versus the number of nodes on a network. The destination nodes in a multicast group occupy 10% of the overall network nodes. The multicast delay-variation of our proposed algorithm is found to be better than that of the existing algorithms[6,13-15].

Fig.11 and Fig.12 show the multicast delay-variation for a network of 100 nodes and 200 nodes respectively.

The multicast group size is between 10% and 70% of the total nodes of the network. We observe that the multicast trees obtained by our proposed algorithm has an average multicast delay variation better than DDVCA[13], KIM[6] and AKC[15] algorithms. This is because our proposed algorithm considers the candidate nodes that have delay-variations close to the minimum delay variation. In addition, our algorithm selects the best path out of $k$-shortest paths from source to the best core node.

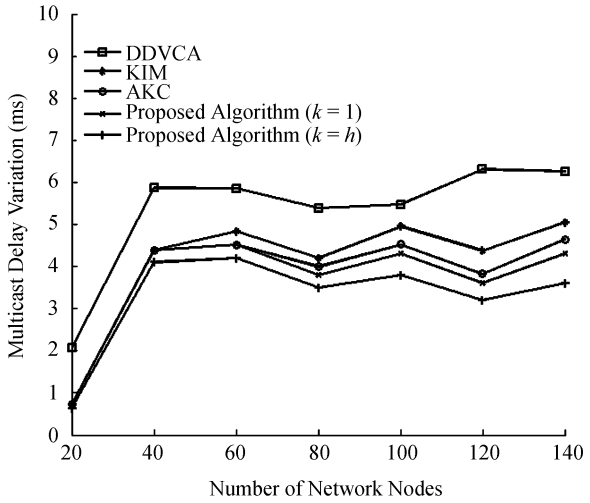


Fig.10. Comparison of multicast delay-variations for varying network size.

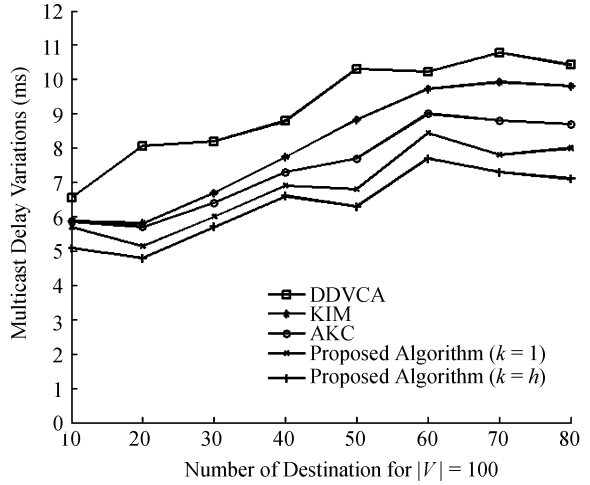


Fig.11. Comparison of multicast delay-variations for varying group size.

### 4.2 Comparison of Execution Time

Table 3 shows the simulation results of actual execution time versus the number of nodes with 10% of the nodes in the multicast group. It can be easily observed that the execution time of our proposed algorithm without considering $k$-shortest paths is comparable to other existing algorithms[6,13,15]. However, the proposed algorithm with $k$-shortest paths has more execution time than the other existing algorithms. This is because the execution time of our proposed algorithm with $k$-shortest paths depends on the value of $k$. We have computed $k$-shortest paths with the algorithm proposed in [16] by breaking each link of the original shortest path calculated by Dijkstra's algorithm. This is found to be quite efficient while finding $k$-shortest paths between a single pair of nodes. It can be observed that the rate of increase of $k$ goes down as the number of nodes increases. Therefore, the rate of increase in execution time also goes down with the increase in the number of nodes.

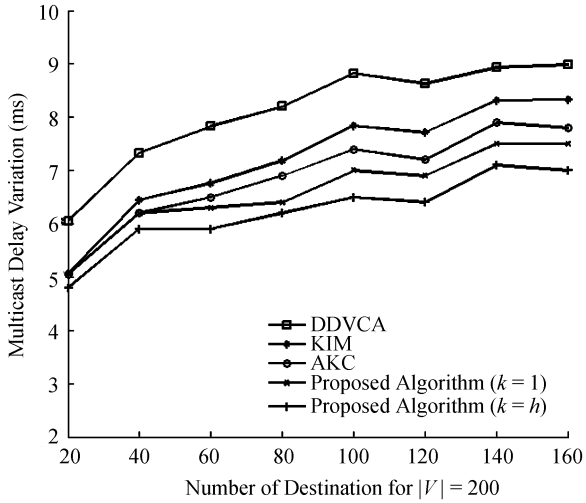


Fig.12. Comparison on multicast delay-variations for varying group size.

