ELSEVIER

Computer Communications 21 (1998) 1693–1706

computer communications

Algorithms for delay-constrained low-cost multicast tree construction

R. Sriram, G. Manimaran, C. Siva Ram Murthy*

Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600 036, INDIA

Received 6 March 1998; received in revised form 30 June 1998; accepted 1 July 1998

Abstract

With the proliferation of multimedia group applications, the construction of multicast trees satisfying quality of service (QoS) requirements is becoming a problem of prime importance. Multicast groups are usually classified as sparse or pervasive groups depending on the physical distribution of group members. They are also classified based on the temporal characteristics of group membership into static and dynamic groups. In this paper, we propose two algorithms for constructing multicast trees for multimedia group communication in which the members are sparse and static. The proposed algorithms use a constrained distributed unicast routing algorithm for generating low-cost, bandwidth and delay constrained multicast trees. These algorithms have lower message complexity and call setup time due to their nature of iteratively adding paths, rather than edges, to partially constructed trees. We study the performance (in terms of call acceptance rate, call setup time and multicast tree cost) of these algorithms through simulation by comparing them with that of a recently proposed algorithm (V. Kompella, J.C. Pasquale, G.C. Polyzos, Two distributed algorithms for the constrained Steiner tree problem, in: Proc. Comp. Comm. Networking, San Diego, CA, June 1993) for the same problem. The simulation results indicate that the proposed algorithms provide larger call acceptance rates, lower setup times and comparable tree costs.

Keywords: Multicast routing; Steiner trees; Constrained-optimization; QoS; Interactive multimedia; Group communication

1. Introduction

A majority of the current and future distributed real-time applications such as teleconferencing, remote collaboration and distance education require the underlying communication network to provide multicast services. Also, these applications involve the transmission of multimedia information and therefore it is essential to satisfy QoS constraints (such as bounded end-to-end delay, bounded delay-variation and bandwidth requirement). At the routing level, these two requirements are translated into the problem of determining a multicast tree, usually rooted at the source node and spanning the set of receiver (destination) nodes. The QoS constraints typically impose a restriction on the acceptable multicast trees.

In multicast communication, messages are concurrently sent to all the members of a given multicast group. These groups are usually classified based on the physical distribution of group members [6] and the temporal characteristics of group membership. Sparse groups are those, which have very few members when compared with the size of the network. Real-time applications such as the ones mentioned previously typically fall into this category. They can be contrasted with pervasive groups which have a large number of members attached to most of the links in the network. Applications such as widespread directory services and network distribution groups fall into this category. Another classification that is of importance to routing is the division of groups into static groups (where group membership is constant) and dynamic groups (where changes in group membership are allowed, i.e. members are allowed to join/leave the group).

In this paper, two algorithms for constructing delay-constrained multicast trees for static sparse groups, are described. The rest of the paper is organized as follows. In Section 2, we describe some of the existing approaches to multicast tree construction and also discuss the motivation for our work. The formal problem specification and the proposed routing algorithms are presented in Section 3. In Section 4, we provide proofs of correctness of the proposed algorithms. Section 5 discusses the simulation results and compares the performance of our algorithms with that of a previously published algorithm. Section 6 concludes the paper highlighting the advantages of our new algorithms.

* Corresponding author. Fax: 0091 44 2350509; e-mail: murthy@iitm.ernet.in

0140-3664/98/$ - see front matter
PII: S0140-3664(98)00208-4

2. Background and motivation

Multicast route determination is traditionally formulated as a problem relating to tree construction. The tree structure allows parallel transmission to the various destination nodes and also minimizes data replication. In the following section, we will describe some of the general approaches to multicast tree construction.

2.1. General approaches to multicast routing

From the existing literature, three fundamental techniques for multicast routing have been identified in [7]. They are:

1. Source-based routing: this approach is essentially based on the reverse path forwarding (RPF) algorithm (proposed by Dalal and Metcalfe) and involves the computation of an implicit spanning tree per source node that is usually the shortest delay tree. The RPF algorithm, in conjunction with pruning techniques, has been widespread throughout it's employment in IP-multicast (Distance-Vector Multicast Routing Protocol (DVMRP) [6], Protocol Independent Multicast (PIM) [5]). It's main advantage is it's simplicity. However it is most suitable for broadcasting and performs poorly when applied to sparse multicast groups (which is the focus of our paper).
2. Center-based trees: this approach is ideal for multiple-sender/multiple-recipient communication. The unifying feature of this class of algorithms is that they identify a center node for each multicast group and construct a distribution tree rooted at this center. The core based tree (CBT) algorithm [1] comes under this category.
3. Steiner-tree approach: this approach models the multicast tree construction problem in terms of the graph-theoretic Steiner problem. The key focus of algorithms that come under this category is their emphasis on overall tree cost minimization. The algorithms proposed in this paper come under this category.

Non-Steiner approaches (such as those proposed or employed on the internet, including DVMRP, PIM and core-based trees) are suitable for datagram environments such as the internet in which the routes taken by multicast packets may vary [3]. For such environments, there is no point in emphasizing cost minimization and more importance is given to minimization of algorithm overhead [3,7]. However, Steiner tree heuristic approaches apply to virtual-circuit environments such as ATM networks [3]. In such networks, the route (virtual circuit) selected for a certain connection is used to forward all packets of that connection. Hence, it makes sense to model the cost of a connection in terms of the cost of the corresponding tree and to concentrate on cost-minimization so as to improve overall network utilization [3].

2.2. The Steiner tree approach

It has been shown that the multicast tree problem can be modelled as the Steiner problem in networks [11,18], which is NP-complete [9]. Consequently a number of centralized algorithms that construct low-cost multicast routes such as those in Refs. [4,12,23] are based on approximation algorithms for Steiner tree construction [15,22]. Some of these algorithms produce solutions that are provably within twice the cost of the optimal solution and run in polynomial time, usually ranging between $O(n^3)$ and $O(n^4)$. However, to be implemented as a working protocol, distributed heuristics are required. Introduction of the QoS requirements into the Steiner problem results in the constrained Steiner tree problem, for which, once again, distributed heuristics are required.

A number of centralized and distributed algorithms, some dealing with pure tree cost minimization and others dealing with both cost minimization and QoS constraints, have been proposed. In [2] distributed SPH (shortest path heuristic) and K-SPH (Kruskal type SPH) heuristics have been described that deal only with the construction of unconstrained Steiner trees. In [12], a problem that involves optimization on both cost and delay metrics is studied. However, the algorithm assumes that the two metrics are related functions and exploits this dependency to achieve a compromise between minimizing average source-destination delay and reducing tree cost. The idea of destination biasing is employed in [21], along with the greedy strategies of shortest path trees and minimal spanning trees to construct low-cost unconstrained multicast trees. In Ref. [20], the cost metric has been dropped from consideration and instead, an attempt has been made, to construct multicast trees that satisfy an upper bound on the delay along the source-destination paths as well as an upper bound on the delay-variation between these paths. This bound on delay variation allows for synchronization between the various receiver nodes. To construct trees for video distribution, the Steiner problem has been modified in [16] to include non-constant link costs. Here, link cost is assumed to be an increasing function of bandwidth and length of the link and is not dependent on the utilization or availability of the link.

2.3. Motivation for our work

It is obvious from the above discussion that a great variety exists in both the nature of multicast routing problems and in the types of solutions that have been proposed for them. In this paper, we consider the problem of constructing low-cost multicast trees that satisfy a specified bound on the delay between source and any receiver. Delay constraint is a very common and fundamental requirement of many multimedia applications. Cost minimization captures the need to distribute the network resources efficiently amongst the various multicast channel establishment requests. Therefore this problem specification captures the requirements of both

the application (delay constraint) as well as the network (effective utilization). The above problem has been formulated in [13] as the constrained Steiner tree problem.

In Ref. [13], two centralized heuristic algorithms (adapted from the KMB algorithm [15]) for this multicast tree construction problem are described. Both the algorithms are source-based routing algorithms which assume that the source has all the information (network topology, link delays and link costs) necessary to construct the tree. The algorithms have a common initial stage that involves the computation of a closure graph on the source node and the set of receiver nodes. However, the two algorithms use different heuristic functions to construct a spanning tree of the closure graph which then yields the required multicast tree. Distributed versions of these two algorithms are presented in [14].

