Cost-optimal Multicast Trees for Multi-source Data Flows in Multimedia distribution

K. Ravindran, A. Sabbir, D. Loguinov, && G. S. Bloom

Department of Computer Science
City University of New York (City College)
Convent Avenue at 138th Street, New York, NY 10031 (USA)
Contact E-mail address: *ravi@cs.engr.ccny.cuny.edu*

Abstract

From a graph-theoretical perspective, the problem of constructing multicast distribution paths, modeled as 'steiner trees', is NP-complete. So, many heuristics-based algorithms are available to generate near-optimal trees. Typically, an algorithm first assigns the edge costs for all links, and then examines various candidate paths for interconnecting a given set of nodes. This strategy does not work well for the evolving multimedia application configurations (such as audio-video and image distributions) where it is often necessary to construct multiple distribution paths, viz., one per media data stream. This is because the overlapping of multiple tree segments over a common link forces the link cost to change, based on the delay and bandwidth characteristics of data streams flowing over these segments. Accordingly, algorithms that hitherto have assumed non-varying link costs during various phases of a run now need to take into account the variability of link costs as candidate trees with different levels of path overlapping are examined in a given run. In other words, as an algorithm runs by examining various paths, the link costs change. The paper embarks on a study of heuristics-based algorithms to tackle this 'modified steiner tree' problem. The algorithms allow more cost-efficient routing of data than feasible otherwise with a classical treatment of the 'steiner tree' problem.

Keywords: Multimedia distribution trees, 'shortest path' routing, flow & QOS based link costs, 'steiner tree' computation, overlapping trees, heuristic tree algorithms.

1 Introduction

Multipoint routing of data is often realized by *tree-structured paths* over switching nodes and inter-node links. A tree-structured path consist of a root node where the data of a source is made available, a set of intermediate nodes that perform data routing, and one or more destinations at leaf nodes of the tree that consume the data. The links connecting the nodes in a tree, i.e., edges of the tree, should have enough bandwidth to support the data rate of the source (e.g., a bandwidth of 2.5 *mbps* needed for a link to carry compressed video data). The tree that carries data, i.e., multicast tree, provides the basic network capability for multi-destination data delivery, as required by distributed applications (e.g., clients accessing a replicated multimedia web server, users interacting with one another in a conference session) [1]. See Figure 1.

From a graph-theoretical perspective, the multipoint routing problem may be defined as follows. Let $\mathcal{P}(\mathcal{V}, \mathcal{E})$ be an undirected network consisting of nodes $\mathcal{V}$ interconnected by communication links $\mathcal{E}$, with a link cost assignment function $c_e|_{\forall e \in \mathcal{E}}$ (say, based on bandwidth allocation). Given a set of nodes $\mathcal{U} \subseteq \mathcal{V}$ that contain user entities, i.e., nodes representing data sources and destinations, it is necessary to find a tree spanning the nodes $\widehat{\mathcal{U}}$ and links $\widehat{\mathcal{E}}$ such that $\mathcal{U} \subseteq \widehat{\mathcal{U}} \subseteq \mathcal{V}$ and $\sum_{\forall e' \in \widehat{\mathcal{E}}} c_{e'}$ is a minimum. Such a tree is called a minimum *steiner tree* for $\mathcal{U}$ in the topology $\mathcal{P}$ [2]. Given a configuration, it may often be necessary to construct multiple distribution paths: one-per-source, for multicasting from multiple sources to a common set of destinations (e.g., multimedia conferencing), and one-per-stream for multicasting multiple streams of data from a source to a set of destinations (e.g., audio+video in digital TV broadcast).

Combinatorics researchers have shown the steiner tree (ST) problem to be NP-complete [3, 4]. Since then, many heuristic-based algorithms have been proposed to generate near-optimal trees [5, 6, ?]. Typically, an algorithm first assigns the edge costs for all links, and then examines various candidate paths for interconnecting a given set of user entities. From among these paths, the algorithm chooses a path with the minimum cost. It is not guaranteed however that the algorithm exhaustively searches the space of all possible paths. So there may exist paths that incur less cost than the path declared as 'cost-minimal' by the algorithm.

In this paper, we analyze the scope of the ST problem

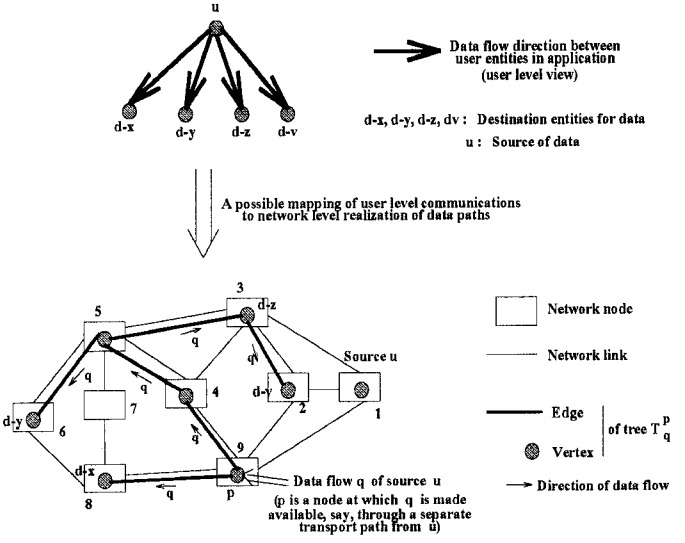

Figure 1: Tree-structured paths for multicasting

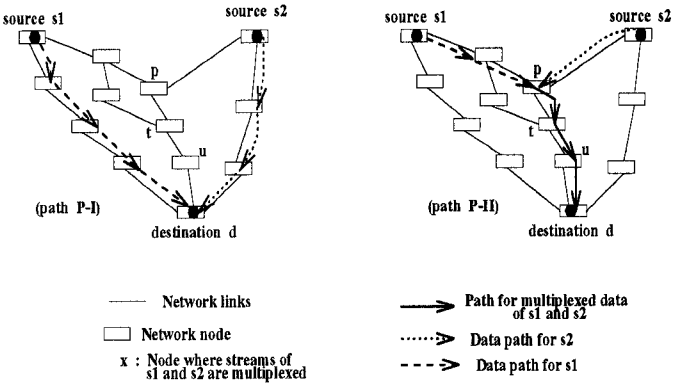

Figure 2: 'path sharing' to reduce transport cost

and solutions, in light of the evolving multimedia applications that require multipoint communications among users and the network strategies that attempt to optimize the consumption of underlying communication resources. As we shall see, the emerging characteristics of multimedia applications and multi-service networks casts the ST problem with a more complex model of link cost assignments than what has been assumed in a hitherto classical treatment of the problem.

It is possible that the trees generated for various streams have some of their paths *overlapping* with one another on a common link. An overlapping path depicts the sharing of the underlying link resources, thereby influencing the overall link cost assignment, and hence the tree generation itself. For instance, the fixed cost of using a link (e.g., network tariff per unit of 'connect time') can get amortized across the streams that share this link. Furthermore, a statistical multiplexing of various data streams flowing over the link is possible, which allows reducing the per-stream resource allocations, particularly with bursty data streams. For instance, video streams from 2 MPEG sources can be supported with less bandwidth on a shared tree than that possible if they are sent on separate trees.