**Table 3.** Execution Time vs Number of Nodes with 10% of the Nodes in the Multicast Group

| No. Nodes | DDVCA (ms) | KIM (ms) | AKC (ms) | Proposed Algorithm ($k = 1$) (ms) | Proposed Algorithm ($k = h$) (ms) |
|---|---|---|---|---|---|
| 20 | 02.65 | 02.71 | 02.78 | 02.83 | 03.91 |
| 40 | 04.12 | 04.20 | 04.30 | 04.30 | 04.90 |
| 60 | 05.02 | 05.19 | 05.23 | 05.26 | 05.64 |
| 80 | 05.63 | 05.86 | 05.92 | 05.92 | 06.12 |
| 100 | 08.17 | 08.34 | 08.42 | 08.42 | 10.08 |
| 120 | 11.64 | 11.80 | 12.00 | 12.32 | 14.18 |
| 140 | 18.23 | 18.69 | 18.78 | 18.82 | 21.55 |

A comparison of execution time with the number of nodes in the multicast group is shown in Table 4 for a network of 100 nodes. The execution time of the proposed algorithm with $k$-shortest paths is observed to be more than that of the existing algorithms where as the execution time of the proposed algorithm

without $k$-shortest paths is comparable to the existing algorithms. We also observe that the rate of increase in execution time goes down with the increase in the number of destinations.

**Table 4.** Execution Time vs Percentage of Nodes in the Multicast Group (total nodes = 100)

| Nodes in Multicast Group (%) | DDVCA (ms) | KIM (ms) | AKC (ms) | Proposed Algorithm ($k = 1$) (ms) | Proposed Algorithm ($k = h$) (ms) |
|---|---|---|---|---|---|
| 10 | 08.17 | 08.34 | 08.42 | 08.42 | 10.08 |
| 20 | 10.25 | 10.59 | 10.86 | 10.82 | 11.89 |
| 30 | 10.90 | 11.02 | 11.12 | 11.50 | 15.46 |
| 40 | 15.74 | 15.94 | 16.23 | 16.43 | 17.96 |
| 50 | 18.29 | 18.57 | 18.92 | 19.25 | 21.00 |
| 60 | 21.08 | 21.35 | 21.83 | 21.76 | 23.80 |
| 70 | 21.46 | 21.87 | 22.16 | 22.23 | 24.29 |

## 5 Conclusions

In this paper, we proposed a novel heuristic algorithm for core selection in QoS multicast routing. We introduced a delay-variation parameter to regulate the selection of candidate core nodes. The best core node is selected out of these core nodes using some heuristic measures. Then, the best core node is connected to the destination nodes with the shortest paths. The mathematical time complexity of our algorithm is $O(mn^2)$ if $m \geqslant h$. To verify the effectiveness of our algorithm we compared it with a number of other commonly used core selection algorithms. The simulation results reveal that the multicast tree generated by our algorithm has less delay-variation than that found by other algorithms. The simulation results also show that the actual execution time of our algorithm is more than the existing algorithms, which is due to the computation of $k$-shortest paths from source to the best core nodes. However, the rate of increase in execution time goes down with the increase in the number of nodes. Our algorithm also gives better performance without computing $k$-shortest paths in terms of average multicast delay-variation with an execution time comparable to the existing algorithms. Though actual execution time increases with the computation of $k$-shortest paths, our algorithm performs significantly better than all the existing algorithms in terms of delay-variation.

**Acknowledgements** We would like to thank anonymous reviewers and editors for their valuable suggestions to improve the paper.

## References

[1] Cain B, Ballardie A, Zhang Z. Core based trees (CBT version 3) multicast routing, protocol specification. Inter-Domain Multicast Routing, 2003, Internet Draft, http://tools.ietf.org/id/draft-ietf-idmr-cbt-spec-v3-01.txt.

[2] Fenner B. Protocol independent multicast-sparse mode (PIM-SM): Protocol specification (revised). Internet RFC 4601, August 2008.

[3] Wall D W. Mechanisms for broadcast and selective broadcast [Ph.D. Dissertation], Stanford University, 1980.

