CCIT Report #432 June 2003

Optimizing Hybrid Multicast and Unicast Overlay Networks

Oren Unger
Department of Electrical Engineering, Technion
and Zoran Microelectronics
Email: unger@tx.technion.ac.il

Israel Cidon
Department of Electrical Engineering
Technion - Israel Institute of Technology
Email: cidon@ee.technion.ac.il

Abstract

Overlay networks architecture should support high-performance and high-scalability at low costs. This becomes more crucial when communication, storage costs as well as service latencies grow with the exploding amounts of data exchanged and with the size and span of the overlay network. For that end, multicast methodologies can be used to deliver content from regional servers to end users, as well as for the timely and economical synchronization of content among the distributed servers. Another important architectural problem is the efficient allocation of objects to servers to minimize storage, delivery and update costs.

In this work, we suggest a multicast based architecture and address the optimal allocation and replication of dynamic objects that are both consumed and updated. Our model network includes application servers which are potential storage points connected in the overlay network, consumers which are served using multicast and/or unicast traffic and media sources which update the objects using multicast communication. General costs are associated with distribution (download) and update traffic as well as the storage of objects in the servers.

Optimal object allocation algorithms for tree networks are presented with complexities of $O(N)$ in case of multicast distribution and $O(N^2)$ in case of hybrid unicast/multicast distribution. A special case of the hybrid distribution problem automatically selects, for each user, between multicast and unicast distribution.

Using the techniques of the optimal tree algorithm we also present an efficient approximation algorithm for general networks in case of multicast only distribution.

Index Terms

Content Distribution, Location Problems, Multicast, Overlay Networks, Tree Networks

I. Introduction

Recent years have witnessed tremendous activity and development in the area of content and services distribution. Geographically dispersed consumers and organizations demand higher throughput and lower response time for accessing distributed content, outsourced applications and managed services. In order to enable high quality and reliable end-user services despite unpredictable Internet and Intranet conditions, organization and applications service providers (ASPs) employ content distribution networks (CDN) and overlay networks. These networks bring content and applications closer to their consumers, overcoming slow backbone paths, network congestions and physical latencies. Multiple vendors such as Cisco [1], Akamai [2] and Digital Fountain [3] offer CDN services and overlay technologies. Recently, more collaborative models such as distributed storage and peer-to-peer computational models require both consumption and modification of the content by multiple, geographically distributed users [4, 5].

An overlay network is a set of application servers that are connected through the general Internet Infrastructure. Naturally, organizations and ASPs try to optimize the overall cost of the overlay network mainly in terms of storage and communication costs. Efficient allocation of information objects to the overlay network servers reduces the operational cost and improves the overall performance. This becomes more crucial as the scale of services extend to a large number of users over international operation where communication and storage costs as well as network latencies are high. The optimization problem becomes more difficult as the service becomes dynamic and needs to be changed, updated and synchronized frequently.

The popularity of multicast for distribution of the content is increasing with the introduction of real-time and multimedia applications that consume high bandwidth and are delivered to a large number of consumers. Although multicast is efficient for a large number of consumers, unicast can still be more effective for a small number of consumers, especially for a sparse distribution of the consumers.

Overlay multicast networks are overlay networks which use multicast as the transport protocol between the vertices of the network [6]. Most of the overlay multicast networks are based on a single multicast tree that connects the participating vertices [6–9]. A network wide single multicast tree, may suffer from scalability, reliability and QoS problems as well as high communication costs due to the use of international and long distance links. Moreover, most IP backbones (backbone of Internet providers and even private WAN backbone) are not IP multicast enabled. On the other hand, multiple multicast trees, especially in LANs or campuses, rooted at regional servers, may take advantage of underlying IP multicast support and scale better. It can also save long distance communication costs and provide a better QoS by storing the high demand objects locally. The new approach suggested in this paper is to combine the replication of mirrors/proxies used in CDNs with multicast based distribution/update and achieve better scalability of the service while maintaining a low cost of storage and communication.

Our initial model is a tree graph that has a server located at each of its vertices. The vertices also include optional entries to local consumers and media sources. Each server is assigned with a storage cost and each edge is assigned with distribution and update communication costs. The distribution demand of the consumers and the update requirements of the media sources are known a-priory. The consumers are served from servers using multicast and/or unicast communication. The media sources update and modify the objects within the servers. The update traffic between a media source and the relevant servers is most efficiently conducted using multicast communication, since it can reduce significantly the overall update transport and the update latency.

Our goal is to find an optimal allocation, e.g., the set of servers which store an object, with the minimum overall (communication and storage) cost. The consumers are assigned to the servers in a way that each consumer is served by exactly one server for an object. It is clear that by changing the number of copies, we introduce a tradeoff between the storage/update costs that increase with the number of copies and the distribution cost that decreases with this number.

In [10], we presented an optimal allocation algorithm for the multicast only distribution on trees. In this work we extend the problem in two new directions. The first direction is an optimal allocation algorithm for the hybrid unicast/multicast distribution on trees with computational complexity of $O(N^2)$. We present two different cases where the mode of operation per consumer (multicast or unicast) is given a-priori or is automatically optimized by the algorithm itself. The second direction is an algorithm for general networks in case of multicast only distribution. In the general network case, we replace the fixed tree structure with a (approximated) Steiner tree as the suggested multicast tree structure and apply the same optimization techniques over that tree. Once new content locations are assigned we can apply this scheme repeatedly until it converges.

A. Related work

Application level multicast and overlay multicast protocols have been studied in recent years. Most of the works are focused on the structure of the overlay topology (i.e. the way the multicast tree is constructed) for a single tree [6–9]. Our work assumes the overlay network topology is a tree, but instead of focusing on the construction of such a tree, we focus on the way the tree should be partitioned to multiple regional multicast trees while optimizing the communication and storage cost.

