Video Multicasting with Hierarchical Encoding: The Fixed and the General problem

Michalis Faloutsos Rajesh Pankaj Kenneth C. Sevcik

October 6, 1997

Abstract

The increasing demand for point-to-multipoint connections has established multicast routing as a crucial aspect of network management. Hierarchical encoding is a newly proposed technique that offers a variety of quality levels in one encoding; a user or a switch can filter the data that will be processed further. In this paper, we examine how hierarchical encoding can increase the efficiency of multicast routing and propose polynomial algorithms to this effect. Theoretical and experimental work demonstrates the advantage of the proposed algorithms over previous solutions.

1 Introduction

Multicasting can be defined as the distribution of the same information stream from one to many nodes concurrently. A lot of emerging applications are of multicast nature, and thus multicasting has become an important research area. The main idea is that instead of multiple point-to-point connections it is more efficient to distribute the information on a tree. Efficiency is directly related to network resources, whose use we consider as the cost of routing. Naturally, the goal is to minimize the cost of the multicast tree.

The multicast problem, also known as the Steiner tree problem, is NP-complete. It has been studied extensively and algorithms for the exact solution have computational complexity exponential to either the number of destinations [DW72] or to the number of nodes in the network or both [Win87]. Hence, researchers focused on approximation algorithms for the Steiner problem that run in polynomial time. The result is a number of algorithms whose solution is never worse than two times the optimal one [TM80] [KMB81] [RS83] [Meh88]. However, the increase of the application requirements has introduced several constraints and create variations of the multicast problem. User synchronization and round-loop requirements introduce end-to-end delay constraints that have been addressed by Kompella et al [KPP93] [KPP92] and by Zhu et al. [ZPGLA95]. Physical limitations of the switches bring forward degree constraints, an issue that has been studied by Bauer and Verma [BV95]. The dynamic user-participation requirement introduces the dynamic problem that has been studied by Doar and Leslie [DL93] and Waxman et al. [Wax88] [Wax93] [IW91]. Removing the assumption of symmetric graphs, Ramanathan [Ram96] and Faloutsos et al. [FPS96] study the effect of the asymmetry in the multicast problem.

The hierarchical encoding technique was proposed recently for efficient resource use. This encoding method has received a lot of attention from the networking community [Gha89] [KV89] and is still an active research area. Hierarchical encoding is a layered way of encoding the kind of information that can appear in different quality levels such as audio and video data. In hierarchical encoding, the information of a session is divided into an enumerated set of **levels**, and higher levels can only be used on top of the lower levels. Therefore, a destination can receive a continuous sequence of such levels, including the first level. We refer to such a set of levels by the term **stream**. Note that each stream corresponds to a different quality of the data, with the lower stream being of lower quality. As an example, we can think of a video data that is encoded into two levels: level l_1 is the color information, and level l_2 is the high definition information. Correspondingly, we have the two streams: s_1 which consists of l_1 and s_2 which consists of $l_1 + l_2$. A destination that wants the high definition image must receive stream s_2, in other words both levels.

In this paper, we propose ways to exploit the hierarchical encoding scheme in routing multicast connections efficiently. The ability to offer a variety of streams constitutes the **heterogeneous multicast problem**. In contrast, we refer to the one-stream problem as the homogeneous problem. To solidify the notion of efficiency, we identify several issues that efficient heterogeneous algorithms must take into account. We will use these issues as guidelines in our approach to the problem.

1. *Cost sensitivity.* We want to extend the existing algorithms to consider the different resource requirements of the streams, given that higher streams use more resources. Conceptually, we want the routes to gravitate towards the high stream nodes. Furthermore, we want the new algorithms to behave as efficient homogeneous algorithms for one stream.

2. *Stream rewarding.* As it will become apparent later, higher streams should offer a higher reward to the service provider. In absence of such a policy, the service provider will offer only the lowest stream.

3. *Low complexity.* The need for low complexity algorithms is introduced by the response time constraints (users are impatient!) and scalability issues. Ideally, we want to maintain the complexity of the homogeneous algorithms.

4. *Flexibility.* The heterogeneous algorithms should be able to incorporate additional constraints with minor changes. As we briefly mentioned, there are variations of the multicast problem, where we want to be able to use hierarchical encoding as well.

The first contribution of this paper is a number of polynomial algorithms that meet the above requirements. We study the worst-case performance of our algorithms and demonstrate the improvement over previous solutions. Using simulations, we examine the advantage of our algorithms on various problem instances. The second contribution of the paper is the study of the heterogeneous problem in its generality, that to the best of our knowledge is conducted for the first time. Furthermore, series of simulations demonstrate the gain from a heterogeneous approach.

Finally, we want to stress the tight relations of the heterogeneous problem with real applications. First, heterogeneity is in accordance with the broadband network concept, which is based on the notion of user-defined quality of service. In the heterogeneous framework, the provider is able to negotiate the quality with every user separately. Second, a lot of multicast applications can use hierarchical encoding. Apart from video-on-demand, we can mention teleconferencing, computer supported collaborative work (CSCW), and electronic newspapers. Finally, heterogeneity may even be compulsory in some cases. The great variety of machines and technologies that interoperate require the flexibility of hierarchical encoding. For example, mobile users have considerably different requirements and hardware support than an office workspace in a possibly joint teleconference.

The rest of this paper is structured as follows. In section 2, we present our model and the related background work. In section 3, we focus on a special case of the heterogeneous problem; every destination requires exactly one stream. Although this is a simpler case, it can reflect some practical situations. In section 4, we address the general heterogeneous problem where destinations can receive any stream within a range of streams. This stream selection is an additional optimization problem, which has to be addressed in parallel with the routing problem. In section 5, we present our experimental results that indicate the benefit from both the heterogeneous approach and the proposed algorithms.