[4] Oliveira C A S, Pardalos P M. A survey of combinatorial optimization problems in multicast routing. *Computers and Operations Research*, 2005, 32(8): 1953-1981.

[5] Chung S M, Youn C H. Core selection algorithm for multicast routing under multiple QoS constraints. *Electronics Letters*, 2000, 36(4): 378-379.

[6] Kim M, Bang Y C *et al.* On efficient core selection for reducing multicast delay variation under delay constraints. *IEICE Trans. Communications*, 2006, E89-B(9): 2385-2393.

[7] Harutyunyan H A, Dong X. A new algorithm for RP selection in PIM-SM multicast routing. In *Proc. Int. Conf. Wireless and Optical Communications*, Banff, Canada, Jul. 14-16, 2003, pp.208-216.

[8] Lee D L, Youn C H, Jeong S J. RP reselection scheme for real-time applications in delay-constrained multicast networks. In *Proc. IEEE International Conference on Communications*, New York, USA, Apr. 28-May 2, 2002, pp.1290-1294.

[9] Mukherjee R, Atwood J W. Rendezvous point relocation in protocol independent multicast-sparse mode. *Telecommunication Systems*, 2003, 24(2-4): 207-220.

[10] Putthividhya W, Tavanapong W*et al.* selection with QoS support. In *Proc. IEEE Int. Conf. Communications*, Paris, France, Jun. 20-24, 2004, pp.2132-2137.

[11] Font F, Mlynek D. Applying clustering algorithms as core selection methods for multiple core trees. In *Proc. IEEE Int. Conf. Communications*, Paris, France, Jun. 20-24, 2004, pp.2030-2035.

[12] Rouskas G N, Baldine I. Multicast routing with end-to-end delay and delay variation constraints. *IEEE Journal on Selected Area in Communications*, 1997, 15(3): 346-356.

[13] Sheu P R, Chen S T. A fast and efficient heuristic algorithm for the delay- and delay variation-bounded multicast tree problem. *Computer Communications*, 2002, 25(8): 825-833.

[14] Ahn Y, Kim M, Bang Y C, Choo H. On algorithm for the delay-and delay variation-bounded multicast trees based on estimation. In *Proc. HPCC 2005*, Sorrento, Italy, Sept. 21-23, 2005, pp.277-282.

[15] Ahn S, Kim M, Choo H. Efficient algorithm for reducing delay variation on delay-bounded multicast trees in heterogeneous networks. In *Proc. WCNC*, Las Vegas, USA, Mar. 31-Apr. 3, 2008, pp.2741-2746.

[16] Zhao J, Hassanein H, Wu J, Gu G. End-to-end QoS routing framework for differentiated services networks. *Computer Communications*, 2003, 26(6): 566-578.

**Manas Ranjan Kabat** received his M.E. degree in information technology and computer engineering from Bengal Engineering College, India, and the Ph.D. degree in computer science and engineering from Sambalpur University, India. He is currently working as senior lecturer in the Department of Computer Science and Engineering at Veer Surendra Sai University of Technology, Odisha, India. His research involves multicast routing, reliable multicast, high speed computer networks and e-governance. He has published about 12 research papers in various international journals and conferences.

**Manoj Kumar Patel** received his M.Sc. degree in mathematics and MCA degree from Sambalpur University and IGNOU, India. He is currently pursuing the Ph.D. degree at Sambalpur University, India. He is also working as a technical asst. Gr-I (computer) in the Department of Computer Science & Engineering at Veer Surendra Sai University of Technology, India. His research area includes QoS routing, reliable multicast and soft computing techniques. He has published about 7 research papers in various international journals and conferences.

**Chita Ranjan Tripathy** is a professor at Department of Computer Science and Engineering, Veer Surendra Sai University of Technology, Odisha, India. He is currently the dean of academic affairs, Veer Surendra Sai University of Technology. He received his Master's and Ph.D. degrees in computer science and engineering from Indian Institute of Technology, Kharagpur, India. He has published more than 80 research papers in different international journals and conferences. His research area includes computer networks, parallel processing and reliability. He has received his "Sir Thomas Ward Memorial" gold medal for researches in parallel processing. He is a fellow of Institution of Engineers (India) and lifetime member of Indian Society for Technical Education (ISTE), Instrument Society of India and Orissa Information Technology Society (OITS).