The centralized algorithms proposed in [13] are unsuitable for larger networks as the process of maintaining consistent network information at every node becomes prohibitively expensive. The distributed versions of these algorithms [14] avoid this problem by using only local information at each node. However, because the algorithms construct the tree by adding only one edge at a time, the time required to actually setup the tree could be very large. This is especially true for sparse, widely distributed multicast groups where the multicast tree could include a large number of links. This large setup time also means that if we attempt to include resource reservation along with the tree construction process, then resources (such as link bandwidth) might be reserved for a long time before they are actually needed. This results in poor network utilization, decreased overall throughput and lower call acceptance rates. Another disadvantage of the algorithms is that they do not provide for any tunable parameters, which could be used to achieve a compromise between the optimality of the tree and the time required to setup the tree.

In this paper, we describe two algorithms for delay-constrained low-cost multicast routing that overcome the disadvantages described above. The algorithms are general, in the sense that, they can be used in conjunction with any constrained cost-minimizing unicast routing algorithm [10,17].

3. The proposed algorithms

In this section, we will first describe the network model and formally state the problem using this model. We will then present a detailed description of our proposed algorithms.

3.1. Network model

In this paper, the network is modelled as an undirected graph $G = (V,E)$, where V is the set of nodes and E is the set of links. An edge $e \in E$ connecting nodes u and v will be denoted by (u,v). We associate the following four functions with each link $e \in E$.

Delay function	D:	$E \rightarrow R^+$
Cost function	C:	$E \rightarrow R^+$
Total bandwidth function	TB:	$E \rightarrow R^+$
Available bandwidth function	AB:	$E \rightarrow R^+$

A path $P = (v_0, v_1, v_2, \cdots, v_n)$ in the network, has two associated characteristics

$$Cost\ C(P) = \sum_{i=0}^{n-1} C(v_i, v_{i+i})$$

$$Delay\ D(P) = \sum_{i=0}^{n-1} D(v_i, v_{i+1})$$

Similarly, a tree $T = (V_T, E_T)$ which is a subgraph of G has an associated cost defined as

$$C(T) = \sum_{e \in E_T} C(e)$$

Given a path P and two nodes v_1 and v_2 belonging to this path, the portion of P connecting these two nodes will be denoted by Sub $P(v_1, v_2)$. $D_P(v_1, v_2)$ and $C_P(v_1, v_2)$ will denote, respectively, the delay and cost of this portion of P. Similarly, given a tree T and two nodes v_1 and v_2 belonging to this tree, we will let $P_T(v_1,v_2)$ denote the path between v_1 and v_2 in this tree. Then, the delay and cost of this path are respectively denoted as $D_T(v_1, v_2)$ and $C_T(v_1,v_2)$. We assume that for each $e \in E$, $C(e)$, $D(e)$ and $TB(e)$ are fixed, though $AB(e)$ varies depending on the usage of the link.

3.2. Problem formulation

We model a multicast tree-establishment request (also referred to as a multicast call request) in the network described previously, as a 4-tuple:

$C = (s, R, B, \Delta)$, where

$s \in V$ is the source node for the call; $R = \{d_1, d_2, \cdots, d_m\} \subset V$ is the set of receiver nodes for the call; B is the bandwidth requirement for the call; and Δ is the delay constraint to be satisfied for each source receiver pair.

Given such a call C on the network $G = (V, E)$, we define a bandwidth and delay constrained spanning tree to be a tree $T = (V_T, E_T)$ rooted at s and satisfying the following conditions:

- $V_T \subseteq V$ & $E_T \subseteq E$
- $s \in V_T$ & $R \subseteq V_T$
- $D_T(s,v) < \Delta\ \forall\ v \in R$
- $AB(e) > B\ \forall\ e \in E_T$

Let $S(C)$ denote the set of all such spanning trees corresponding to call C. The problem can now be formulated as:

find a spanning tree T_S such that $C(T_S) = \min[C(T_S):T \in S(C)]$

Such a tree is a bandwidth and delay constrained Steiner tree (BDCST).

3.3. Routing strategy

In this section, we will first informally describe our two algorithms and then develop notations that will facilitate their formal description.

The proposed algorithms assume the existence of a unicast routing strategy that constructs low-cost delay-bounded loop-free paths between a given pair of nodes. A preferred link approach for developing such algorithms along with a set of three heuristics is presented in [17]. Another such unicast routing algorithm that uses distance vector tables at the various nodes is described in [10]. In both the algorithms presented in this paper, the unicast routing strategy is responsible for ensuring that only those links that have an available bandwidth that is greater than the bandwidth requirement of the call are selected. The algorithms presented in [17] satisfy this requirement.

3.3.1. Algorithm A1

In this algorithm, two distinct phases can be identified. The first phase will be distributed, provided, the underlying unicast routing strategy is distributed. The second phase of the algorithm is a centralized computation to be performed at the source node.

Phase 1 : the first phase involves the construction of delay-constrained least cost paths between source and every destination, using the unicast routing strategy. At the end of this phase of the algorithm, the source node will have a set of $m(=|R|)$ paths, $(P_1, P_2, \cdots, P_m)$ with the following properties:

$\forall 1 \le i \le m$	path P_i connects source node s to the ith destination node d_i
$\forall 1 \le i \le m$	delay $D(P_i) < \Delta$
$\forall e \in P_i, 1 \le i \le m$	$AB(e) > B$

Phase 2 : this phase constructs a multicast routing tree rooted at the source, using the m paths produced at the end of the first phase. Starting with an empty tree, the computation proceeds by adding one new source-destination path at a time (at each stage), to an existing tree. While augmenting an existing tree by adding another path, loops have to be removed in such a way that the resulting structure produced is connected and none of the delay constraints are violated. This is accomplished as follows.

The computation proceeds by constructing a sequence of m trees $T_1, T_2, \cdots, T_m$ where tree T_i is constructed using paths $P_1, P_2, \cdots, P_i$.Let us consider the stage in the computation when the path P_j, connecting s and d_j, is to be added to the tree T_{j-1}. Let $I_j = (x_{j1}, x_{j2}, \cdots, x_{jk})$ be the sequence of nodes at which P_j intersects T_{j-1}, ordered in increasing order of their distance (along P_j) from d_j. By definition, $x_{j_k} = s$. This sequence of nodes is called the intersection set (I-set) of P_j. This sequence of nodes is scanned in the above specified order to augment T_{j-1} and produce tree T_j. If $k = 1$, the scan terminates immediately and T_j is obtained from T_{j-1} by merely attaching the path P_j at s. For $k > 1$, let us consider the stage in the scan when intersection node x_{j_t} is under consideration. The following condition (C) is tested at this intersection point:

$$D_T(s, x_{j_t}) + D_{P_j}(x_{j_t}, d_j) < \Delta$$

The following two cases arise depending on the outcome of the test.

If condition (C) is satisfied, then, the portion of P_j between nodes x_{j_t} and d_j is attached to T_{j-1} using x_{j_t} as the point of attachment. This gives the required tree T_j. The condition guarantees that the delay between s and d_j in this new tree will satisfy the delay bound.

If condition (C) is not true, then the portion of P_j between x_{j_t} and $x_{j_{t+1}}$ will also need to be included in the tree T_j. Since both x_{j_t} and $x_{j_{t+1}}$ are already nodes of T_{j-1}, there is a possibility of a loop being created. To break this loop, the path between x_{j_t} and s in T_{j-1} is scanned, starting from x_{j_t}, until a node x which falls into one or more of the following categories is encountered:

- case 1: $x = s$;
- case 2: degree of x in T_{j-1} is greater than 2;
- case 3: x is a member of the destination set R;
- case 4: x is one of the vertices in the set in $\{x_{j_1}, x_{j_2}, \cdots, x_{j_{t-1}}\}$

Once such node x has been determined, the portion of T_{j-1} connecting x_{j_k} and x is deleted and condition (C) is now tested for the next intersection point.

The scan continues until the condition (C) becomes satisfied and tree T_j is generated. In the worst case, this will definitely happen when the scan reaches vertex $x_{j_k} = s$. This tree augmentation step is executed repeatedly until all the m paths have been added to the tree. Fig. 1 illustrates the steps involved in a particular stage in Phase 2 of the algorithm when node d_i is being attached to T_{i-1}. Fig. 1(a) and (b) depict the case where condition (C) is not true at nodes x and y, respectively. Fig. 1(a) is an example of case 1 before, whereas Fig. 1(b) is an example of case 2. In Fig. 1(c), condition (C) is satisfied at node y and the scan stops at this point (the portion *sabz* of the path is neglected).