2 Model Definition and Related Work

First, we give an overview of the commonly used model of the problem. Given an undirected weighted graph $G(V, E)$ and a subset of **participant** nodes $S \subseteq V$, we want to find the minimum weight tree that spans the participants. The number of nodes $|V|$ is N and the number of participants $|S|$ is M. For the homogeneous case, the routing cost on an edge is its weight, in other words the cost is the distance. However, as we will see in the heterogeneous problem, the routing cost becomes a function of the required resources. The cost of a path is the sum of the routing cost of its edges. In this paper, when we refer to the path between two nodes, we imply the minimum cost or shortest path. Along these lines, the term "nearest" refers to the minimum cost path.

As previously mentioned, research has focused on approximation algorithms, which creates the need for comparison. The most common measure of the quality of an approximation is the **worst-case inefficiency (WCI)** of a multicast algorithm which is defined as the largest ratio of the cost of the solution of the algorithm over the optimal one for any problem instance. As we already mentioned, several greedy algorithms have a WCI of 2. In this paper, we will focus on a greedy algorithm studied by Takahashi and Matsuyama [TM80], which we will call the **Nearest Participant First (NPF)** algorithm. An overview of NPF is presented in the next paragraph. The union of the shortest paths between all source and destination paths also forms a solution with a WCI of $O(M)$. We call this algorithm the **Shortest Paths** algorithm. Note that every destination is routed separately obliviously of the other destinations. A comprehensive presentation of algorithms for the homogeneous problem can be found in a survey by Winter [Win87].

In the NPF algorithm, we start from the root and construct the multicast tree by adding one participant at a time.

Nearest Participant First Algorithm (NPF)

1. START from the source.
2. REPEAT until all destinations are reached
   (a) SELECT the nearest unreached destination d_{new} to the partial tree.
   (b) JOIN d_{new} with its shortest path to the tree.

An efficient implementation of NPF has polynomial complexity bounded by $O(M\ N^2)$ [Win87].

We can distinguish two phases; the shortest path phase (which dominates) and the tree creation phase (REPEAT loop). First, we need the shortest paths between each participants and all other nodes. If we use Dijkstra's algorithm $O(N^2)$ as described in most graph theory textbook, the complexity of this phase becomes $O(M\ N^2)$. Second, we can think of the following way to implement the tree creation phase. For every participant, we keep track of its shortest path to the tree so far and the corresponding connection point. We call this information **joining information**. Every time a node is added to the tree (as part of a path), we update the joining information of each unreached participant. Namely, we examine whether the participant is closer to the added node. During this procedure we can also keep track of the globally shortest path, and thus, the SELECT step becomes trivial. Given that we can add N nodes at most, and thus have N updates for each of the $M-1$ destinations, the total complexity of the REPEAT loop is $O(M\ N)$. We will use this analysis in the calculation of the complexity of the heterogeneous algorithms.

Takahashi and Matsuyama [TM80] study the efficiency of NPF and prove that its WCI is bounded by $2(1-\frac{1}{M})$. Surprisingly, this algorithm has received little attention compared to the well known KMB algorithm [KMB81]. Recently, Ramanathan [Ram96] suggests that KMB can be considered as a special case of NPF. Conceptually, KMB corresponds to an NPF algorithm where the unreached destinations consider joining only to participants in the multicast tree in steps 2.a and 2.b of the algorithm (as opposed to joining the partial tree anywhere).

We can consider another greedy algorithm similar to NPF that selects the next destination at each step randomly. Namely, we start from the root, and at each step, we pick a random destination and connect it with the nearest node of the tree. We call this algorithm the **Random Participant First (RPF)** algorithm, and we use the term **random order** to denote the order in which the participants are selected. RPF corresponds to an extension of NPF to an on-line problem in which new participants join after the tree has been set up. Imase and Waxman [IW91] prove that the worst-case inefficiency of RPF is bounded

$$WCI_{\mathrm{RPF}} \leq \log(M)$$

The introduction of randomness may seem peculiar at this point, but we will use the above bound of RPF to prove bounds for the WCI of the algorithms we propose.

2.1 The Heterogeneous Problem

The heterogeneous problem has received very little attention. Shacham [Sha92b] [Sha92a] considers heterogeneity, but he examines only a special case of the heterogeneous problem. All destinations receive the highest stream of the session that the network congestion can allow. This formulation of the problem is called the **best-effort problem** and is a special case of the heterogeneous problem. The proposed algorithm is an extension of the Shortest Paths algorithm, and thus its WCI is at least that of the homogeneous case, which is $O(M)$. The complexity of the algorithm is polynomial and bounded by the complexity of a shortest paths algorithm ($O(N^2)$). The best-effort problem neglects two issues. First, some destinations may not want to receive the highest stream. In our video data example, a user with a low quality television set can not exploit the high definition level. Second, efficient use of the network may require optimization of the offered streams. For example, offering the highest stream to a distant destination may be possible but not beneficial, because the routing cost may be higher than the gain, i.e., the price the user is willing to pay. In a more recent work, Shacham and Meditch [SM94] examine the problem of routing unrelated streams to destinations over a given tree. Although they address a significantly different problem, their ideas inspired our work.

At this point, we can present our assumptions. The main assumption is that all the levels of a session have to use the same route. If this is not necessary, we can multicast each level independently only to the appropriate destinations. However, this independence requires synchronization of the levels, which is not trivial. The difficulty stems from the differences in the delay and the delay jitter of levels that follow different paths. Delay in one of the levels demands the buffering of the other levels, and we need more buffer space to avoid loss of information. We will also need protocols and processing power to use the data of the levels correctly. On the other hand, routing the levels together can facilitate the network management through the enforcement of priority policies. For example, giving higher priority to the lower and fundamental levels, we can shift the losses to the higher levels, and thus achieve graceful degradation of the quality.