ST algorithms previously have assumed constant link costs during various phases of a given run. We however believe that more efficient routing will result if one takes into account the variability of link costs arising from the sharing of links by multiple trees. So link cost variability needs to be factored into ST algorithms, as candidate trees with different levels of path overlapping are examined. The changing link costs may influence the overall tree setup. Also, when a new source joins or when a receiver changes its bandwidth demand, the global cost optimality of the tree may be affected, possibly triggering a tree reconfiguration.

Consider, as illustration, a channel set up for connecting sources s_1 and s_2 to a destination over the physical topology, as shown in Figure 2. Suppose s_1 and s_2 generate bursty data flows with a per-hop bandwidth of 1 unit (normalized), and that a 35% bandwidth gain is achievable by multiplexing these streams over a shared link[1]. If the data of s_1 and s_2 are considered individually, the path P-I consisting of 4 hops and 3 hops respectively will be set up with a total bandwidth of 7 units (normalized). With link sharing between these data taken into account, the path P-II consisting of 5 hops and 4 hops respectively with an overlap of 3 of their edges will be set up, requiring a total bandwidth of 6.9 units. Consider an ST algorithm $\mathcal{A}$ that finds these paths. Taking into account s_1 alone to start with, $\mathcal{A}$ treats the 4-hop path of P-I as cost-minimal and eliminates the path through the nodes p, t and u since it contains 5 hops. When $\mathcal{A}$ next considers s_2 also, it needs to re-evaluate the cost efficacy of the path through p, t and u when shared by s_1 and s_2. For doing so, $\mathcal{A}$ needs to reduce the bandwidth allocation cost of the links p-to-t, t-to-u, and u-to-d as apportioned to s_1 from 1 unit to 0.7 unit each (and so for s_2 also). With this changed link cost assignment, this shared path does in fact turns out to be more cost-minimal now.

The higher degree of cost optimality achievable in tree constructions by taking into account the impact of path sharing on link costs allows resource-efficient routing — in a network-wide sense, particularly in the case of geographically distributed multimedia applications (such as remote class rooms over Internet). The change of link costs with respect to the degree of link sharing however presents a new dimension to the complexity of 'tree generation' problems, because of the need to compute the cost variability of each link with respect to the number of flows sharing it (besides computing a minimal path length)[2]. This complexity calls

[1]For example, a link carrying 2 compressed video data streams with a peak rate of 3 *mbps* each will need to allocate a sustained bandwidth of 4.2 *mbps*.

[2]Even with using a weighted sum of tree edges, an ST algorithm may take into account only the cost differences from

for a re-examination of currently available ST algorithms and a possible introduction of new heuristics and/or modification of existing heuristics. Our paper walks through these problems and offers solutions.

The paper first develops a model of determining the cost of multicast trees. The model factors in the topological configuration of trees and the sharing of paths by various data streams. Using the model, we then embark on a study of heuristic-based algorithms to tackle the 'modified steiner tree' problem.

2 'cost' notions in multicast networks

We first present a canonical view of multicast functions in the network that epitomizes the evolving multimedia communication strategies. That the development of 'shortest tree' algorithms has assumed an additional twist can be seen through this view.

2.1 Macro-level parameters for tree setup

The physical topology of the underlying network $\mathcal{P}(\mathcal{V},\mathcal{E})$ consists of a set of nodes $\mathcal{V}$, connected with one another through a set of links $\mathcal{E}$. The user entities, viz., sources and destinations of multimedia data, reside in distinct nodes $\mathcal{U} \subseteq \mathcal{V}$, and form the communication endpoints. If $\mathcal{U}_s$ and $\mathcal{U}_d$ are the nodes containing data sources and data destinations respectively, then $\mathcal{U}_s, \mathcal{U}_d \subseteq \mathcal{U}$. A source can generate one or more data streams (e.g., a multimedia workstation generating video and audio streams), and a receiver may consume the data streams generated by various sources. The tuple $(\mathcal{U}_s, \mathcal{U}_d, \mathcal{V}, \mathcal{E})$ may be viewed as prescribing a *configuration*, i.e., the placement of sources and destinations relative to one another in physical topology of the network.

When a link $l \in \mathcal{E}$ is included in a multicast tree $\mathcal{T}$, we say that l supports a path segment $(l, \mathcal{T})$. Since l may be part of multiple trees $\mathcal{T}_1, \mathcal{T}_2, \cdots$, the bandwidth capacity of l — denoted as CAP(l) — is partitioned into chunks $b(l,\mathcal{T}_1), b(l,\mathcal{T}_2), \cdots$ as required by the data streams flowing over these trees and allocated therein (e.g., a 45 *mbps* DS3 line carrying 3 MPEG-2 video streams of 2 *mbps* each). An algorithmic constraint for determining whether l can be included as part of a new multicast tree $\mathcal{T}'$ is then: $b(l,\mathcal{T}') \leq [\text{CAP}(l) - \sum_{\forall i} b(l,\mathcal{T}_i)$ — i.e., l should have enough left-over bandwidth capacity to transport the data flowing over $\mathcal{T}'$. Thus, that l lies in the shortest path (in terms of the number of hops) between a source and a receiver does not imply that l will be included as part of the tree connecting them.

The cost of $(l, \mathcal{T})$ is directly related to the amount of bandwidth allocated by l to transport the data of $\mathcal{T}$ over l, as given by a relation:

$$c : \mathcal{B} \rightarrow \mathcal{Z} \quad \text{for } \mathcal{Z} \subseteq \mathcal{R}^+, \tag{1}$$

where the cost of an incremental bandwidth allocation δb satisfies the condition:
$[c(b_2 + \delta b)_l - c(b_2)_l] \geq [c(b_1 + \delta b)_l - c(b_1)_l]$ for $b_2 > b_1$. To satisfy this condition, c may be drawn from a space of 'monotonic convex' functions $\mathcal{C}$, to represent a specific policy implemented by the network. For simplicity, we assume only the limiting case in this paper.

2.2 'steiner trees' for multicast routing

The steiner tree for a configuration consisting of user nodes $\mathcal{U}$ placed in a physical topology $\mathcal{P}(\mathcal{V},\mathcal{E})$ is an *acyclic graph* $\mathcal{G}(\widehat{\mathcal{E}})$ connecting a set of nodes $\widehat{\mathcal{U}}$ through a set of links $\widehat{\mathcal{E}}$ such that $\sum_{\forall e \in \widehat{\mathcal{E}}} c(b(e,\_))_e$ is a minimum, where $\widehat{\mathcal{E}} \subseteq \mathcal{E}$ and $\mathcal{U} \subseteq \widehat{\mathcal{U}} \subseteq \mathcal{V}$.