3.3.2. Algorithm A2

This algorithm is a purely distributed algorithm, provided the unicast routing strategy is distributed. In this algorithm, as in algorithm A1, the multicast tree is constructed by adding one destination node at a time to the tree. In A1, this addition was purely computational and all the decisions

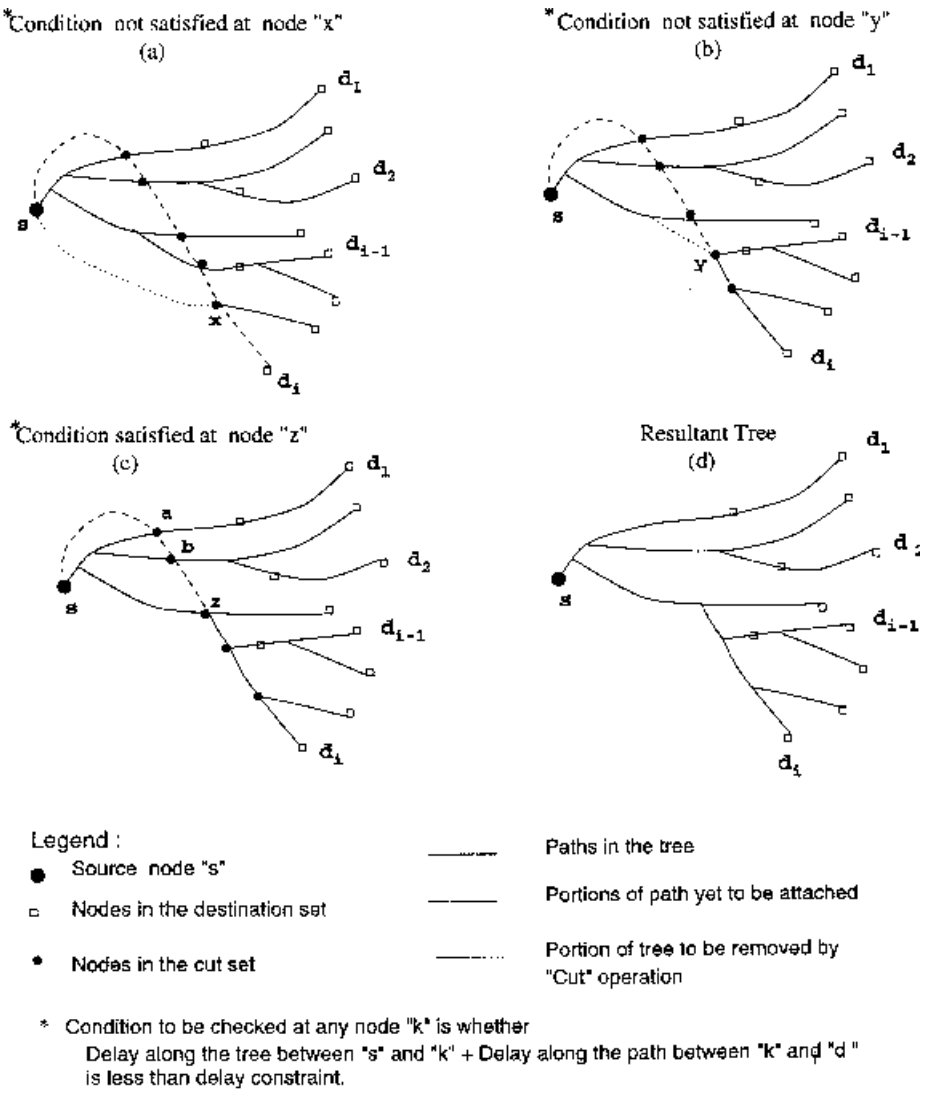

Fig. 1. Illustration of a stage in Phase 2 of Algorithm A1.

were made at the source node as part of Phase 2. However, in algorithm A2, the decision about how to attach to the existing tree is taken by the destination nodes, from a set of options provided by the nodes of the already existing tree. We will let (T_1, T_2,⋯, T_m) denote the sequence of partial trees constructed by adding one destination at a time.

At any given stage in the construction of the multicast tree, a priority list of nodes currently in the tree, is maintained. The nodes in the list are stored in decreasing order of their priorities. The priorities of the nodes are defined by the following function:

$$PF(x) = \Delta - D_T(s,x)$$

where $PF(x)$ denotes the priority of node x and T is the tree currently constructed. Nodes with larger PF values have a higher priority. This priority function for each node captures the residual delay at that node. This is the maximum delay of any path that can be attached to the tree at this node (without violating the delay constraint for the other end of the path). The intuitive reasoning behind this priority function is that nodes with larger residual delays have a better chance of constructing low-cost paths to other destination nodes. Hence, they must be given a higher priority.

To attach a destination d_j to the tree T_{j-1}, the following steps are performed.

1. The first κ nodes in the priority list (if there are less than r nodes in the tree, then all nodes are used) initiate delay-constrained low-cost path setup to node d_j, using the unicast routing strategy. Node x will attempt to construct a path to d_j with a delay constraint of $(\Delta - D_{T_{j-1}}(s,x))$, where $D_{T_{j-1}}(s,x)$ is the delay between s and x in the partial multicast tree T_{j-1}.
2. If the unicast path setup initiated by some node x reaches a node y that is already in tree T_{j-1}, then y kills the path setup initiated by x. Instead, y initiates low-cost path setup to d_j with a delay constraint of $(\Delta - D_{T_{j-1}}(s,y))$.
3. Once d_j receives setup packets from the nodes specified previously, it computes a selection function for each path

generated by the previous step. If the unicast routing strategy is unable to construct a path that satisfies the delay-constraint, then d_j will receive failure packets from such nodes and will not use these paths in the later stages. For a non-failure path P connecting node x and node d_j, the selection function is defined as:

$$SF(P) = \frac{C(P)}{\Delta - D_{T_{j-1}}(s,x) - D(P)}$$

The denominator of the selection function denotes the residual delay (=priority value) of d_j in the resultant tree if P was chosen as the attachment path. If preference is given to paths with lower SF values, the selection function can be used to simultaneously minimize the cost of the path, at the same time, maximizing the priority value of d_j.

4. d_j selects the path with the least SF value to attach itself to the tree. Tree T_{j-1} augmented with this selected path constitutes the next tree T_j in the sequence. d_j updates the priority list to possibly include nodes in this newly attached path. It then sends this updated priority list as an attatch message up the tree T_j towards s.
5. Nodes which receive this attach message, check to see if their node number occurs within the first κ values in the list. If so, then they initiate constrained path setup towards the next destination node d_{j+1} as in step 1 before. All nodes also propogate this attached message up the tree towards the source node s.
6. When s receives this attach message, it determines the set Y of nodes that have a priority that is within the first r values and which did not occur on the path through which the attached message was forwarded. It then sends a message, via the shortest delay paths, to all the nodes in Y. (The message will contain the address of the next destination node d_{j+1}).
7. On receiving this message, nodes in Y initiate delay-constrained path setup to d_{j+1} as in step 1.