We also assume that our network is an undirected graph. Although, our algorithms can be extended to the directed case in a straightforward way, their performance can differ greatly. Even for the homogeneous case, in highly asymmetric graphs the WCI of the greedy algorithms is $O(M)$ [FPS96]. Namely, multicasting can be as bad as multiple point-to-point routing in the worst case. Since the homogeneous problem is a special case of the heterogeneous problem, the general bound is at least as bad. The effect of the asymmetry of a graph in the heterogeneous problem is an issue that we have started studying, but it could not be included here due to space limitations.

3 The Fixed Heterogeneous Problem

In a heterogeneous session, destinations can define a range of streams, with which they are willing to participate in the session. Each stream in this range is a **preferred** stream for that destination. When each destination defines only one preferred stream, we have a special case of the heterogeneous problem that we call the **fixed heterogeneous problem**. In this section, we study the fixed problem. The problem can be stated as follows: given an undirected graph, a source and a set of destinations with one preferred stream each, we want to identify the minimum cost tree from the source that offers the preferred stream to each destination.

At this point, we have to redefine the cost on an edge to accommodate the cost sensitivity requirement that we mentioned earlier. The routing cost of an edge is defined as a function of the routing requirements of the connection. For the rest of this paper, we will use the concept of flow (in terms of bandwidth) to represent routing requirements. Clearly the use of one parameter to express these requirements is not trivial, but the task of relating resources with utilization is a significant issue by itself and extends beyond the scope of this paper. We define the **cost function** $cost(e, f)$, which represents the cost of passing flow f on edge e. We define the cost of an edge e on which we want to route a connection of bandwidth b as: $cost(e, f+b) - cost(e, f)$, where f is the existing flow. We define the cost of a path/session as the sum of all the cost increases of all edges before and after routing the required bandwidth, i.e., $\sum_{e \in E} cost(e, f'_e) - cost(e, f_e)$, where f_e and f'_e are the flow before and after routing the session. Given a partial tree, the cost of connecting a destination with a stream is called **joining cost**, and depends on the destination, the stream and the partial tree of that moment. Note that the actual mathematical function can be tailored to reflect conditions and assumptions of the problem. A commonly-used cost function is the *usage*

$$cost(e, f) = \begin{cases} \alpha + \beta\ f, & \text{if } f < Cap \text{ and } a, b \in \mathcal{R} \\ \infty, & \text{otherwise} \end{cases}$$

Another function is the *exponential*

$$cost(e, f) = e^f$$

This function lends itself to theoretical study, mainly because of its derivation properties. Other functions can be used, but we consider the two previous functions.

Our first algorithm for the heterogeneous problem is based on the observation that lower stream destinations can tap into the higher streams and acquire the levels they want. Therefore, we route the higher stream destinations first. Among destinations of the higher stream, we choose the one nearest to the partial multicast tree. This strategy gives rise to the **Maximum Stream Participant First** or **MaxStrPF** algorithm.

Maximum Stream Participant First Algorithm (MaxStrPF)

1. START from the source.
2. REPEAT until all destinations are reached
   (a) SELECT destination: Find the set of unreached destinations that have maximum preferred stream. From this set, select the nearest (minimum cost) destination to the partial multicast tree, d.
   (b) JOIN: Include d and the related path in the tree.

In another approach, we can change the selection criterion and choose the destination that can be added with minimum cost at each step. Observe that low stream nodes have an advantage over the high stream ones, since higher streams are more costly. We call this algorithm **Minimum Cost Participant First** or **MinCostPF**. A more detailed description of the algorithm can be found in appendix C.

Note that, when a destination intends to join the tree with stream s at a node that carries a lower stream, the joining cost includes the cost of bringing the required levels to the connection point. We call this extra cost the **overrouting cost**.

In implementing both algorithms, for every node of the multicast tree, we can keep track of the overrouting cost for each streams. Namely, whenever an existing node in the tree increases its existing stream the overrouting cost of all its descendants (in the tree) is modified. For each tree node, the updates are limited to $\min(M, L)$. since each update is related with a stream increase, but also such stream increases happen when a new destination is added in the tree. Each update modifies the overrouting cost of each stream. Thus, the complexity for this phase is bounded by $O(L\ N\ \min(M, L))$.

Efficient implementations of both algorithms have complexity equal to that of the homogeneous NPF algorithm, that is $O(M\ N^2 + L\ N\ \min(M, L))$. If we restrict the number of streams to $O(N)$, the complexity of the algorithms becomes $O(M\ N^2)$. This satisfies the *low complexity* requirement. The *cost sensitivity* requirement is met by the way we calculate the cost. Finally, the *flexibility* requirement is also satisfied since it follows the framework of NPF. Note that the *stream rewarding* requirement is not applicable in the fixed problem. Note that the inefficiency of both algorithms is bounded by 2 for the cases where the problem approaches the homogeneous case. If we assume a linear relationship between cost and bandwidth, the following theorem holds for MaxStrPF.

Theorem 1 *Assume that the cost of an edge is linear to its flow. In the fixed heterogeneous problem with M participants, the WCI of MaxStrPF is bounded by*

$$WCI_{MaxStrPF} \leq \log(M)$$

The proof can be found in APPENDIX A. We are currently examining the performance of MaxStrPF when the cost is an exponential function of flow and the performance of MinCostPF for both cost models.

4 The General Heterogeneous Problem

In the previous section, we restricted each node to one preferred stream. Relaxing this constraint, we arrive at the **general** (or simply) **heterogeneous problem**. Each destination defines a range of preferred streams and participates in the session with one of them, while streams below the preferred range are called **failing streams**. Given the preferred ranges, the network manager has to decide which stream to offer to each destination. In other words, in the heterogeneous problem, the stream selection problem is added to the routing problem. Note that the two problems are correlated and can not be solved independently.