Let $L(x,y) \subseteq \widehat{\mathcal{E}}$ be a path consisting of a set of adjacent links that connect a pair of nodes $x, y \in \widehat{\mathcal{U}}$. Since $\mathcal{G}$ is acyclic, there is exactly one $L(x,y)$ for any given x and y. A multicast path for carrying a data flow q from a node p to a set of receivers $\mathcal{U}_d$ is then a subtree projected from $\mathcal{G}$ with root at p and leaves at nodes $\mathcal{U}_d$, given as:

$$\mathcal{T}_q^p(\mathcal{G}) = \bigcup_{\forall y \in \mathcal{U}_d} L(y,p).$$

To support the multicast flow of q from p to $\mathcal{U}_d$, a steiner tree is first created to connect the nodes $\{p, \mathcal{U}_d\}$ through a set of nodes $\widehat{\mathcal{U}}$ and links $\widehat{\mathcal{E}}$, which is then projected into a multicast distribution tree with root at p and leaves contained in $\mathcal{U}_d$, where $\mathcal{U}_d \subseteq \widehat{\mathcal{U}} \subseteq \mathcal{V}$ and $\widehat{\mathcal{E}} \subseteq \mathcal{E}$. Note that p need not be the same as the node $u \in \mathcal{U}_s$ where the source that generates q resides (such as nodes 1 and 9 in Figure 1). In[3] Figure 3, the multicast trees are $\mathcal{T}_{q_a}^{x_a} = \widehat{\mathcal{E}}\ \mathcal{T}_{q_a}^{x_b} = (\widehat{\mathcal{E}} - L(x_a, w)$. The topological structure of a multicast path depends on the configuration parameter $(\mathcal{U}_s, \mathcal{U}_d, \mathcal{V}, \mathcal{E})$ and the bandwidth cost parameter $\{c(b(e))\}_{\forall e \in \mathcal{E}}$. For the cases of $\{p, \mathcal{U}_d\} = \mathcal{V}$ and $|\mathcal{U}_d| = 1$, the tree construction can be done in polynomial time — such as Kruskal's 'minimum spanning tree' (MST) algorithm in the former case and Dijkstra's 'shortest path' algorithm in the latter case [7]. Tree construction in other cases is an NP-complete problem [3, 4]. So heuristic-based algorithms are employed that construct close-to-optimal trees. A variety of such algorithms have been studied elsewhere [8, 9, 6, 10, 11].

one link to another, but not the non-linear effects of bandwidth sharing across flows in each link.

[3] A word about the term 'tree', as used in this paper, is in place here. 'steiner tree' is a graph-theoretic term referring to an acyclic graph constructed over a network of nodes and links, whereas 'multicast tree' is a (network) protocol-oriented term referring to the data flow path from a source to a set of receivers. This paper treats 'multicast trees' as graph-theoretic projections derived from 'steiner trees'.

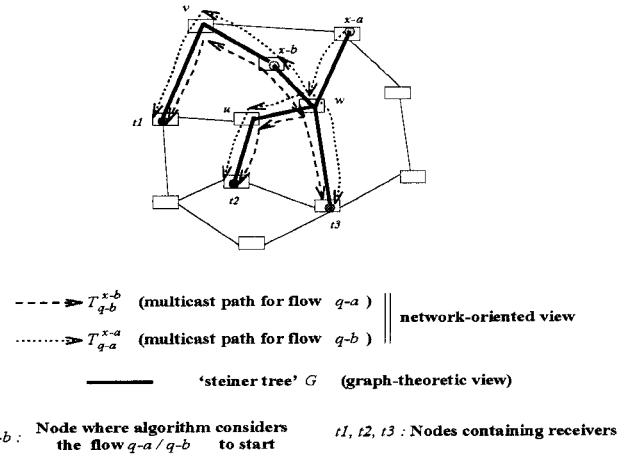

Figure 3: Multicast paths projected from steiner trees

2.3 Cost incurred for multicast transport

A cost assignment policy allows mapping the flow-specific resource allocations to a link cost, which may then be used as input to network algorithms for computing resource-efficient data paths.

A relation $f \in \mathcal{F}$ maps the data rate of the streams flowing along a tree edge $(l, \mathcal{T})$ to the bandwidth needs for transporting these streams through the connection set up over the link l:

$$f : r \rightarrow \mathcal{B} \quad \text{for } r, \mathcal{B} \subseteq \mathcal{R}^+ \text{ (i.e., positive real numbers)}, \quad (2)$$

where r represents the data rate of the stream to flow along $(l, \mathcal{T})$ and $f(r)$ is the required bandwidth allocation on l. For example, when r is prescribed as '2 *mbps* average' and '5 *mbps* peak' with a loss tolerance of 5%, an allocation policy may determine $f(r) = 3.5$ *mbps* (say). The function f satisfies the 'monotonicity' and 'boundedness' condition: $f(r) < f(r + \delta r) \leq f(r) + f(\delta r)$ for $\delta r > 0$. The function space $\mathcal{F}$ may represent a family of bandwidth allocation policies implementable by the network.

The network-wide cost of transporting data over a tree $\mathcal{T}_q^-$ may then be estimated as:

$$\text{tot\_cost}(\mathcal{T}_q^-) = \sum_{\forall (l, \mathcal{T}_q^-)} c(f(q))_l, \quad (3)$$

where $f(q)$ is the required bandwidth allocation attributed to q. If $\mathcal{T}_q^p[1], \cdots, \mathcal{T}_q^p[K]$ are the trees that can carry the flow q from a node p to a set of receivers, a tree $\mathcal{T}_q^p[i]$ is cost minimal iff:

$$\text{tot\_cost}(\mathcal{T}_q^p[i]) \leq \text{tot\_cost}(\mathcal{T}_q^p[j])|_{j=1,\cdots,K}.$$

Such a tree forms a minimum steiner tree for $\mathcal{U}$ in the topology $\mathcal{P}$ [2]. How to find the steiner tree is a fundamental algorithmic problem in multicast routing.

2.4 Current 'steiner tree' algorithms

To construct a tree $\mathcal{T}_q^p$, an algorithm may start by assigning edge costs as: $c(f(q))_l \quad \forall l \in \mathcal{E}$. This requires $|\mathcal{U}_s| \times |\mathcal{E}|$ steps. The statically assigned link costs and the placement of $\mathcal{U}$ in physical topology are used as input parameters to the algorithm.

Some of the algorithms are 'shortest distance routing' and 'truncated MSTs'. In the 'shortest distance routing', the least cost path from p to each $u' \in \mathcal{U}_d$ is determined, and then all these paths are merged. In 'truncated MSTs', a MST is first constructed for $(\mathcal{V}, \mathcal{E})$, and then, edges that do not connect p to any of the nodes $\mathcal{U}_d$ are removed from the MST.

Another method is to consider a set of networks, each of which containing the nodes p and $\mathcal{U}_d$, and a subset of the nodes $(\mathcal{V} - \mathcal{U}_d - \{p\})$. The minimum of the MSTs for all such networks may then be used as the cost-optimal multicast tree. This method is computationally quite expensive when $|\mathcal{U}_d| \ll |\mathcal{V}|$, as in the case of video distribution over the Internet [12].

A common theme in these algorithms is that the edge costs for all links are assigned before exploring various candidate trees. So, if the routing problem can be reduced to a form that allows static assignment of link costs (or weights), then the problem can be solved with the same level of cost minimality achievable by a chosen algorithm[4].

How the basic 'steiner tree' (ST) problem manifests itself from an algorithmic complexity standpoint in light of the evolving multimedia applications is described next.

3 Topology overlap of multicast trees

When a 'steiner tree' is created with the intent of carrying data flows from multiple sources, the traffic interactions between these flows makes the link cost assignment (and hence the tree construction) more complex than the case of carrying a single data flow[5]. In this section, we provide a graph-theoretic treatment of this problem.