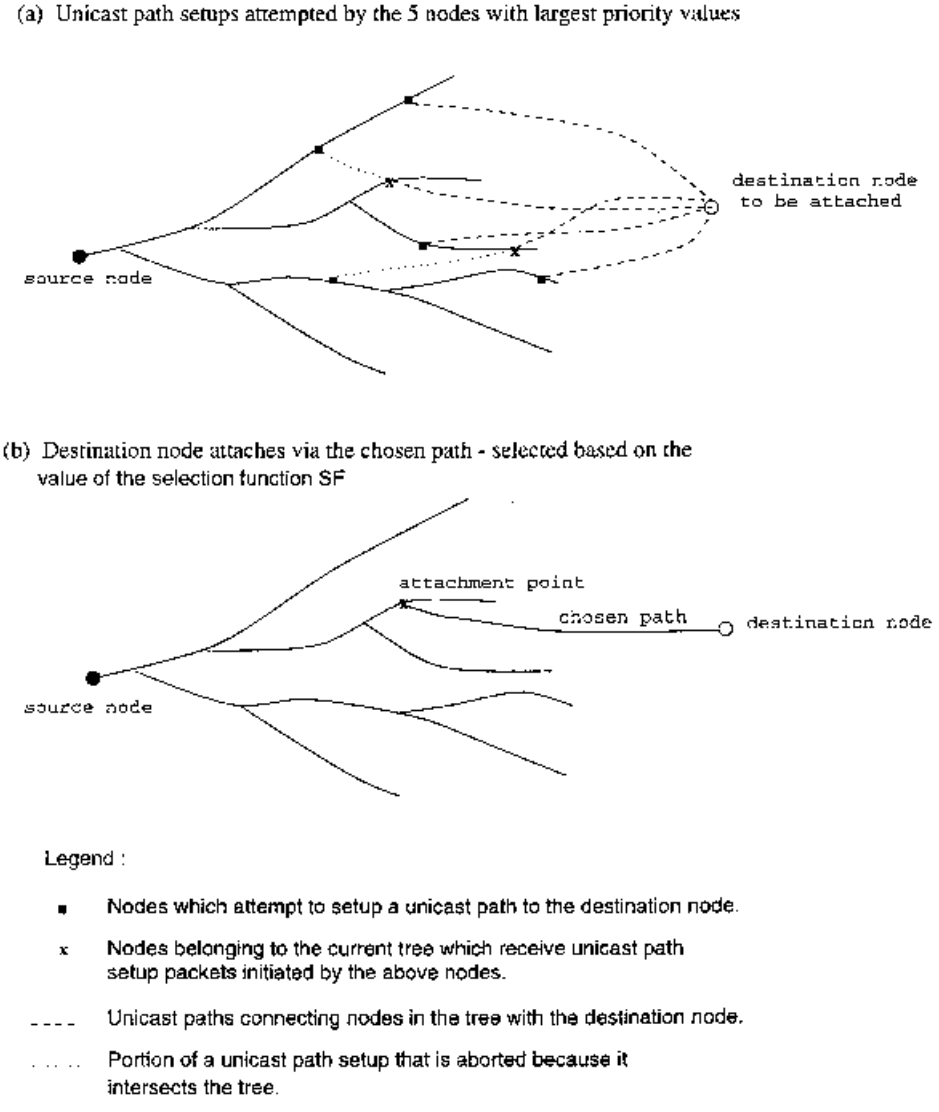

Fig. 2. Illustration of a stage in Algorithm A2.

The set of steps above are repeated until all the m destination nodes are attached. The final tree T_m is the required multicast tree.

Fig. 2 illustrates how a particular destination attaches itself to an existing tree in algorithm A2. In Fig. 2(a) five nodes parallely attempt to setup unicast paths towards the destination node. Two of these paths intersect the tree (at points indicated with crosses in the figure), in which case these intersection points initiate unicast path setups to the destination. Fig. 2(b) depicts the augmented tree, where the destination node has attached itself to the tree via the chosen path (path with lowest *SF* value).

3.3.2.1. Special Cases. Case 1: depending on the value of κ, it is possible that at some stage in the algorithm there may be less than κ nodes in the tree constructed upto that point. In this case, the next destination needs to be informed that it will receive less than κ packets (either through successful unicast call setups or failures as in step 3 above). This can be achieved by the source informing the destination about the number of attachment paths that it can expect to receive (this can be done as part of the unicast path that the source will attempt to establish to that destination). For ease of presentation, from now on we will assume that all partially constructed trees have at least κ nodes.

Case 2: in the case when the next destination to be attached is already part of the tree, the following can be done. As soon as that destination node receives a unicast call setup packet(s) (step 1) mentioning itself as the destination, it can generate an attached message, with an unchanged priority list (step 4) and containing the node number of the next destination to be attached and propogate it up the tree towards the source. It can then ignore any other setup packets, mentioning itself as the destination, that it receives.

3.3.3. Formal description of algorithm A1

In this section, we will formally present algorithm A1 using the notations given below.

1. UR(s, d, Δ, B): denotes the low-cost path between s and d determined by the unicast routing strategy for a delay-constraint Δ and bandwidth requirement B.
2. Attach(P, T, x): returns a tree obtained by attaching path P to tree T at the attachment point x.
3. $\deg_T(x)$:denotes the degree of a node x in tree T.
4. GetISet(P, T): let T be a tree rooted at s and let P be a path joining some node x and node s. This function returns the I-set of P with respect to T, i.e. it returns a sequence of nodes $(x_1, x_2, \cdots, x_k = s)$ at which P intersects T ordered such that x_i is closer (along P) to x than x_{i+1} $\forall$ $0 < i < k$.
5. $\mathrm{Cut}_T(x_i)$: let T be a tree that is rooted at source s and let x_i be a node in the tree that belongs to the I-set $\{x_1, x_2, \cdots, x_n\}$. Then this function scans the path (in T) connecting x_i and s, starting at x_i. The scan continues until a node y, which satisfies one or more of the following conditions, is encountered:

- $y = s$
- $\deg_T(y) > 2$
- $y \in R$ where R is the set of destination nodes
- $y \in \{x_1, x_2, \cdots, x_{i-1}\}$

The function returns a tree, obtained from T, by removing $P_T(x, y)$.

Using the above notations, the steps to be executed by a source node s, attempting to construct a multicast tree, spanning the receiver set $R = \{d_1, d_2, \cdots, d_m\}$, satisfying the delay constraint A and the bandwidth constraint B are as follows:

A1 (s, $R = \{d_1, d_2, \cdots, d_m\}$, Δ, B)
begin

For $i = 1$ to m

$P_i = \mathrm{UR}(s, d_i, \Delta, B)$

$T_1 = P_1$
For $i = 2$ to m

Let $\{x_1, x_2, \cdots, x_k\} = \mathrm{GetISet}(P_i, T_{i-1})$
If ($k = 1$) then $T_i = \mathrm{Attach}(P_i, T_{i-1}, s)$
else

Set $j = 1$ and joined = false
while (joined = false) do

If $(D_{T_{i-1}}(s, x_j) + D_{P_i}(x_j, d_i) < \Delta)$ then

$T_i = \mathrm{Attach}\ (\mathrm{Sub}_{P_i}(x_j, d_i), T_{i-1}, x_j)$
joined = true

else

$T_{i-1} = \mathrm{Cut}_{T_{i-1}}(x_j)$
$j = j + 1$

Tree T_m is the required multicast distribution tree.
end

3.3.4. Formal description of algorithm A2

We will describe the algorithm A2 as a set of actions to be taken by the nodes upon receipt of various kinds of messages. The response by a node to the receipt of a message, is dependent on whether the node is a source node, one of the destination nodes or any other node. The description of the various message types and the procedures enumerating the actions to be taken by the nodes are presented in Appendix A.

4. Proofs of correctness

In this section, we will establish the correctness of the two

algorithms described in the previous sections. To establish the validity of A1 and A2, we assume that the unicast routing strategy that is employed as part of these algorithms is correct.

In both the unicast and multicast models, we say that an algorithm for constrained routing is correct, if the route chosen by the algorithm satisfies both the delay and bandwidth constraints. Cost minimization is just an objective of the algorithms and does not play a role in defining their correctness. Formally, the correctness of a multicast routing algorithm is defined as follows:

Definition of correctness: If T is the tree constructed by a multicast routing algorithm A in response to a call request $C = (s, R = \{d_1, d_2, \cdots, d_m\}, B, \Delta)$, then A is correct if T satisfies the following properties:

1. $AB(e) > B$	$\forall\, e \in T$
2. $D_T(s,d_i) < \Delta$	for $1 \le i \le m$

4.1. Proof of correctness of algorithm A1

In the following proofs, we will use the same notation that was used in the formal description of algorithm A1 in Section 3.3.3.

Lemma 1. If the underlying unicast routing strategy is correct, then all the links chosen by algorithm A1 to construct the multicast tree satisfy the bandwidth requirement.

Proof. Since the unicast routing strategy is assumed to be correct, all the links that constitute the paths chosen as part of Phase 1 will satisfy the bandwidth requirement. Since Phase 2 does not introduce any new links but merely removes some of the links that are part of loops, it is obvious that the lemma holds.□

Lemma 2. If the underlying unicast routing strategy is correct, the tree constructed by algorithm A1 in response to a given call request satisfies the delay-constraint for every source destination pair in the multicast group.