We can now recall the stream rewarding requirement we mentioned in the introduction. With the current model, the network manager will offer the lowest preferred stream to each destination minimizing the total cost. This approach does not consider the gain of offering a high stream. For this, we introduce the concept of **happiness** that measures the user-satisfaction that corresponds to a certain stream. Alternatively, user-satisfaction can be translated to a price that the user is willing to pay for a certain service. For each destination d, a happiness value $hap_d(s)$ is associated with each stream s, and these values are non-decreasing with stream. Furthermore, for different destinations the happiness values of a stream may not be the same. For uniformity, we will assume that failing streams have a happiness value of minus infinity, and thus, every destination-stream pair has a happiness value. Of course, the total happiness of a session is the sum of the happiness values of all destinations.

Given the above formulation, the general heterogeneous problem becomes an optimization problem with two competing parameters. On the one hand, we want to minimize the total cost C, and on the other hand, we want to maximize the total happiness H. Given two different solutions, we need to be able to compare them. For this, we introduce the $HapCost(H, C)$ function that combines the two parameters into a single objective value. By definition, the function measures the efficiency of the solution, therefore, large happiness and low cost correspond to large values of the function.

Given the new model, the heterogeneous problem can be defined as follows. The input is an undirected graph, a source, a set of destinations with happiness values for every stream, and an objective function $HapCost()$. We want to find the tree and the stream assignment whose happiness and cost maximize the objective function.

As a comparison metric for algorithms, we define the inefficiency of an algorithm as the ratio of the optimal solution $HapCost()$, OPT, over that of the algorithm, ALG: OPT/ALG.

Let us focus on the stream assignment problem, and consider only one destination. Trivially, the route is the shortest path between the source and the destination. All we have to do is choose one of the preferred streams. We choose the stream that maximizes the objective function, and call it **optimized stream**. More rigorously, the optimized stream is given by $\max_s [HapCost(hap_d(s), c)]$ where c is the joining cost for d with stream s.

We will incorporate the above procedure in the MaxStrPF algorithm of the previous section. Namely, at each selection step, we find the optimized stream for each unreached destination. Then, given these streams, we choose a destination as in the fixed problem. Recalling MaxStrPF, we can choose the destination with the maximum optimized stream. We call this algorithm the **Optimized Stream Maximum Participant First** or **OptMaxStrPF** and we present it below. It is easy to see that OptMaxStrPF behaves like MaxStrPF for a fixed heterogeneous problem.

Optimized Stream Maximum Participant First Algorithm (OptMaxStrPF)

1. START from the source.
2. REPEAT until all destinations are reached
   (a) SELECT stream: For each destination d, find the stream $s_{o,d}$ that maximizes the objective function
   $$s_{o,d} = \max_{s} [HapCost(hap_d(s), c)]$$
   where c is the joining cost for d with stream s
   (b) SELECT destination: Find the set of unreached destinations that have maximum optimized stream. From this set, select the the destination, d, that is nearest to the partial multicast tree.
   (c) JOIN: Include d and the related path in the tree with $s_{o,d}$.

In another approach, we observe that, at each step, we have to select a destination-stream pair. Therefore, we choose the pair whose added happiness and cost maximize the objective function. We call this algorithm **Optimized Participant First** or **OPF**.

Optimized Participant First Algorithm (OPF)

1. START from the source.
2. REPEAT until all destinations are reached
   (a) SELECT destination-stream: Find the (destination, preferred stream) pair whose added happiness and cost maximize the objective function:
   $$(d_o, s_o) = \max_{(d,s)} [HapCost(hap_d(s), c)]$$
   where c is the cost of joining d to the tree with stream s.
   (b) JOIN: Include d_o and the related path in the tree offering stream s_o.

In the implementation of both algorithms, every destination has to keep track of the minimum-cost path for every preferred stream. Note that the preferred streams of a node may have a different minimum-cost paths. Thus, for every node added in the tree, we have to re-examine the joining cost of each preferred stream for each destination.

In an efficient implementation of the above algorithms the complexity is polynomial and bounded by $O(M\ N^2 + M\ N\ L)$, where L is the number of distinct streams (see Appendix B).

As in the homogeneous case, we have to calculate the shortest paths between each participant and all other nodes, which as we already said costs $O(M\ N^2)$ assuming we use Dijkstra's algorithm, During the iterations, we update the joining information of each destination at most N times $O(M\ N)$, since the maximum number of nodes that can be added is N. For each such update, we calculate the minimum-cost path of $O(L)$ streams at most. Therefore, the total cost of this phase is $O(M\ N\ L)$. Note that the network manager can restrict the number of distinct streams, when the complexity becomes an issue. If we restrict the maximum number of streams L to $O(N)$, the complexity of the algorithms becomes equal to that of the homogeneous problem, which is $O(M\ N^2)$.

In conclusion, we can see that the two algorithms conform with the cost sensitivity, flexibility and low complexity requirements of the introduction. They also satisfy the stream rewarding requirement, since they encourage the distribution of high streams with the use of the happiness concept. To the best of our knowledge, these are the first algorithms that address the general heterogeneous problem.

5 Experimental Results

We implemented a simulator in C++ to examine the performance of the above algorithms in practice. We chose to experiment on the family of graphs that we call **random euclidean** graphs. They were chosen, first, because they are widely used in network simulations, and second, because these graphs seem to have some characteristics that we intuitively expect from real networks. Briefly, these graphs are created by uniformly distributing nodes on a euclidean plane. Edges are set between nodes on a probabilistic basis that favors nodes of small euclidean distance. A more detailed description of this graph generation can be found by Doar and Leslie [DL93] or Waxman [Wax88]. The graph we used was a 40-node random euclidean with 108 edges. The participants are chosen randomly, and so is the root of the session among them. Note also that when we study the effect of the number of the participants, we add participants to the existing session. This incremental change makes the graphs easier to interpret. We used the average of thirty-two different executions in an attempt to smoothen the irregularities of special cases. We assume that the information is divided into five distinct levels. All preferred ranges include the lowest stream, while the higher preferred stream is randomly chosen. The happiness values are increasing linearly with the stream index and they are identical for all destinations, because we thought that it would be easier to get an intuitive feeling of the dynamics of the problem. We consider the flow as a linear function of the stream index. We assume the cost of an edge to be exponential with its flow: $cost(f, edg) = e^{A\,f}$, for any edge edg. We use this function for compatibility with theoretical work in admission control and competitive analysis [Plo95] [GKR95]. The parameter A is a measure of the cost of routing, and we call it the **stream cost**. We can think of stream cost as measure of the cost of bandwidth.