3.1 Sharing of multicast paths

Consider the path segments of $\mathcal{T}_{q_a}^{x_a}$ and $\mathcal{T}_{q_b}^{x_b}$ carrying the data flows, say, q_a and q_b respectively to a receiver $t \in \mathcal{U}_d$. The shared path carrying the aggregation of q_a and q_b, denoted as $\mathcal{T}_{q_a \oplus q_b}^{z}(\mathcal{G})$, is said to consist of overlapping segments of the multicast trees rooted at merge point node z and carrying the flows q_a and q_b to the receiver t. Referring to Figure 3, the trees overlap on all the

[4]Constructing a tree with minimum number of edges is a special case where the link weights are equal.

[5]Throughout this paper, the term 'cost' refers to the amount of network resource allocation and transport overhead incurred for a data flow through a multicast tree, and not to the run-time overhead of an algorithm that computes a multicast tree. The latter is referred to as 'algorithmic complexity' in the paper.

links except $L(x_a, w)$. Such trees are supported in many multicast protocols: the 'core-based tree' [13], 'protocol independent multicast' [14] and 'multi-rooted tree' [15].

The sharing of link resources between multicast trees reduces the overall link cost assignment. For instance, the fixed cost of using a link (e.g., network tariff per unit of 'connect time') can get amortized across the streams that share this link. Furthermore, effective 'statistical sharing' of resources across the various data streams flowing over a path is possible with a knowledge of 'traffic correlation' between them. This allows reducing the per-stream resource allocations, particularly when data is bursty.

For a given source-destination placement, the bandwidth savings due to overlap of multicast trees for various component flows may accrue if the trees have many overlapping segments, do not contain significantly more number of hops than the non-overlapping trees constructed individually, and the per-hop bandwidth gains due to 'statistical multiplexing' are high. In other cases, it may be more cost-effective to construct non-overlapping trees with fewer hops. Referring to Figure 2, the cost-minimal path P-II has 5 hops each for s_1 and s_2, whereas the path P-I has only 4 hops. Thus with path-sharing across data flows, cost minimality does not imply minimality of the number of hops in the path of each flow[6].

3.2 A model for reduction in edge costs

Suppose a new data stream q_k is injected into a tree $\mathcal{T}_{\oplus\{q_1,\cdots,q_{k-1}\}}$ that causes q_k to flow along with the on-going stream $\oplus\{q_1,\cdots,q_{k-1}\}$ over a tree edge $(l, \mathcal{T}_{\oplus\{q_1,\cdots,q_{k-1}\}})$, where $k \geq 2$. This flow aggregation causes an increase in the bandwidth allocation on l, as[7] governed by the condition:

$$f(\oplus\{q_1,\cdots,q_{k-1}\}) \;<\; f(\oplus\{q_1,\cdots,q_{k-1},q_k\}) \;\leq\; \sum_{i=1}^{k} f(q_i).$$

The 'weak additivity' captures the bursty nature of data flows, wherein the new q_k may share part of the bandwidth allocated for the on-going $q_1 \oplus \cdots \oplus q_{k-1}$, thereby accrueing bandwidth savings. Referring to the example illustrated in Figure 2, $f(q_1 \oplus q_2) = 1.4$ with 30% bandwidth savings due to 'statistical multiplexing' of q_1 and q_2 and with $f(q_1) = 1$ and $f(q_2) = 2$. The transport costs to capture the path overlapping between the k multicast trees may then be stated as:

$$\forall l \in \widehat{\mathcal{E}}\;\; \forall c \in \mathcal{C}\; \forall f \in \mathcal{F}\;\; c(f(q_1 \oplus q_2 \oplus \cdots \oplus q_k))_l \;\; < \\ [c(f(q_1 \oplus \cdots \oplus q_{k-1}))_l \;+\; c(f(q_k))_l]. \quad (4)$$

This cost reduction is not possible when a separate 'network connection' needs to be maintained for each data stream, as with non-overlapping path segments created on a link.

A bandwidth allocation relation f is superimposed upon by a cost assignment relation c that satisfies the condition:

$$\forall b_1, b_2 \in \mathcal{B}\; [c(b_2 + \delta b)_l - c(b_2)_l] = [c(b_1 + \delta b)_l - c(b_1)_l].$$

The simplest cost relation that satisfies the condition (4) takes the form: $c(f(q))_l = m + f(q)$ where m is a constant depicting a fixed cost of setting up a 'network connection' over the link l to participate in the tree $\mathcal{T}_q$. Even in the limiting case of 'strongly-additive' bandwidth allocation, as given by: $f(\oplus\{q_1, q_2, \cdots, q_k\}) = f(q_1) + f(q_2) + \cdots + f(q_k)$, the cost attributed to a flow q_i arising from an aggregation of the k flows may be given as

$$c(f(q_i))_l \;=\; \frac{m}{k} + f(q_i).$$

Since $m > 0$, the cost of q_i decreases with an increase in k. This cost behavior has the same effect on a tree construction algorithm as a composite cost behavior superimposed upon an arbitrary level of weak additivity in bandwidth allocations.

Figure 4 illustrates the bandwidth savings arising from a variety of flow aggregation policies and how our simplified model can realize the same cost variability effects on the complexity aspects of any OVT algorithm.

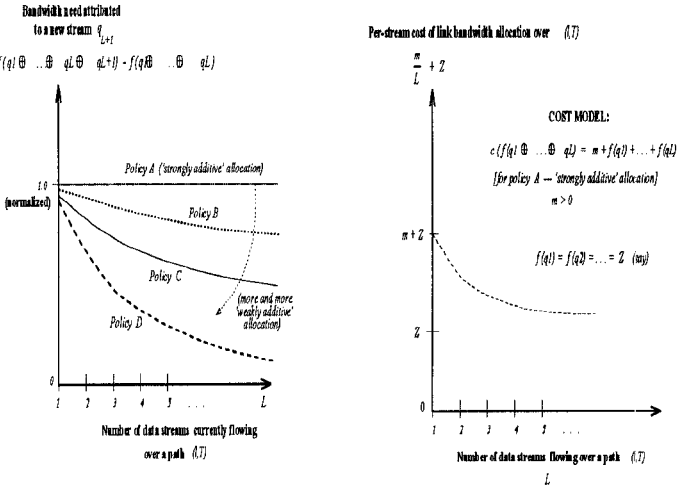

Figure 4: Link bandwidth allocation policies and a simplistic cost assignment for OVTs

[6] An example of analogy is the 'ride-share' by commuters traveling to workplaces. Consider 2 commuters A and B traveling to a workplace W. They may ride in separate cars to a 'park-and-ride' lot R and travel therefrom in a single car to W. In many cases, this car-sharing may be more cost-effective than A and B riding in separate cars all the way up to W, even though the total distance traveled to R and then on to W may be higher for A and/or B. The cost-effectiveness stems from amortization of fixed costs such as 'highway tolls', 'car wear-and-tear' and 'driver fatigueness'.

[7] For the purpose of algorithmic analysis, we assume (without loss of generality) that each link in the topology has enough bandwidth capacity available to satisfy any bandwidth request that may arise due to flow aggregation.