Proof. We will prove this statement by induction on the size of the destination set. For a destination set of size $m = 1$, the final multicast tree is just the path P_1 constructed by the unicast routing strategy in Phase 1. Therefore, by the assumption that the unicast routing strategy is correct, it follows that this multicast tree satisfies the delay constraint. Let us assume that the lemma is true for all destination sets of size less than or equal to k.□

Consider a call $C = (s, R = \{d_1, d_2, \cdots, d_k, d_{k+1}\}, B, \Delta)$ and let $P_1, P_2, \cdots, P_k, P_{k+1}$ be the set of $(k+1)$ paths chosen by the unicast routing strategy in Phase 1. Since Phase 2 operates by adding one destination at a time to the tree, the tree T constructed after k stages in Phase 2 will be a tree that spans $s, d_1, \cdots, d_k$ and which, by the induction hypothesis, satisfies the delay constraint for each pair (s, d_i), $i = 1$ to k. Let $I = \{x_1, x_2, \cdots, x_t\}$ be the I-set of P_{k+1} with respect to T. Let j be the smallest integer in $[1,t]$ such that

$$D_T(s, x_j) + D_{P_{k+1}}(x_j, d_{k+1}) < \Delta \tag{1}$$

Then, the final tree T_f will be constructed from T by applying the Cut_T operation at nodes $x_1, x_2, \cdots, x_{j-1}$ and then attaching the path $Sub_{P_{k+1}}(x_j, d_{k+1})$ at node x_j. By the above inequality, the delay constraint for d_{k+1} is satisfied in T_f. It remains to show that this constraint is also satisfied for the other destination nodes.

If a destination node is part of the path $Sub_{P_{k+1}}(x_j, d_{k+1})$, then the delay constraint is obviously satisfied by the condition Eq. (1) above. Consider some destination node y in the final tree T_f that is not part of this path. We identify the following cases.

- $P_{T_f}(s, y)$ does not include any of the intersection nodes $x_1, x_2, \cdots, x_{j-1}$. In this case, the path $P_{T_f}(s, y)$ is the same as the corresponding path in T. Hence, the delay-constraint is trivially satisfied.
- Let x_l, for some $1 \le l \le j-1$, be the intersection node that is closest to y in the path $P_{T_f}(s, y)$. Then:

$$D_T(s, y) = D_T(s, x_l) + D_T(x_l, y) \tag{2}$$

$$D_{T_f}(s, y) = D_{T_f}(s, x_l) + D_T(x_l, y) \tag{3}$$

Now, x_l being an intersection vertex at which the Cut_T operation was performed, it means that

$$D_T(s, x_l) + D_{P_{k+1}}(x_l, d_{k+1}) \ge \Delta \tag{4}$$

By the nature of the construction, inequality Eq. (1) also implies that

$$D_{T_f}(s, x_l) + D_{P_{k+1}}(x_l, d_{k+1}) < \Delta \tag{5}$$

Inequalities Eqs. (4) and (5) together imply that $D_{T_f}(s, x_l) < D_T(s, x_l)$. This, along with Eqs. (2) and (3) gives $D_{T_f}(s, y) < D_T(s, y)$. Since the delay constraint is satisfied in T, this means that it is also satisfied in T_f.

Thus, the Lemma holds for any destination set of size $k+1$. By induction, the Lemma is proved.

Theorem 1. If the underlying unicast routing strategy is correct, so is algorithm A1.

Proof. Follows directly from Lemmas 1 and 2.□

4.2. Proof of correctness of algorithm A2

The correctness of the unicast routing strategy also guarantees the correctness of the algorithm A2. This is proved in the following theorem.

Lemma 3. If the underlying unicast routing strategy is correct, all the links chosen by algorithm A2 to construct the multicast tree satisfy the bandwidth requirement.

Proof. Algorithm A2 initiates the construction of the multicast tree by making a call to the unicast routing strategy to setup a path between the source and the first destination node. Every other destination node is included in the tree via a path constructed by the unicast routing strategy which connects that destination node to some attachment point in the partially constructed tree. Therefore, any link that is part of the final tree would be part of a path constructed by the unicast routing strategy. Consequently, all links will satisfy the bandwidth constraint.□

Lemma 4. If the underlying unicast routing strategy is correct, the tree constructed by algorithm A2 in response to a given call request satisfies the delay-constraint for every source destination pair in the multicast group.

Proof. In the final multicast tree, the first destination node d_1 is connected to the source node s by a path that is constructed by the unicast routing strategy. Hence, the delay-constraint is satisfied for this node d_1. Consider a stage in the algorithm when a tree T spanning s, $d_1,\cdots$, d_k has been constructed. Destination node d_{k+1} will attach itself to this tree through a path P (chosen by applying the selection function to the various paths that it received) that connects it with some node x in the tree. This path would have been determined by the unicast routing strategy in response to a call with a delay requirement of $\Delta - D_T(s,x)$. Therefore

$$D(P) < \Delta - D_T(s,x)$$

Hence, the delay between source and d_{k+1} in the final tree $= D_T(s,x) + D(P) < \Delta$. Thus, the delay constraint is satisfied for all destination nodes.□

Theorem 2. If the underlying unicast routing strategy is correct, so is algorithm A2.

Proof. Follows directly from Lemmas 3 and 4.□

5. Experimental results

In this section, we present the results of the simulation experiments conducted to analyse and compare the performance of the proposed algorithms with the distributed algorithms proposed by Kompella et al. [14] (discussed in Section 2.3). We will first define the performance metrics, then describe the simulation model and finally present and discuss the results.

5.1. Performance metrics

For an accepted tree establishment request C let us define the functions:

- accepted(C) = 1;
- setup(C) = time required to setup the multicast tree constructed for C;
- surcharge(C) = $(C_A - C_{MCPH})/C_{MCPH}$, where C_A is the cost of the tree constructed by some algorithm A and C_{MCPH} is the cost of the tree constructed using the minimum cost path heuristic (MCPH) defined in Ref. [22]. A low surcharge value indicates that the algorithm is close to achieving the pseudo-optimal tree cost. The MCPH heuristic was used instead of the optimal tree since construction of the optimal cost tree is extremely expensive for large networks. It should be noted however, that the trees built using MCPH do not take delay constraint into account and construct only unconstrained trees.

For a call request C that is rejected, all the functions return a value of 0. Let N be the total number of call-requests generated for simulation. The following metrics were used to analyse the performance of the routing algorithms:

Average call acceptance rate: the average probability of successfully constructing a constrained multicast tree.

$$\text{ACAR} = \frac{\sum_{i=1}^{N} \text{accepted}(C)}{N}$$

Average call setup time: the average time required to setup a multicast tree, measured in terms of number of messages sent.

$$\text{ACST} = \frac{\sum_{i=1}^{N} \text{setup}(C)}{\sum_{i=1}^{N} \text{accepted}(C)}$$

Average normalized surcharge: the average of the surcharge values for all the accepted call requests.

$$\text{ANS} = \frac{\sum_{i=1}^{N} \text{surcharge}(C)}{\sum_{i=1}^{N} \text{accepted}(C)}$$

The first metric is important as it is a measure of overall network throughput and utilization. The second metric is important in the context of real-time multimedia applications that require a call to be put through quickly. Metric 3 is useful in analysing the cost competitiveness of the proposed algorithms with respect to a pseudo-optimal algorithm such as MCPH. It also indicates the relative efficiency of the various algorithms with regard to cost minimization.

5.2. Simulation model

To conduct the simulation studies, we have used randomly generated networks on which the algorithms were executed. This ensures that the simulation results are independent of the characteristics of any particular network topology. Using randomly generated network topologies also provided the necessary flexibility to tune various network parameters such as average degree, number of nodes and number of edges and to study the effect of these parameters on the performance of the algorithms.