For the objective function, we use the following:

$$HapCost_1(Hap, Cost) = \frac{Hap}{Cost}$$

$$HapCost_2(Hap, Cost) = a\ Hap - b\ Cost,\ \ a, b \in \Re$$

In the first function, we consider the network resources as our capital and the satisfaction as our revenue, then for every session we want to increase the "rate of return" on our capital. In the second function, we want to maximize the difference between the happiness and the cost. Note that a and b are real-number constants that adjust the arithmetic values of happiness and cost. Note that if we have statistical knowledge about future sessions, we can incorporate it in the above functions.

The algorithms we compare with are the following:

1. *NPF low:* We perform NPF and offer the minimum stream of the session to all destinations. This algorithm offers a very conservative solution, and we use it as a point of reference.

2. *NPF hi:* We perform NPF and, to every destination, we offer its maximum preferred stream.

3. *het SP:* We perform the Shortest Paths algorithm and offer the maximum preferred stream to destinations. Recall that het SP corresponds to the algorithm suggested by Shacham [Sha92b], which seems to be the only heterogeneous algorithm proposed so far.

4. *OPF:* We perform Optimized Stream Optimized Participant First as described in the previous section.

5. *OptMaxStrPF:* We perform Optimized Stream Maximum Stream Participant First as described in the previous section.

We will call het SP and NPF hi *best-effort algorithms*. Note, also, that NPF low and the best-effort algorithms are fixed stream problems. We will also call OPF and OptMaxStrPF *stream-optimizing* algorithms.

The gain of heterogeneous solutions. In Figure 2, for $HapCost_1(Hap, Cost)$, we plot the $HapCost()$ values of the solutions versus the stream cost parameter for 14 participants. In Figure 3, we have the same plot for the $HapCost_2(Hap, Cost)$. In both figures, we see that as the stream cost increases, the heterogeneous solution outperforms the fixed stream solutions. Furthermore, the best-effort algorithms can lead to arbitrarily inefficient solutions depending on the stream cost. In Figure 2, the return of accommodating a session is zero, which means that our investment in using network resources does not return a profit. In Figure 3, the inefficiency is depicted more dramatically as an exponentially increasing deficit between gain and cost. Notice that in both figures, OPF takes advantage of low stream cost values and offers high stream and thus performs better that NPF low. Meanwhile, when the stream cost increases, NPF offers lower streams, and converges to the solution of NPF low.

The general heterogeneous problem. Figures 4 shows the $HapCost()$ of the solution versus the number of participants. Note that for the two previous and the rest of the graphs we use the $HapCost_1(Hap, Cost)$ objective function. Figure 4 corresponds to a stream cost of 0.5. We can see that the stream-optimizing algorithms (OPF and MaxStrPF) outperform the NPF low and best-effort algorithms. In more detail, we can see that OPF performs better than MaxStrPF on average. In addition, NPF hi performs better than het SP: they offer the same happiness but the routing decisions of NPF hi are better. Notice that the performance of the best-effort algorithms degrades as the number of participants increases. This can be explained if we think that the more participants we have the more likely it is to have high stream destinations away from the source.

The fixed heterogeneous problem. Since the total happiness offered is the same for all solutions, we measure the cost. Note that all algorithms offer the same streams to each destination, except for the NPF low, which is shown for comparison. In the figures, we plot the relative cost of all algorithms is divided by the cost of het SP. In Figure 5, we plot the relative cost of the solution versus the number of participants for stream cost 0.01. Note that MinCostPF is as expensive as the NPF low, while MaxStrPF is costlier than these two. Finally, het SP is less effective compared to MinCostPF and MaxStrPF giving solutions of approximately 25 per cent worse than MinCostPF. We also wanted to examine the effect of the stream cost parameter on the fixed problem solutions. Figure 6 shows the relative cost of the algorithms for varying stream cost parameter for 14 participants. In all the cases, MinCostPF and MaxStrPF outperform het SP by as much as 25 and 17 per cent respectively for low stream cost values. It is worth noticing that for low stream cost values, MinCostPF performs better, but as the stream cost parameter increases the MaxStrPF prevails. This can be explained if we recall that the main idea of MaxStrPF considers first the high stream connections. Finally, note that the three algorithms seem to converge to a smaller relative distances as the stream cost increases. or high stream cost values, the high-stream destinations define the total cost.

Clearly, the above set of experiments is just an indication of the usefulness of our formulation and the relative efficiency of the proposed algorithms. However, there are several other parameters whose effect would be interesting to study. The main one is the topology of the graph. The session can also vary, in terms of the choice of happiness values and of the ranges of the preferred streams. However, the range for both these parameters is very large and actual values depend on the application. Therefore, for more exact simulations, experiments should be conducted on scenarios that reflect specific applications.

6 Conclusions

In this paper, we deal with the problem of merging the hierarchical encoding and the multicast routing. The hierarchical encoding scheme allows destinations to receive different subsets of the same information, when this information is of variable quality, such as audio and video. The goal of this paper is to propose efficient algorithms for this heterogeneous multicast problem. In this case, the term efficiency includes the notions of cost sensitivity, stream rewarding policy, low computational complexity, flexibility, and of course low worst case inefficiency.