3.3 Cost reduction of overlapping trees

Let $\mathcal{T}_{q_1}$ and $\mathcal{T}_{q_2}$ be the trees carrying the data flows q_1 and q_2 separately to receivers $\mathcal{U}_d$ and rooted at the source nodes s_1 and s_2 ($\in \mathcal{U}_s$), respectively. Refer to Figure 2. Suppose an alternate set of two paths with some of their segments overlapping one another is possible. Let $\mathcal{T}^p_{q_1 \oplus q_2}$ denote the overlapping part of these trees, rooted at a node p and with leaves at $\mathcal{U}_d$. We say p is a *cost-feasible* placement iff:

$$[\sum_{\forall x \in L(s_1,p)} c(f(q_1))_x + \sum_{\forall t \in L(s_2,p)} c(f(q_2))_t + \sum_{\forall u \in \mathcal{T}^p_{q_1 \oplus q_2}} c(f(q_1 \oplus q_2))_u] < [\sum_{\forall v \in \mathcal{T}_{q_1}} c(f(q_1))_v + \sum_{\forall w \in \mathcal{T}_{q_2}} c(f(q_2))_w] \quad (5)$$

for some cost assignment policy c on various links in a path, where $L(a,b)$ denotes the 'shortest' path between a pair of nodes a and b. The[8] cost of a link l attributed to a flow, say q_1, reduces from $c(f(q_1))_l$ to $[c(f(q_1 \oplus q_2))_l - c(f(q_2))_l]$ when q_1 starts sharing l with q_2. Accordingly, the cost comparison given by (5) requires an enumeration of all possible shared trees as prescribed by the cost-feasible placements of p under the policy $c \in \mathcal{C}$. The candidate paths to be examined by a routing algorithm may then be given by

$$\{L(s_1,p) \bigcup L(s_2,p) \bigcup \mathcal{T}^p_{q_1 \oplus q_2}\}_{\forall p \in pset(\{s_1,s_2\},\mathcal{U}_d,f,c)},$$

where $pset(\{s_1,s_2\},\mathcal{U}_d,f,c)$ is the set of cost-feasible placements. To determine $pset$, an algorithm needs information about various feasible levels of flow aggregation of q_1 and q_2 (viz., $\oplus\{q_1\}, \oplus\{q_2\}, \oplus\{q_1,q_2\}$) over each link in the physical topology.

Suppose a source s_3 needs to join the multicast session. Two possibilities of bandwidth allocation arise: either to have the path of q_3 (generated by s_3) overlap the existing tree with an aggregated allocation or to create a separate non-overlapping tree for q_3 with individual allocation. The former case is possible if p continues to be a cost-feasible placement with the additional bandwidth allocation for q_3, and the latter case is necessary if no cost-feasbile placement can be found. In other words, if the incremental cost of aggregating q_3 with $q_1 \oplus q_2$ to send over the current tree $\mathcal{T}^p_{q_1 \oplus q_2}$ turns out to be higher than the cost of sending q_3 over a separate tree, the algorithm creates this non-overlapping tree for q_3. Before either case however, the algorithm should establish the cost feasibility or otherwise of p for the flow $q_1 \oplus q_2 \oplus q_3$. We are interested in determining the algorithmic complexity involved in this extra computation.

We now provide an algorithmic treatment of the extended steiner tree problem.

[8]Note that a cost-feasible placement of p may not exist in some toplogies, such as 'ring'.

4 Dynamic link cost based algorithms

Consider the multicast routing of data flows $\{q_i\}_{i=1,2,\cdots,N}$. With a classical ST algorithm, link costs are assigned at start of the algorithm run for each flow q_i as: $c(f(q_i))_l \ \forall l \in \mathcal{E}$, and then alternate paths are examined to determine a cost-minimal tree for q_i (link costs do not change during examination of various paths). The trees so constructed for each q_i are non-overlapping trees (NVT). The NVTs however need not be cost-optimal (in a global sense), since the algorithm does not take into account link cost reductions that are feasible by overlap of tree edges — c.f. relation (4). To overcome this limitation, an OVT construction algorithm should compose together the N instances of NVT creations, as determined by the cost reduction criteria due to topological overlap of the multicast paths.

4.1 Modified steiner tree algorithms

Consider a classical ST algorithm $\mathcal{A}'$ that generates NVTs (by considering each source $\in \mathcal{U}_s$ separately). When used for generating an OVT, $\mathcal{A}'$ is faced with a possibility that link costs change as different candidate paths are explored. In other words, when a given run of $\mathcal{A}'$ explores alternate candidate paths, the cost of paths considered earlier in the run can change. The changeability of link costs may require revalidation of the paths already considered as cost minimal, thereby influencing the generation of final tree itself. Thus to achieve better cost optimality, $\mathcal{A}'$ should be augmented with the functionality of re-examining various cost-feasible paths in the presence of link cost changes.

An OVT construction algorithm $\mathcal{A}$ should be able to enumerate the changes in links costs as it sifts through various candidate paths during a run. This requires information about what data flows can be aggregated over each link. The flow aggregation information is not available to $\mathcal{A}'$ since it merely considers the placement of $\mathcal{U}$ in the physical topology, without regard to which of them are sources and which of them are receivers. Accordingly, $\mathcal{A}$ needs to take into account the relative placement of $\mathcal{U}_s$ and $\mathcal{U}_d$ in the physical topology. Equipped with this additional information, $\mathcal{A}$ can examine new candidate paths carrying the aggregated data flows of various subsets of sources $\in \mathcal{U}_s$ to the receivers $\mathcal{U}_d$, in order to incrementally estimate a cost-minimal subtree. $\mathcal{A}$ may then re-validate the cost minimality of previously estimated subtrees in the context of the new subtree. This may proceed until a tree is found that connects all the nodes $\mathcal{U}_s$ and $\mathcal{U}_d$.

4.2 A general structure of OVT algorithm

We treat a run of $\mathcal{A}$ as consisting of multiple computational steps, with each step requiring a re-estimated link cost assignment relative to the previous step that is used in generating a tree. However, the continued cost minimality

of a tree set up in the previous step is to be ascertained with the re-estimated costs.

A general algorithmic rule for reassigning link costs is to treat each edge in the current tree $\mathcal{T}_q$ as potentially overlappable with an edge in a tree $\mathcal{T}_{q'}$ to be set up in the next step. The cost of each such link $l \in \widehat{\mathcal{E}}(\mathcal{T}_q)$ is then determined from $c(f(q \oplus q'))_l - c(f(q))_l$, whereas the cost of each remaining link $t \in (\mathcal{E} - \widehat{\mathcal{E}}(\mathcal{T}_q))$ is set as $c(f(q'))_t$. With this cost assignment, a 'steiner tree' is generated for estimating the path $\mathcal{T}_{q'}$ (using NVT algorithm). Because of the condition:

$$\forall l \in \mathcal{E}\ \forall c \in \mathcal{C}\ \forall f \in \mathcal{F}$$
$$[c(f(q \oplus q'))_l - c(f(q))_l] \ < \ c(f(q'))_l, \qquad (6)$$

the placement of $\mathcal{T}_{q'}$ is always gravitated towards that of $\mathcal{T}_q$ (in comparison to the case of generating a $\mathcal{T}_{q'}$ independent of $\mathcal{T}_q$). Since $\mathcal{T}_{q'}$ may span only a subset of the links that were considered as potentially overlappable with $\mathcal{T}_q$, the links not in $\mathcal{T}_{q'}$ are restored to the cost assignment $c(f(q))$. Having generated $\mathcal{T}_{q'}$, we need to ascertain if $\mathcal{T}_q$ continues to be the cost-minimal path in the presence of edge overlapping with $\mathcal{T}_{q'}$. For this purpose, the algorithm assigns a cost of $c(f(q \oplus q'))_l - c(f(q'))_l$ to all links $l \in \widehat{\mathcal{E}}(\mathcal{T}_{q'})$ and a cost of $c(f(q))_t$ to the remaining links $t \in (\mathcal{E} - \widehat{\mathcal{E}}(\mathcal{T}_{q'}))$. With the new cost assignment, a 'steiner tree' $\mathcal{T}_q$(new) is computed for q. Now with $\mathcal{T}_q$(new), a new 'steiner tree' $\mathcal{T}_{q'}$(new) is computed. These steps are repeated until the new trees do not yield a cost reduction over the trees created in the previous step.