5.2.1. Random graph generation

In generating random graphs, we have adopted the method used in [23], where vertices are placed randomly in a rectangular coordinate grid by generating uniformly distributed values for their x and y coordinates. The graphs connectivity is ensured by first constructing a random spanning tree. This tree is generated by iteratively considering a random edge between nodes and accepting those edges that connect distinct components. The remaining edges of the graph are chosen by examining each possible edge (u,v) and generating a random number $0 \leq r < 1$. If r is less than a probability function $P(u,v)$ based on the edge distance between u and v, then the edge is included in the graph. The distance for each edge is the Euclidean distance [denoted as $d(u,v)$] between the nodes that form the endpoints of the edge. We used the probability function $P(u,v) = \beta e^{-d(u,v)/2\alpha n}$ where α and β are tunable parameters and n is the number of nodes in the graph. Increasing α increases the number of connections between far off nodes and increasing β increases the degree of each node.

5.2.2. Simulation parameters

The parameters α and β were tuned to produce networks with average node degree in the range 4 to 10. Random edge costs were generated uniformly in the range 1 to 10. Edge delays were made proportional to the Euclidean distance of the edges in the coordinate plane. Link capacity (total bandwidth) was randomly generated within the range [100,300] units. For algorithm A2, except in Fig. 1(a) and (b), the value of the parameter n was chosen to be approximately 12% of the size of the network on which the algorithm was run. Except in the case of Fig. 2(a), (b) and (c), where network size is the variable parameter, the average number of nodes in the network was maintained at 100.

The multicast requests were generated with the following parameters

- Source and destination nodes were chosen uniformly from the node set. The requests involved destination sets whose cardinality was chosen to be within 10%–30% of the total number of nodes in the network. This is so that sparse multicasts can be represented.
- Call duration, bandwidth requirement and delay constraint were uniformly distributed between their respective maximum and minimum values.
- The inter-arrival time of call establishment requests followed exponential distribution with mean $1/\lambda$.
- Bandwidth reservation was carried out as part of the unicast path setup. In the case of algorithm A1, after computation of the final tree at the end of Phase 2, the reserved bandwidth is released in those links which do not form part of the tree. At each stage in A2, the bandwidth reserved in the $\kappa-1$ paths not used for attachment is released.

5.2.3. Discussion of results

In Ref. [14], two distributed algorithms for multicast tree construction have been proposed, namely, the $DMCT_C$ heuristic and the $DCMT_{CD}$ heuristic. The authors have shown through simulation that for larger network sizes and sparse multicast groups, the $DCMT_{CD}$ heuristic, which factors in delay into the edge selection metric, outperforms the $DCMT_C$ heuristic. We have therefore compared the performance of the proposed algorithms, A1 and A2, with this $DCMT_{CD}$ heuristic. Since the A1 algorithm has a centralized second phase, we have chosen not to compare it's ACST values with those for A2 and $DCMT_{CD}$. However, it can be intuitively seen that Phase 1 of A1 will be quite fast, since unicast call setups can be parallely initiated from the source to all the destination nodes, whereas this parallelism is not present in either of the other two algorithms. For the purpose of the simulation experiments, algorithms A1 and A2 used the RDM heuristic proposed in [17], as the underlying unicast routing strategy.

From the simulation results, the following salient features can be observed

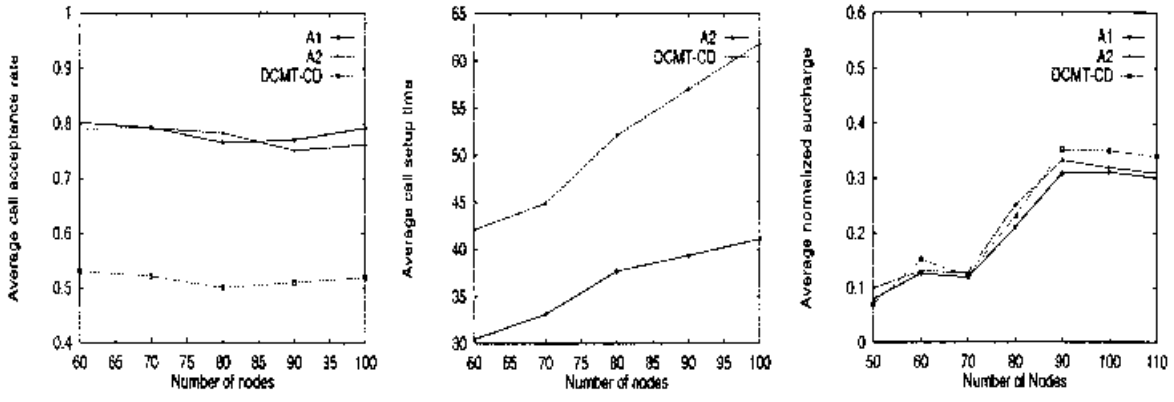

Fig. 3. (a)Effect of network size on ACAR; (b)effect of network size on ACST; and (c)effect of network size on ANS.

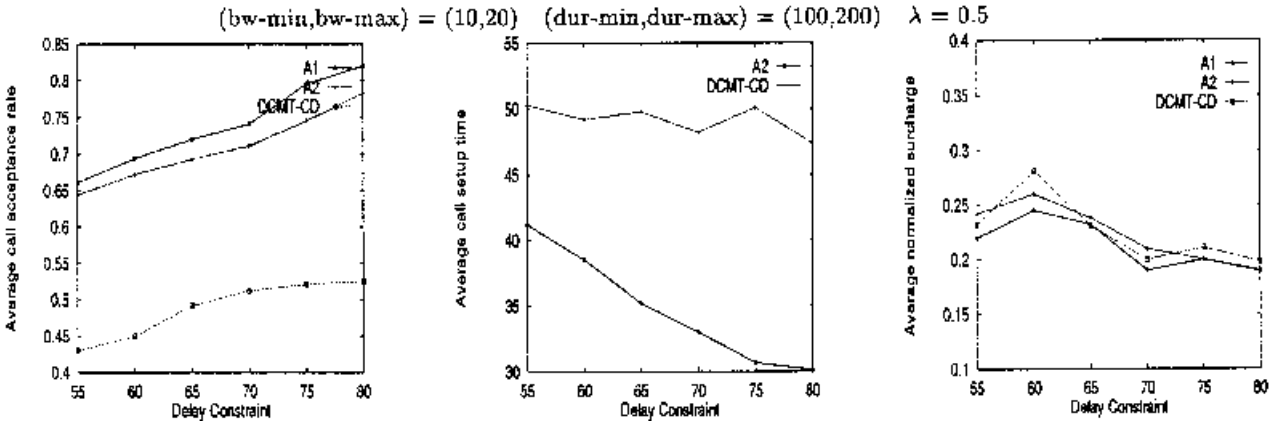

Fig. 4. (a)Effect of delay constraint on ACAR; (b)effect of delay constraint on ACST; and (c)effect of delay constraint on ANS.

1. Network size: Fig. 3(a), (b) and (c) represent the relative performance of the three algorithms when the size of the network is varied. As seen in Fig. 3(a), algorithms A1 and A2 accept a larger percentage of call requests than DCMT$_{CD}$ because of their better management of network resources and lower call setup times (resources are not reserved unnecessarily for a long time). As expected, call setup times for both A2 and DCMT$_{CD}$ increase with increasing network size, as the multicast groups get more widely distributed. The difference between the ACST values for A2 and DCMT$_{CD}$ also increases with increase in network size and, hence, a corresponding increase in the average size of the trees). All three algorithms provide similar performance with regard to the ANS metric with the surcharge not exceeding 35%.
2. Delay constraint: Fig. 4(a), (b) and (c) illustrate the effect of varying the delay tolerance level. For all the algorithms, as expected, call acceptance increases with increase in delay tolerance. As seen in Fig. 4(a), A1 and A2 once again outperform DCMT$_{CD}$, accepting around 30% more calls. Since A2 constructs the multicast tree by attaching one new receiver to the tree at each stage in the algorithm, it's setup times [Fig. 4(b)] are much lower than the setup times of DCMT$_{CD}$ (which requires one cycle of message exchanges for every edge added to the tree). The drop in ACST with increase in delay tolerance is a result of employing the backtracking based RDM heuristic [17] as the unicast strategy. When delay constraints are tighter, the RDM heuristic requires a deeper search in order to establish a delay-constrained path. This results in larger tree setup times.
3. Call arrival rate: Fig. 5(a) and (b) study the effect of varying λ, the call arrival rate, on the ACST and ACAR metrics. The plots indicate that as the demand on the network increases (higher λ), the disparity between the acceptance rates of A1 and A2 and the acceptance rate of DCMT$_{CD}$ increases, suggesting that the proposed algorithms respond better to increased load.
4. Parameter κ: the parameter κ for algorithm A2 is a tunable parameter and provides a tradeoff between ACST and ANS. Thus, depending on the nature of the application, this parameter can be chosen to give more importance either to speeding up tree construction or to lowering tree cost. This is illustrated in Fig. 6(a) and (b), where lower values of κ result in higher ANS but lower ACST.