First, we study the fixed heterogeneous problem. For this, we redefine the routing cost to make our decisions sensitive to the actual routed bandwidth, since this varies within the same session. Second, we formulate and study the general heterogeneous problem. The service provider has to select both the routes and the streams for each destination. The introduction of the notion of happiness rewards the distribution of higher streams (which clearly is costly to the service provider). This way the problem becomes a double optimization problem: we want to maximize the happiness and minimize the cost.

The proposed algorithms meet our requirements of efficiency and outperform existing heterogeneous or homogeneous solutions. For the fixed problem, we prove that MaxStrPF has WCI of $\log(M)$, while the previous heterogeneous algorithm [Sha92b] has a WCI of $\Theta(M)$. Experimental results indicate the advantage of the proposed algorithms over other solutions.

In conclusion, network management can benefit from the use of hierarchical encoding in routing. We indicate that hierarchical encoding can be used beneficially in multicast connections. We believe that these simple and efficient algorithms are a step forward to this direction.

ACKNOWLEDGMENTS. The authors would like to thank Manolis Skordalakis of National Technical University of Athens for facilitating the writing process of this paper. Special thanks to Christos Faloutsos for his keen comments on the structure of the paper.

A Appendix: The Proof of the Theorem for MaxStrPF

In the fixed stream heterogeneous problem, we prove that the WCI of MaxStrPF, WCI_{MaxStrPF} is bounded by $\log M$, where M is number of the participants. We do this by proving that WCI_{MaxStrPF} is bounded by the WCI of RPF.

Theorem 1 *Assume that the cost of an edge is linear to its flow. In the fixed heterogeneous problem with M participants, the WCI of MaxStrPF is bounded by*

$$WCI_{MaxStrPF} \leq \log(M)$$

PROOF. Let the levels be l_i and the streams s_i for $i = 0, ..., max$, such that s_i is the union of l_0 to l_i. Let us partition the destinations into sets D_i destinations that require stream s_i. Let us denote A_i the destinations that require a stream higher or equal to s_i, i.e., $A_i = \bigcup_{j=i}^{max} D_j$.

Let $OPT(W, b)$ be the cost of the optimal tree of the homogeneous problem for the set of nodes W receiving information of bandwidth b (that can be either stream or level).

For the optimal tree OPT_{het} we can see that if we route the levels from separate routes we can find a more efficient solution:

$$OPT(A_0, l_0) + OPT(A_1, l_1) + ... + OPT(A_{max}, l_{max}) \leq OPT_{het}$$

Having assumed that the cost is additive, the cost of routing on the same edge different levels is equal to the sum of the costs: $cost(e, u + w) = cost(e, u) + cost(e, w)$, for any edge e.

We choose the order of destinations for RPF to be the same order Q_{MxS} as MaxStrPFin the heterogeneous case. Let $MSPF_{het}$ be the cost of the tree of MaxStrPFLet also $RPF(W, b)$ be the cost of the homogeneous RPF for the set W of nodes bandwidth b.

Algorithm MaxStrPFcan be described as several executions of RPF choosing nodes in the Q_{MxS} order.

1. Start from stream $k = max$
2. Repeat until $k = 0$
   (a) Execute RPF for the D_k destinations offering stream s_k. Note that at each step, we use the previously routed higher streams.

The cost of the algorithm is described by the following formula:

$$MSPF_{het} = RPF(D_{max}, s_{max}) + ... + RPF(D_1, s_1) + RPF(D_0, s_0) \tag{1}$$

Note that, the routing decisions are independent of the bandwidth. Given that RPF follows the MaxStrPForder Q_{MxS}, we can redescribe MaxStrPFas follows:

1. Start from level $k = max$
2. Repeat until $k = 0$
   (a) Execute RPF for all the destinations A_k that use level l_k offering this level following the Q_{MxS} order

According to the above structure and the additive property, the cost of the algorithm is expressed as follows:

$$MSPF_{het} = RPF(A_{max}, l_{max}) + ... + RPF(A_1, l_1) + RPF(A_0, l_0)$$

We can substitute: $RPF() \leq \log|A_i|\, OPT() \leq \log M\, OPT()$ in the above equation. Then we compare the the resulting formula with (1):

$$MSPF_{het} \leq \log M\ [OPT(A_0, l_0) + OPT(A_{l_1}, l_1) + ... + OPT(A_{max}, l_{max})] \Rightarrow$$
$$\Rightarrow MSPF_{het} \leq \log(M)\, OPT_{het} \tag{2}$$

This concludes the proof. ■

B Implementation and Complexity Analysis

We present a low complexity implementation of MaxStrPF, OPF and any heterogeneous algorithm that follows a similar greedy framework.

We distinguish two phases.

Phase 1. Minimum Distances Calculation. We want to find minimum distances between each participant and all other nodes. Computational Complexity $O(M\,N^2)$ using Dijkstra's algorithm.

Phase 2. Tree creation.

1. Initialize.

   $newnode = source$, the node will join in the next step.

   $newstream = maxstream$, the stream offered to the $newnode$.

   $newpath = \{source\}$, the related path.

2. REPEAT until all destinations are reached

   (a) Join $newnode$ offering $newstream$ using $newpath$.

   (b) Update overrouting cost of tree nodes. Start from the $source$ and move towards the $newnode$. When the existingstream of a tree node increases, update the overrouting cost of all its descendents in the tree.

   For each tree node, the updates are limited to $\min(M, L)$. since each update is related with a stream increase, but also a stream increase happens when a new destination is added in the tree. In each update we modify the overrouting cost of each stream L.

   Total Computational Complexity: $O(L\,N\,\min(M, L))$.

   (c) Update the joining information of unreached destinations. For each node of the path $O(N)$, we update the joining information of each unreached destination $O(M)$, for all $O(L_{pref})$ preferred streams of each destination. In doing so, keep track of $bestnode$ destination according to the selection criterion of the algorithm. And the relate stream $beststream$ and path $bestpath$.