4.3 Walk-through of a sample scenario

Figure 5 shows a sample scenario involving video distribution, to illustrate how path sharing can change link costs, and hence influence tree setups. In the cost model $c(f(q)) = m + f(q)$, we assume $m = 2$ for all links except X and $m = 2.5$ for the link X, and $f(q) = 1$. A run of the tree setup algorithm $\mathcal{A}$ has three steps: I, II and III. In step-I, the algorithm first assigns a cost of $(2+1)$ units to each link in physical topology and generates a tree $\mathcal{T}_q$ therefrom with 5 edges; the cost is 15. In step-II of the run, the algorithm considers a second flow q'. The costs are assigned as $(1+1)$ units for links in the tree carrying q, to take into account the amortization of m by a factor of 2 arising from potential edge overlapping between $\mathcal{T}_q$ and $\mathcal{T}_{q'}$; for the remaining links, the link cost assignment is $(2+1)$ units. The tree generated $\mathcal{T}_{q'}$ has 4 edges, with 3 of them overlapping with $\mathcal{T}_q$, yielding a total cost of 9. The remaining two links are restored the cost of $2+1$ each. Because of the edge overlap, the cost of $\mathcal{T}_q$ reduces to 12. To ascertain if $\mathcal{T}_q$ continues to be the cost-minimal tree, the step-III of $\mathcal{A}$ reassigns the links cost of the non-overlapping edge of $\mathcal{T}_{q'}$ to $1+1$ units, treating it as potentially overlappable with a reconfigured $\mathcal{T}_q$. Now $\mathcal{A}$ finds that a new $\mathcal{T}_q$ with edge overlap on this link does reduce the cost of $\mathcal{T}_{q'}$ and $\mathcal{T}_q$ to 8 and 11.5 respectively

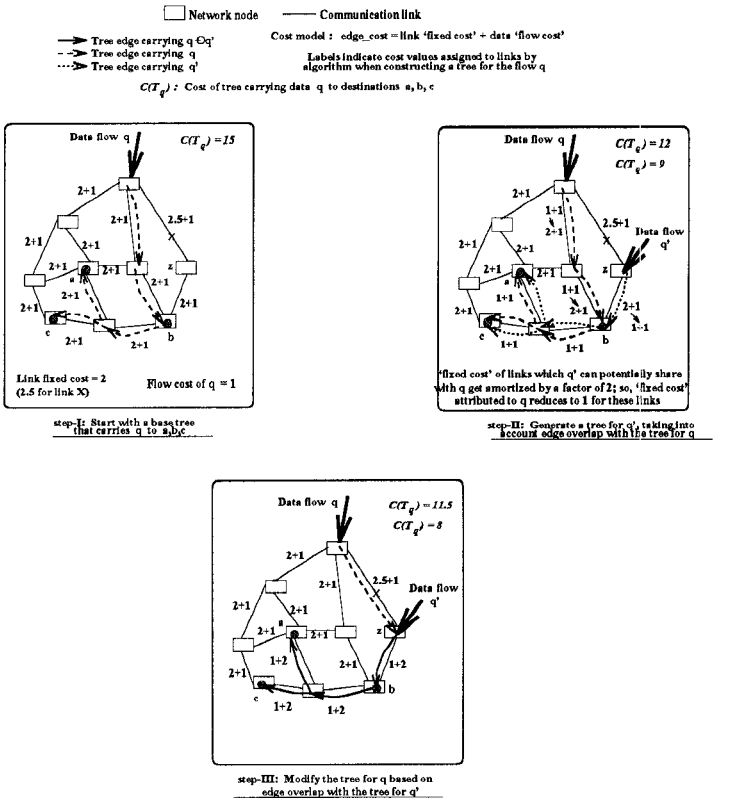

Figure 5: Tree setup scenario, with changing link costs

(now q is routed through link X, which was considered as incurring more cost in step-I and step-II).

4.4 Algorithmic complexity

A generalization of the above procedure for N data flows (where $N \geq 2$) is embodied into $\mathcal{A}$, taking into account the possible cost reductions due to various levels of edge overlapping among the component trees for $\{q_1, q_2, \cdots, q_N\}$.

As can be seen, $\mathcal{A}$ may employ $\mathcal{A}'$ as a building block, augmented with the functionality of accommodating link cost changes arising from path overlaps:

$$\mathcal{A}(\oplus\{q_1, q_2, \cdots, q_k\}) \ \equiv \ [\mathcal{A}(\oplus\{q_1, \cdots, q_{k-1}\}) \ \underline{\textbf{PATH\_OVERLAP}} \ \mathcal{A}'(q_k)] \quad \text{for } 2 \leq k \leq N;$$
$$\mathcal{A}(\oplus\{q_1\}) \ \equiv \ \mathcal{A}'(q_1).$$

$\mathcal{A}$ iteratively sifts through various possible levels of 'path overlapping', computes the link cost assignment for each level of 'overlap', and compares the NVT cost with that generated earlier.

The k^{th} step of the OVT algorithm, namely

$\mathcal{A}(\oplus\{q_1, q_2, \cdots, q_k\})$, causes $3 \times \sum_{i=1}^{k} k_{C_i}$ tree creations. Therefore, the total number of tree creations during the entire run is $\sum_{k=2}^{N}(3 \times \sum_{i=1}^{k} k_{C_i})$. Thus the complexity of $\mathcal{A}$ is $\mathcal{O}(Q^N)$ steps, where $Q > 1$ (the actual value of Q depends on $\mathcal{V}$, $\mathcal{E}$, $\mathcal{U}_s$ and $\mathcal{U}_d$. As can be seen, the number of NVT creations in $\mathcal{A}$ is exponential over N, with each NVT creation step being of polynomial complexity over $|\mathcal{E}|$ if the underlying 'steiner tree' algorithm employs heuristics and of non-polynomial complexity over $|\mathcal{E}|$ otherwise.

Summary

Overall, finding a OVT is related to the 'steiner tree' problem, but is complicated due to changing link costs. Since the original ST problem is itself NP-complete, we need heuristics that addresses both the problems together by considering the topological parameters $(\mathcal{U}_s, \mathcal{U}_d, \mathcal{V}, \mathcal{E})$ and the cost assignment parameters (c, f).

5 Heuristics for OVT construction

The goal of this paper is more towards formulating an extended form of the 'steiner tree' problem than coming up with OVT algorithms with the least complexity. From this perspective, our immediate interest is in presenting simple heuristics (even if they are somewhat conservative) that can bring out the benefits of our extended formulation of the problem.