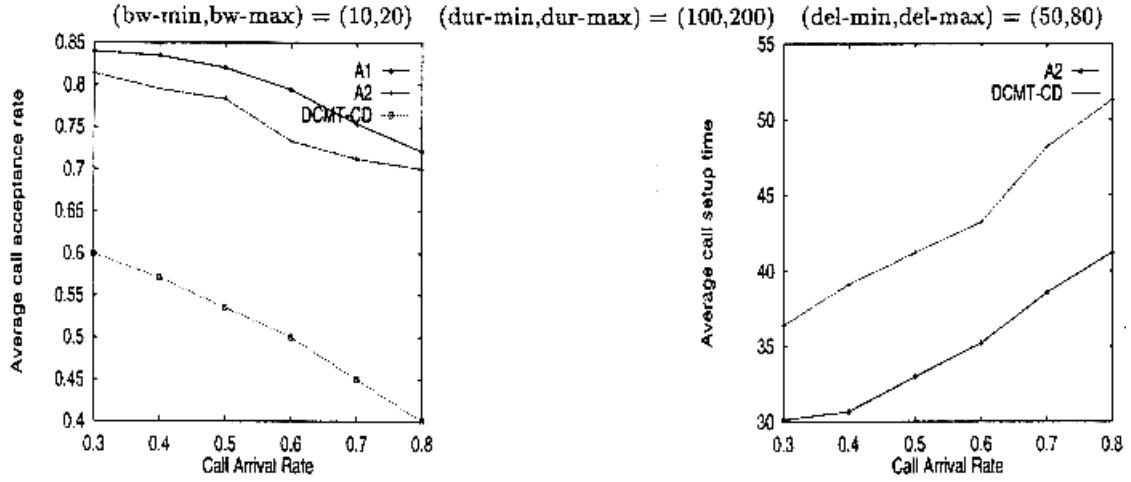

Fig. 5. (a)Effect of call arrival rate on ACAR; and (b)effect of call arrival rate on ACST.

(bw-min,bw-max) = (10,20) (dur-min,dur-max) = (100,200) (del-min,del-max) = (50,80) $\lambda = 0.5$ No. of nodes = 100

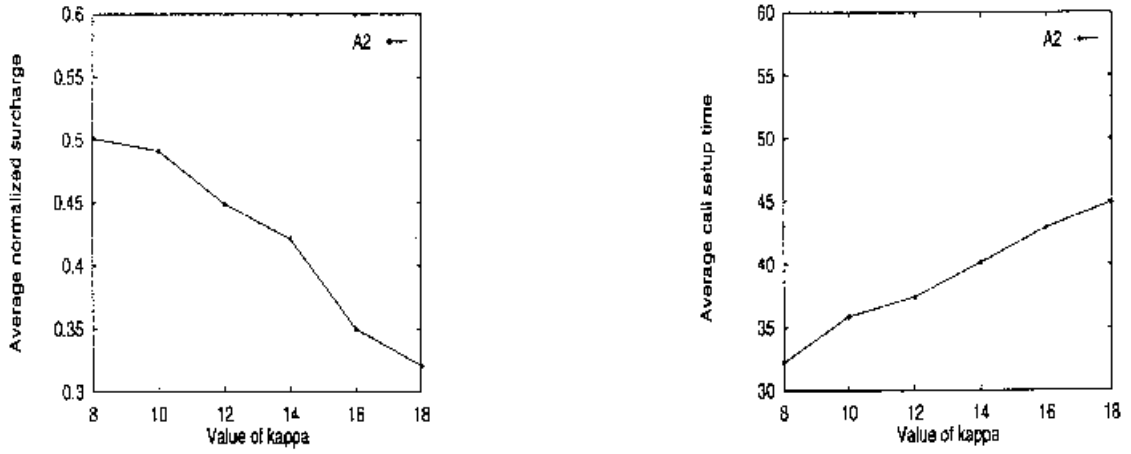

Fig. 6. Effect of κ on AC; and (b)effect of κ on ACST.

In summary, we conclude that the proposed algorithms provide better call acceptance rates and lower call setup times than the best known $DCMT_{CD}$ heuristic. They also provide similar performance with respect to tree cost minimization with all three algorithms restricting the surcharge values (with respect to the MCPH heuristic) to within 35%.

6. Conclusion

In this paper, we proposed two new algorithms for constructing delay-constrained low-cost multicast trees for the distribution of multimedia information to sparse multicast groups. We established the correctness of the two algorithms and presented simulation results that studied their performance (when used in conjunction with the unicast routing heuristics described in [17]). The studies revealed the following advantages of our algorithms over those proposed in [14]

- Our algorithms provide better overall network utilization and higher network throughput as evidenced by the higher call acceptance rate.
- These algorithms also provide lower call setup times (i.e. tree construction time) because of their ability to build up the tree by adding paths (as opposed to the algorithms in [14] which build the tree by adding one edge at a time).
- These algorithms are very flexible and general, in that, they can be used in conjunction with any constrained unicast routing algorithm.
- The second algorithm A2 enjoys the additional advantage of being parameterized by a tunable parameter κ, that represents a tradeoff between call setup time (lower if κ is lower) and average tree cost (lower if κ is higher).

Areas for future research include extending these algorithms to support dynamic multicast groups. Algorithm A2, because it adds one receiver at a time to the tree, is quite suitable for handling additions of multicast group members. However it will need some modification to support the case when group members can also leave the group. It will also be worth investigating whether it is possible, with some minimum global information, to decide on the optimal (in terms of tree cost) order in which destination nodes are to be attached to the tree.

7. Further reading

For further reading see Refs. [8,19].

Appendix A Formal description of algorithm A2

To facilitate the description of this algorithm, we will use the following kinds of messages.

Appendix A.1 Message types:

- Setup message: this is the path setup message that is forwarded along a route chosen by the unicast routing strategy. This message can be characterized by the tuple (src, dest, srcdelay, delayc, next, plist) where the entries have the following meaning:

- src: source of the call setup
- dest: intended destination node
- srcdelay: delay between the source of the multicast and the source of this call setup along the currently constructed tree
- delayc: delay constraint to be satisfied $= \Delta -$ srcdelay
- next: the next destination to be attached to the tree after this one

- list: current priority list.
- Attach message: This is the message used by a destination node to inform the other nodes in the tree about how it is going to attach itself to the tree. This message is forwarded from the destination node along the chosen attachment path and then up the tree from the attachment point to the source node. Such a message can be characterized by the tuple (current, next, plist), where the entries represent the current destination node being attached, the next destination node to be attached and the updated priority list (i.e. the priority list taking into account the nodes in the attachment path).
- DoSetup message: this is the message sent by the source node to some of the other nodes asking them to initiate call setups to the next destination node. This message is characterized by the tuple (dest) which contains the node number of the destination node which needs to be attached next.
- Failure message: this message is sent to a destination node, via the shortest delay path, by some intermediate node which has received a setup packet to be forwarded to that destination. Such a message is sent when the intermediate node decides that it is not possible to reach the destination satisfying the delay and bandwidth constraints. The identity of the node which makes this decision is dependent on the details of the unicast routing strategy.

Notation:

1. Message: a message will be represented by the tuple (type, path, other-entries) where type represents one of the above message types, path represents the list of nodes though which this message has passed and other-entries represent the type-dependent entries as specified in the above definitions. Component x of a message M will be referenced using the notation $M.x$.
2. Msgget(): returns the message received at a particular node.
3. Forward (dest,msg) : sends 'msg' to the destination node 'dest'.
4. Bestpath: every destination node maintains details about the best attachment path (i.e. the path with the lowest selection function value) that it has received upto that point, in this variable.
5. NumReceived: at every destination node, this variable keeps a count of the number of setup and failure messages that it has received upto that point.
6. UpdateList(plist, attachpath): given the existing priority list ''plist'', and the attachment path ''attachpath'', this function returns an updated priority list that could possibly include nodes of the attachment path.
7. UpdateBest (path): when executed by a destination, this calculates the selection function SF for this ''path'' and if this value is lower than the currently encountered best value, the BestPath variable at this destination node is updated.
8. First κ(plist): returns the first κ values in this priority list.
9. Me: at any node, Me refers to the node number of that node.
10. Mydelay: at any node that is part of the multicast tree, this refers to the delay along the tree between the multicast source and this node.