   Total Computational Complexity of Step: $O(L_{pref}\,M\,N)$.

   Note that for the fixed problem algorithms, we have only one stream per destination so the complexity of this step is $O(M\,N)$.

   (d) Set:

   $newnode = bestnode$

   $newstream = beststream$

   $newpath = bestpath$

Thus, the total computational complexity of the algorithm is bounded by $O(M\,N^2 + L\,M\,N + L^2\,N)$ for the general problem and by $O(M\,N^2 + L\,N\,\min(M, L))$ for the fixed problem. Note that the network manager or service provider can restrict the number of distinct streams, when the complexity becomes an issue. If we restrict the number of streams to $O(N)$, the complexity of the algorithms is bounded by $O(M\,N^2)$.

In Phase 1, we assume that the minimum distance paths are the same for all streams, and this is always true for the exponential cost function we mentioned. However, this may not be true if we assume finite link capacity, and high streams exceed the capacity of some edges. In that case, we have to find the minimum distance paths for each stream. Note that the exponential coast model does not assume a capacity limit. A straight forward implementation would repeat Phase 1 for every stream $O(L\,M\,N^2)$. However, we believe that more efficient implementations could be considered.

C Appendix

Minimum Cost Participant First Algorithm (MinCostPF)

1. Start from the source.
2. Repeat until all destinations are reached
   (a) Select the the nearest (minimum cost) destination, d_{new}, to the partial tree.
   (b) Include d_{new} and its path in the partial tree.

References

[BV95] F. Bauer and Anujan Varma. Degree-constrained multicasting in point-to-point networks. *Proc. IEEE INFOCOM*, pages 369– 376, 1995.

[DL93] M. Doar and I. Leslie. How bad is naive multicast routing? *Proc. IEEE INFOCOM*, pages 82–89, 1993.

[DW72] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[FPS96] M. Faloutsos, R. Pankaj, and K. C. Sevcik. Multicasting in directed graphs. *18th Biennial Symposium on Communications, June 2-5*, pages 29–32, 1996.

[Gha89] M. Ghanbari. Two-layer coding of video signals for VBR networks. *IEEE Journal on Selected Areas on Communications*, 7(5):771–781, 1989.

[GKR95] R. Gawlick, C. Kalmanec, and K.G. Ramakrishnan. On-line routing for permanent virtual circuits. *Proc. IEEE INFOCOM*, 1995.

[IW91] M. Imase and B.M. Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4:369–384, 1991.

[KMB81] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica, Springer-Verlag*, 15:141–145, 1981.

[KPP92] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicasting for multimedia applications. *Proc. IEEE INFOCOM*, pages 2078–2085, 1992.

[KPP93] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transaction on Networking*, pages 286–292, June 1993.

[KV89] M. Karlsson and M. Vetterli. Packet video and its integration into the network architecture. *IEEE Journal on Selected Areas on Communications*, 7(5):739–751, 1989.

[Meh88] K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters*, pages 125–128, 1988.

[Plo95] S. Plotkin. Competitive routing in ATM networks. *IEEE Journal on Selected Areas in Communications*, pages 1128–1136, August 1995. Special issue on Advances in the Fundamentals of Networking. (Invited paper).

[Ram96] S. Ramanathan. An algorithms for multicast tree generation in networks with asummetric links. *Proc. IEEE INFOCOM '96*, pages 337–344, 1996.

[RS83] V. J. Rayward-Smith. The computation of nearly minimal Steiner trees in graphs. *International Journal of Mathematical Education in Science and Technology*, 14(1):15–23, 1983.

[Sha92a] N. Shacham. Multicast routing of hierarchical data. *Proc. IEEE International Conference on Communication (ICC)*, pages 1217–1221, 1992.

[Sha92b] N. Shacham. Multipoint communication by hierarchically encoded data. *Proc. IEEE INFOCOM*, pages 2107–2114, 1992.

[SM94] N. Shacham and James S. Meditch. An algorithm for optimal multicast of multimedia streams. *Proc. IEEE INFOCOM*, pages 856–864, 1994.

[TM80] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

[Wax88] B. M. Waxman. Routing of multipoint connections. *IEEE Journal of Selected Areas in Communications*, pages 1617–1622, 1988.

[Wax93] B. M. Waxman. Performance evaluation of multipoint routing algorithms. *Proc. IEEE INFOCOM*, pages 980–986, 1993.

[Win87] P. Winter. Steiner problem in networks: a survey. *Networks*, 17:129–167, 1987.

[ZPGLA95] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves. A source-based algorithms for delay constrained minimum cost multicasting. *Proc. IEEE INFOCOM*, pages 377–385, 1995.

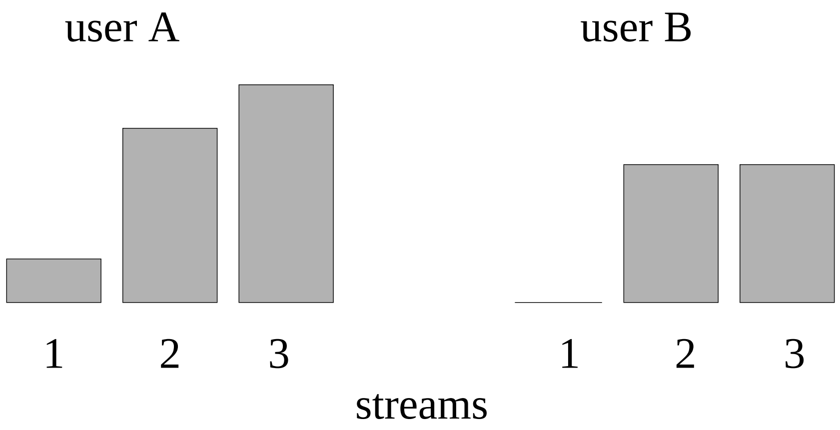

Figure 1: Happiness values for different users.