5.1 Problem-reduction to standard 'steiner trees'

Basically, we transform the problem to one of generating standard 'steiner trees', and then apply the currently available ST algorithms to solve this reduced version of the problem. There are two reduction methods we apply to the OVT problem, as described below (let $\mathcal{U}_s = s_1, \cdots, s_N$ and $\mathcal{U}_d = d_1, \cdots, d_M$).

Equal edge cost assignment (ECA)

Here, we assume the maximum sharing on each link. The edge costs are set as $c(f(\oplus\{q_1, \cdots, q_N\}))_l \ \forall l \in \mathcal{E}$, the rationale being that each link can potentially have all the tree edges overlapping. This is an approximation, given that only a subset of the trees may actually overlap their edges over a given link. With the above cost assignment, a tree is created for the configuration set $\{s_1, s_2, \ldots, s_N, d_1, \ldots, d_M\}$.

Fixed merge point positioning (FMP)

Here, we assume the maximum sharing after a designated merge point node and no sharing before this merge point. An OVT is rooted at a fixed center node Y to connect all destinations $d_1, \ldots, d_M$. Distinct configuration sets are identified: $\{s_i, Y\}_{i=1,2,\ldots,N}$ and $\{Y, d_1, \ldots, d_M\}$. Separate NVTs are created for each $\{s_i, Y\}$ under a link cost assignment of $c(f(q_i)_l)|_{\forall l \in \mathcal{E}}$ and a OVT is created for $\{Y, d_1, \ldots, d_M\}$ under a link cost assignment of $c(f(\oplus\{q_1, \cdots, q_N\})_l)|_{\forall l \in \mathcal{E}}$. The 'center'-rooted tree is an approximation, given that it is possible for non-overlapping edges to cross-over each other at nodes other than Y, but cannot overlap their edges until reaching Y.

Having reduced the OVT problem to a 'steiner tree' problem, we now need to apply heuristics-based algorithms. Given a 'steiner tree' heuristics H, the ECA reduction allows applying H on a source-destination topological configuration with equal edge costs across all links $\in \mathcal{E}$ based on a maximum estimate of the edge overlap, whereas the FMP reduction allows applying H to distinct configuration sets and superposing the resulting paths.

5.2 Algorithmic procedures

In the second step, two types of steiner tree heuristics are studied: 'MST-type' that is based on constructing 'minimum spanning trees' on a global configuration (GLO) and 'construction-type' that is based on incremental addition of sources and destinations to a current configuration (INC) [5, 6]. The GLO heuristic employs Kruskal's algorithm on a fully-connected logical topology $\mathcal{L}(\mathcal{U}, \mathcal{E}')$, derived from $\mathcal{P}(\mathcal{V}, \mathcal{E})$ by assigning the cost of each link $e \in \mathcal{E}'$ as a sum of the costs of component links in $\mathcal{E}$ that make up e. The INC procedure sets up a path to connect a joining user u to a tree $\mathcal{T}(\widehat{\mathcal{U}}, \widehat{\mathcal{E}})$ over the topology $\mathcal{P}(\mathcal{V} - \widehat{\mathcal{U}}, \ \mathcal{E} - \widehat{\mathcal{E}})$, with link costs assigned based on the data rate of u and/or the data rates of the streams currently flowing in $\mathcal{T}$, depending on whether u is a source and/or destination respectively.

We are interested in determining the algorithmic complexity involved in the above process, viz., the number of time units spent and/or the number of interaction steps invoked between nodes during algorithm execution. The actual complexity may depend on the choice of heuristics in the algorithm. However, for any given heuristic $\mathcal{H}$, determining the additional complexity that arises relative to a basic ST algorithm incorporating $\mathcal{H}$ is our goal in this paper. Towards this goal, we employ the GLO and INC setup procedures as building blocks for realizing OVT algorithms. See Figure 6 to illustrate context of GLO and INC procedures in OVT setup algorithms.

Though the reduction methods proposed are somewhat conservative, they still achieve overall cost reduction in many cases (as shown in our simulation study later).

6 Simulation study

We have conducted extensive simulations of the GLO and INC algorithms on various physical topologies. Re-

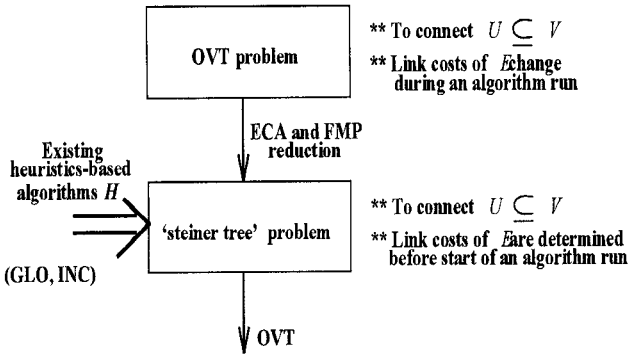

Figure 6: Our problem reduction approach for algorithmic construction of OVTs

sults of the simulation studies are discussed in this section.

6.1 Simulation procedures

We use 'random graph' technique to generate a physical topology involving a large number of nodes (25-100 nodes) and containing a source-destination configuration. Simulation parameters are the number of nodes in physical topology and the placement of sources and destinations. The cost functions we assumed are: $c(f(q_i)) = (m+1)$ for each q_i and $c(f(\oplus\{q_1, \cdots, q_k\})) = m + k.1$, where m depicts a fixed cost.

The sequence of steps in the simulation procedure can be briefed as below:

1. Generate the physical topology of network along with the fixed cost of links;
2. Designate the placement of source entities (in distinct nodes) and their per-hop flow costs;
3. Select a source-destination configuration involving a given number of destination entities (placed in distinct nodes) and the source entities resulting from step 2.
4. Do the following steps for the selected source-destination configuration:
   - Run GLO algorithm for each of NVTs, OVT(ECA) and OVT(FMP);
   - Compute the costs of multicast data paths set up by GLO algorithm;
   - Run INC algorithm for each of NVT, OVT(ECA) and OVT(FMP);
     - Start with one destination entity in the given configuration;
     - Run the algorithm for the joining of each additional destination entity;
   - Compute the costs of multicast data paths set up by INC algorithm;
5. Repeat steps 4-5 for different possible source-destination configurations.

6.2 Simulation results

The simulation program is run for large topologies containing 25 nodes. See Figure 7 for the simulation results. The simulated physical topology has 25 nodes, and the application configuration has 5 sources.

The number of destinations is varied between 1 and 15. We examined 300 distinct placements of sources and destinations in the physical topology (20 different placements for a given number of destinations). We average the costs for each number of destinations across the various placements. From the plots of cost versus number of destinations, we observe that as the number of destinations increases, the costs incurred also increase (this is because the number of hops in data paths increases).

The marginal increase in cost of OVT_{ECA} over NVT with respect to topology size may be explained as follows. For a given placement of destinations in physical topology, let $ch_{s,N}$ be the average size of tree across all possible placements of sources $s_1, s_2, \ldots, s_N|_{N \geq 2}$. The variation of $ch_{s,N}$ with respect to N may be given as:

- $\mathcal{O}(N^{c_1})$ for NVTs, where $0 < c_1 \leq 1.0$, depicting that each NVT has at least one edge, and, in the extreme, all NVTs have the same number of edges;
- $\mathcal{O}((\frac{N}{R})^{c_2})$ in OVT_{ECA}, where R is the network-wide average number of edges overlapping on a link in the OVT ($1 \leq R < N$), and $c_1 \leq c_2 \leq 1.0$, depicting possible longer paths to destinations than with separate NVTs;
- $\mathcal{O}((\frac{N}{R'})^{c'_2})$ in OVT_{FMP}, where $(1 \leq R' < R < N)$, and $c_1 \leq c'_2 \leq 1.0$.