The following procedure describes the actions to be taken by the source node of the tree-establishment request.

Appendix A.2 Actions taken by the source of the multicast

A2-source(s, $R = \{d_1, d_2, \cdots, d_m\}$, Δ, B)

begin

Initiate UR(s,d_1,Δ,B)
while (tree not fully setup) do

M = Msgget()
if (M.type = Setup) then initiate UR(s,M.dest,Δ,B)
else if (M.type = Failure) Forward(M.dest,M)
else if (M.type = Attach) then

remaining = First κ(M.plist) − M.path
for each ($v \in$ remaining)

Forward (v, (Do Probe, M.next))

end

The following procedure decribes the actions of a destination node before it becomes part of the tree. The actions taken by a destination node once it becomes part of the tree are the same as those taken by any other tree node and are specified in the procedure following this. We assume that the destination node does not become part of the tree before it's turn comes. If so, then the procedure can be modified as described in special case 2 of Section 3.3.2.

Appendix A.3 Actions taken by destination nodes

A2-destination(s, R $\{d_1, d_2, d_m\}$, A, B)

begin

Set NumReceived ← and BestPath ← empty
while (tree not fully setup) do

M = Msgget()
if (M.type = Failure) then begin

if (M.type = Me) set NumReceived ← NumReceived + 1
else Forward(M.dest, M)

else if (M.type = Setup) then

if (M.dest = Me) then

Set NumReceived ← NumReceived + 1
UpdateBest(*M*.path)

else Forward(*M*.dest, *M*) /*not yet part of tree*/

if (NumReceived = κ) then

Set the Mydelay variable appropriately using the values in BestPath
newlist = UpdateList(BestPath.plist, BestPath)
Forward(*s*, (Attach, Me, BestPath.next, newlist))

end

The following procedure describes the actions to be taken by any tree node (and the destination nodes once they have joined the tree) on receiving the different types of messages.

Appendix A.4 Actions taken by any other tree node

A2-anyother(*s*, $R = \{d_1, d_2, \cdots, d_m\}$, Δ, *B*)

begin

while (tree not fully setup) do

M = Msgget()
if (*M*.type = DoProbe) then initiate UR(Me, *M*.dest, Δ − Mydelay, *B*)
else if (*M*.type Setup) then initiate UR(Me, *M*.dest, Δ − Mydelay, *B*)
else if ((*M*.type Attach) and (Me ∈ First(*M*.plist)))
then initiate UR(Me, *M*.next, Δ − Mydelay, *B*)

end

References

[1] T. Ballardie, P. Francis, J. Growcroft, Core-based trees (CBT): an architecture for scalable inter-domain multicast routing, Comp. Commun. Rev. 23 (4) (1993) 85–95.

[2] F. Bauer, A. Varma, Distributed algorithms for multicast path setup in data networks, IEEE/ACM Trans. Networking 4 (2) (1996) 181–191.

[3] F. Bauer, A. Varma, ARIES: a rearrangeable inexpensive edge-based on-line Steiner algorithm, IEEE JSAC 15 (3) (1997) 382–397.

[4] C.H. Chow, On multicast path finding algorithms, in: Proc. IEEE INFOCOMM '91, IEEE Communications Society, Bal Harbour, FL, 1991, pp. 1274–1283.

[5] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, L. Wie, The PIM architecture for wide-area multicast routing, IEEE/ACM Trans. Networking 4 (2) (1996) 153–162.

[6] S.E. Deering, D.R. Cheriton, Multicast routing in datagram internetworks and extended LANS, ACM Trans. Comp. Syst. 8 (2) (1990) 85–110.

[7] C. Diot, W. Dabbous, J. Crowcroft, Multipoint communications: a survey of protocols, functions and mechanisms, IEEE JSAC 15 (3) (1997) 277–290.

[8] Y.K. Dalal, R.M. Metcalfe, Reverse path forwarding of broadcast packets, Commun. ACM 21 (12) (1978) 1040–1048.

[9] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco, CA, 1979.

[10] H.F. Salama, D.S.Reeves, Y. Viniotis, A distributed algorithm for delay-constrained unicast routing, in: Proc. IEEE INFOCOMM, IEEE Communications Society, Kobe, Japan, 1997.

[11] F. Hwang, D. Richards, Steiner tree problems, Networks 22 (1992) 55–89.

[12] B.K. Kadaba, J.M. Jaffe, Routing to multiple destinations in computer networks, IEEE Trans. Commun. 31 (3) (1993) 343–351.

[13] V. Kompella, J.C. Pasquale, G.C. Polyzos, Multicast routing for multimedia communication, IEEE/ACM Trans. Networking 1 (3) (1993) 286–292.

[14] V. Kompella, J.C. Pasquale, G.C. Polyzos, Two distributed algorithms for the constrained Steiner tree problem, in: Proc. Comp. Comm. Networks, San Diego, CA, June 1993, pp. 343–349.

[15] L. Kou, G. Markowsky, L. Berman, A fast algorithm for Steiner trees, Acta Informatica 15 (1981) 141–1451.

[16] F. Nicholas, Maxemchuck, Video distribution on multicast networks, IEEE JSAC 15 (3) (1997) 357–372.

[17] R. Sriram, G. Manimaran, C.S.R. Murthy, Preferred link based delay-constrained least cost routing in wide area networks, Comp. Commun. 21 (1998) 1655–1669.

[18] P. Winter, Steiner problem in networks: a survey, Networks 17 (2) (1987) 129–167.

[19] S. Ramanathan, Multicast tree generation in networks with asymmetric links, IEEE/ACM Trans. Networking 4 (4) (1996) 558–568.

[20] G.N. Rouskas, I. Baldine, Multicast routing with end-to-end delay and delay variation constraints, IEEE JSAC 15 (3) (1997) 346–355.

[21] A. Shaikh, K. Shin, Destination driven routing for low-cost multicast, IEEE JSAC 15 (3) (1997) 373–381.

[22] H. Takashami, A. Matsuyama, An approximate solution for the Steiner problem in graphs, Int. J. Math. Educ. Sci. Technol. 141 (1) (1983) 15–23.

[23] B.M. Waxman, Routing of multipoint connections, IEEE JSAC 6 (1988) 1617–1622.

R. Sriram obtained the B.Tech. degree in Computer Science and Engineering from Indian Institute of Technology, Madras, in 1998. He is a recipient of President of India Gold Medal for academic excellence in the B.Tech programme. Currently, he is a research student at the Computer Science Department, Stanford University, USA. His research areas of interests are routing, multicasting, and fault-tolerance in real-time networks.

G. Manimaran obtained the B.E. degree in Computer Science and Engineering from Bharathidasan University, Thiruchirappalli, in 1989, M.Tech. in Computer Technology from the Indian Institute of Technology, Delhi, in 1993, and Ph.D in Computer Science and Engineering from the Indian Institute of Technology, Madras, in 1998. His research interests are resource management in parallel and distributed real-time systems, real-time networks, and fault-tolerant computing.

C. Siva Ram Murthy obtained the B.Tech. degree in electronics and communications engineering from Regional Engineering College, Warangal, in 1982, M.Tech. in computer engineering from Indian Institute of Technology (IIT), Kharagpur, in 1984, and Ph.D. in computer science from Indian Institute of Science (IISc), Bangalore, in 1988. From March 1988 to September 1988 he worked as a Scientific Officer in the Supercomputer Education and Research Centre at IISc. He subsequently joined IIT Madras as a Lecturer of Computer Science and Engineering. He became an Assistant Professor in August 1989 and is currently an Associate Professor at the same place. He has held visiting positions at German National Research Centre for Information Technology (GMD), Sankt Augustin, Germany, University of Washington, Seattle, USA, and University of Stuttgart, Germany. He is a recipient of the Seshagiri Kaikini Medal for the best Ph.D. thesis and also of the Indian National Science Academy Medal for Young Scientists.