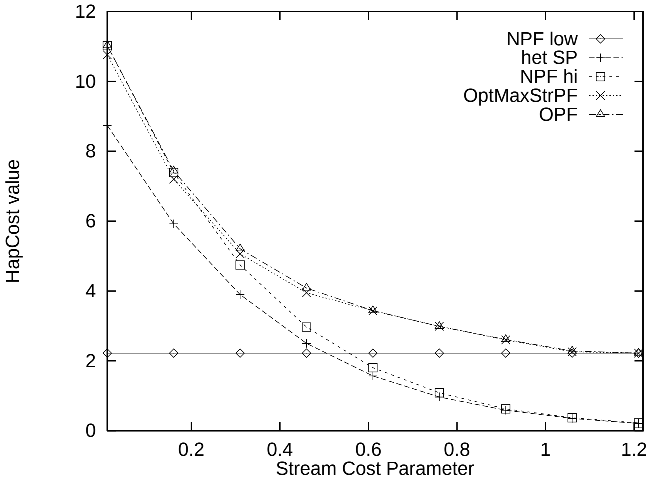

Figure 2: General problem: HapCost versus the stream cost of streams with objective $HapCost() = Hap/Cost$

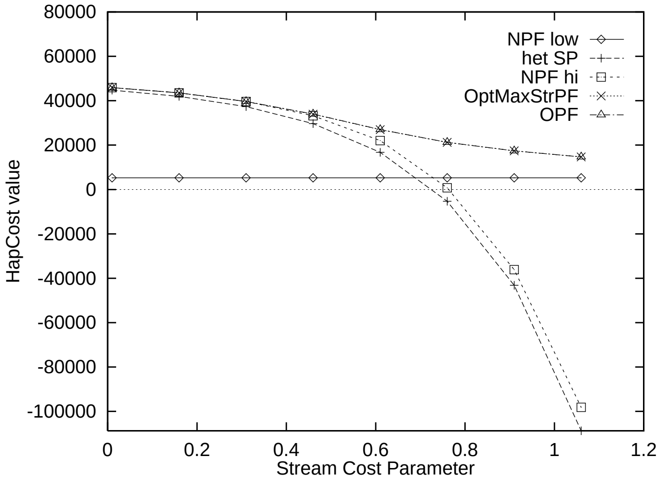

Figure 3: General problem: HapCost versus the stream cost of streams with objective $HapCost() = Hap - Cost$

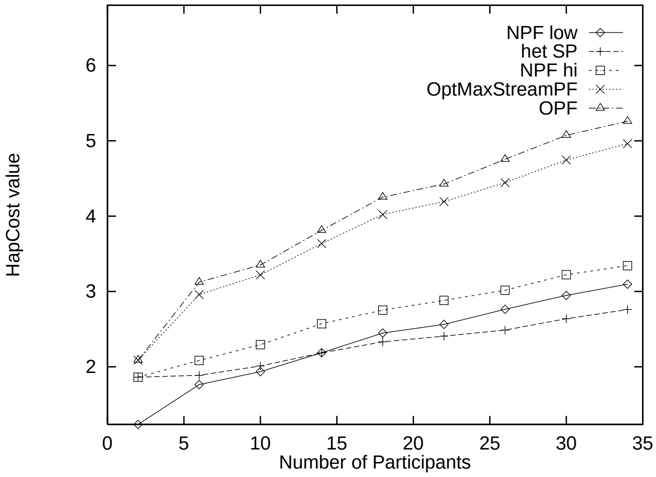

Figure 4: General problem: HapCost versus the number of participants for stream cost 0.5.

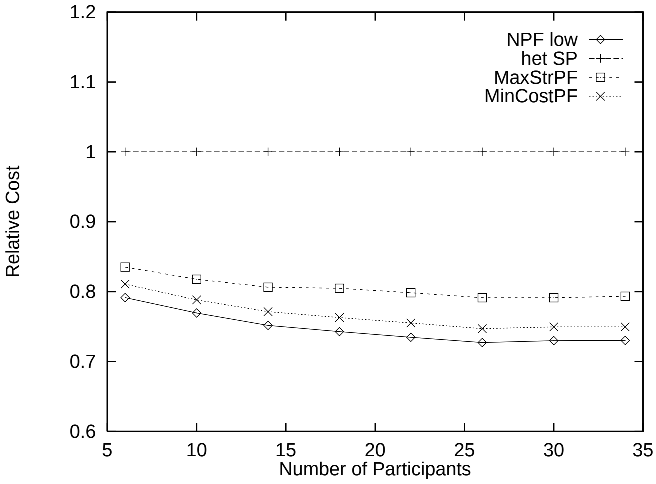

Figure 5: Fixed problem: Relative cost versus the number of participants for stream cost 0.01.

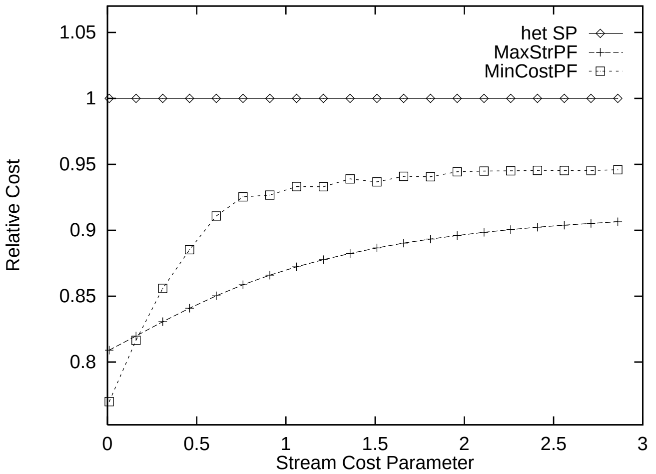

Figure 6: Fixed problem: Relative Cost versus the stream cost parameter for 14 participants.