Thus, the ratio of the number of paths to be created to the number of alternate paths that can be explored is often lower for OVTs than for NVTs. So the GLO/INC algorithm is more likely to pick up a 'good' OVT. Intuitively, as the physical topology size increases, the possible number of alternate paths to be explored becomes higher, and hence it is more likely for GLO/INC algorithm to find lower cost OVT paths than NVT paths.

Overall, a main goal of this paper is to expose the algorithmic issues in constructing OVTs, with specific relationship to their computational complexity. The explanation of simulation results in terms of algorithm-specifics should be viewed in this context.

7 Conclusions

The multipoint routing problem has a new dimension of complexity in the context of multimedia application environments (such as teleconferencing and interactive virtual class rooms). The application types focused in this paper consist of multiple media streams flowing through tree-structured paths to a set of destinations. These trees may overlap many of their edges on the intervening links,

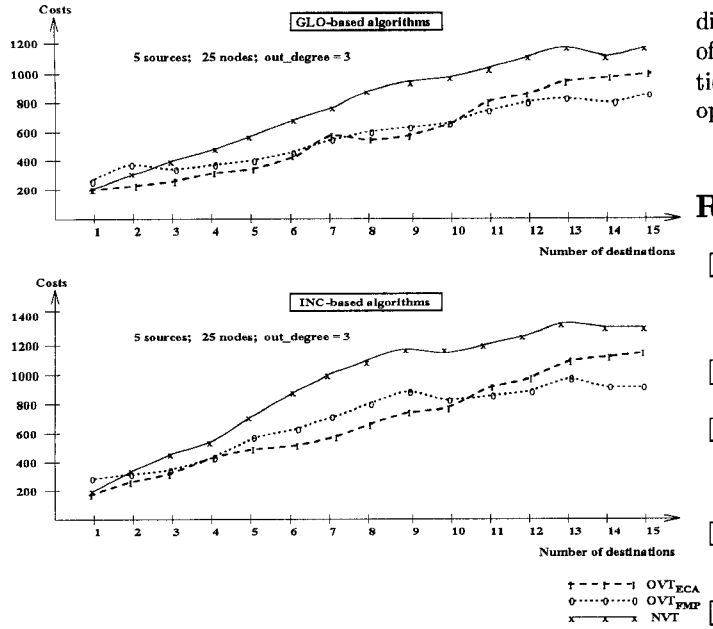

Figure 7: Relative performance of GLO and INC algorithms with NVT, ECA and FMP methods

depicting a sharing of the underlying link resources (such as bandwidth). Since resource sharing impacts link costs, tree construction algorithms need to take into account the cost implications of 'edge overlaps' while attempting to minimize the global cost of trees. This becomes necessary to support multimedia application environments (such as video distribution over 'Mbone').

Since constructing multicast trees, viz., the 'steiner tree' problem, is NP-complete, many heuristics based algorithms have been proposed that generate near-optimal steiner trees. An algorithm often assigns the edge costs first, and then examines various candidate paths for interconnecting a given set of user entities, in order to choose a path with the minimum cost. The 'edge overlapping', as an additional factor, now presents a new dimension to the complexity of 'tree generation' problems and the associated heuristics-based algorithms. The paper provided a walk-through of these problems and offered solutions.

We first formulated a model of link cost assignment for use by multicast routing algorithms. Using the model, we embarked on a study of the heuristics-based algorithms to tackle the 'modified steiner tree' problem. The premise is that these algorithms allow more cost-efficient routing. Basically, we employed reduction methods to transform the problem to one of generating standard 'steiner trees', and then applied the currently available 'steiner tree' algorithms to solve this reduced version of the problem. We also presented a simulation study of these solutions.

It should be noted that the proposed solutions provide a first level exposure to the algorithmic problem of multicast tree constructions with overlapping paths to carry different media streams. A concrete solution in the form of integrating flow & QOS based routing in current multicast protocols (such as CBT and PIM) is a larger and open research issue.

References

[1] S. E. Deering and D. R. Cheriton. **Multicast Routing in Datagram Internetworks and Extended LANs**. In *ACM Transactions on Computer Systems*, Vol.8, No.2, pp.85-110, May 1990.

[2] P. Winter. **Steiner Problem in Networks: A Survey**. In *Networks*, vol. 17, pp.129-167, 1987.

[3] M. R. Garey, L. R. Graham, D. S. Johnson. **The Complexity of Computing Steiner Minimal Trees**. In *SIAM Journal of Applied Math.*, vol. 32, pp.835-859, 1977.

[4] E. N. Gilbert and H. O. Pollack. **Steiner Minimal Trees**. In *SIAM Journal of Applied Math.*, vol. 16, pp.1-29, 1968.

[5] B. M. Waxman. **Routing of Multipoint Connections**, In *IEEE Journal on Selected Areas in Communications*, Vol.SAC-6, No.9, pp.1617-1622, Dec. 1988.

[6] X. Jiang. **Routing Broadband Multicast Streams**. In *Computer Communications*, Butterworth-Heinemann, vol.15, no.1, January, 1992.

[7] D. Bertsekas and R. Gallager. **Routing in Data Networks**. Chapter 5 in *Data Networks*, Prentice Hall Publ. Co., 1992.

[8] G. N. Rouskas and I. Baldine. **Multicast Routing with End-to-End Delay and Delay Variation Constraints**. In *IEEE Journal on Selected Areas in Communications*, vol.15, no.3, April 1997.

[9] J. Kadirire. **Minimizing Packet Copies in Multicasting by Exploiting Geographic Spread**. In *Computer Communication Review*, ACM SIGCOMM, vol.24, no.3, pp.47-62, July 1994.

[10] V. P. Kompella, J. C. Pasquale and G. C.Polyzos **Multicasting for Multimedia Applications**. In proc. *Computer Communications*, INFOCOM'92, Italy, 1992.

[11] D. C. Verma and P. M. Gopal. **Routing Reserved Bandwidth Multi-point Connections**. In Proc. *Communication Architectures, Protocols and Applications*, ACM SIGCOMM, pp.96-105, Sept. 1993.

[12] N. Maxemchuk. **Video Distribution on Multicast Networks**. In *IEEE Journal on Selected Areas in Communications*, vol.15, pp.357-372, April 1997.

[13] T. Ballardie, P. Francis and J. Crowcroft. **Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing**. In Proc. *Comm. Architectures, Protocols and Applications*, ACM SIGCOMM, pp.85-95, Sept. 1993.

[14] S. Deering, D. Estrin, D. Farinacci and V. Jacobson. **An Architecture for Wide-Area Multicast Routing**. In Proc. *Comm. Architectures, Protocols and Applications*, ACM SIGCOMM, 1994.

[15] K. Ravindran. **Architectures and Protocols for Data Multicasting in Multi-service Networks**. In *Computer Communications Review*, ACM SIGCOMM, July 1996.