The object allocation problem, also referred as the file allocation problem in storage systems [11] or data management in distributed databases has been studied extensively in the literature. Kalpakis et al. [12] and Krick et al. [13] present a model of a network with unicast reads, multicast writes and storage costs. [12] presents a problem with additional constrains for a tree network and the algorithm they suggest is less efficient than our hybrid distribution algorithm. [13] deals with general networks and suggests an optimal algorithm in tree networks which is also less efficient than our hybrid distribution algorithm. These works don't solve the multicast reads and multicast writes problem. Moreover, [12, 13] use an MST based update in which a media source sends a unicast message to the closest server (which stores an object) and the server itself forwards a multicast message over the MST to other servers (i.e. - multicast is used only between the servers). This scheme does not employ properly the native IP multicast model where a single source can send traffic directly to all the servers via a multicast tree. In terms of computational complexity, our algorithm is $O(N^2)$, compared to $O(N^5)$ in [12] and $O(N \cdot diam(T) \cdot \log(deg(T)))$ in [13] (Worst case is $O(N^2 \cdot \log(N))$). Additional works that address the severs/replicas placement problem for the read only unicast distribution model can be found in [14–17].

II. The Model

A. Objects

For each object o of the objects set O, we determine the set of servers which store a copy of the object. The algorithm handles each object separately, so the costs described below are defined (and can be different) for each object o.

B. The tree network

Let $T = (V, E)$ be a tree graph that represents a communication network, where $V = \{1, \ldots, N\}$ is the set of vertices and E is the set of edges. The tree is rooted at any arbitrary vertex r (r=1). Each vertex in the tree represents a network switch and a potential storage place for object copies. Each vertex in the tree is also an entry point of content consumers and/or media sources to the network. Distribution demands of consumers connected to vertex i are provided by the network from the server at the closest vertex (or the closest multicast tree rooted at) j which stores a copy of the object. An object update may be provided by any media source and is sent to all the vertices that store the object using multicast.

Denote the subtree of T rooted at vertex i as T_i.
Denote the parent vertex of vertex i in T ($i \neq r$) as P_i.
Denote the edge that connects vertex i to its parent in T, (i, P_i) as e_i ($e_r = \emptyset$).
Denote the set of edges in $T_i \cup e_i$ as E_i ($E_r \equiv E$).
Denote the set of vertices in T_i as V_i ($V_r \equiv V$).
Denote the set of children vertices of vertex i in T as Ch_i (For a leaf i, $Ch_i = \emptyset$).
Figure 1 displays a tree network with various costs related to its vertices and edges.

C. Storage cost

Let the storage cost of the object at vertex i to be Sc_i.
Denote Φ is the set of vertices that store the object.
The total storage cost of the object is $\sum_{i \in \Phi} Sc_i$.

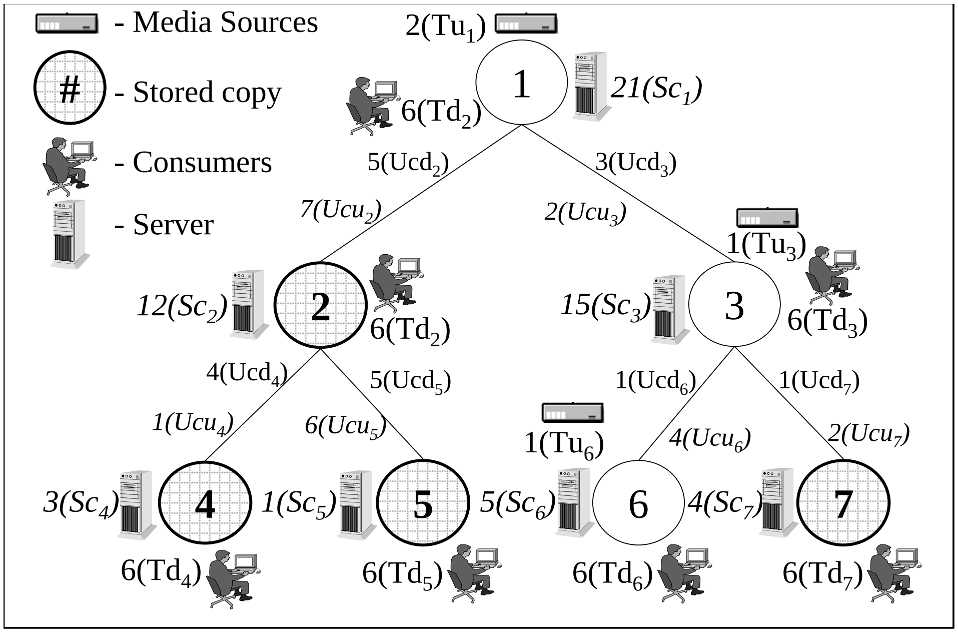

Fig. 1. An example of a tree network and various costs

D. Distribution traffic cost

Denote the cost per distribution traffic unit at edge e_i as Ucd_i (Ucd_i>0). Since $e_r = \emptyset$, $Ucd_r \equiv 0$.

1) Multicast distribution traffic cost: The multicast distribution traffic provided to vertex i, Tdm_i, is either Td or 0. Td is used when at least one consumer connected to i requires the object and 0 is used when no consumers connected to i require the object[1].

Denote Dmt_i the set of edges in the distribution multicast tree rooted at vertex i. If $i \notin \Phi$, $Dmt_i = \emptyset$.

The total multicast distribution traffic cost is $\sum_{i\in\Phi} Td \cdot (\sum_{e\in Dmt_i} Ucd_e)$.

2) Unicast distribution traffic cost: The cost per distribution traffic unit along a path between vertices i and j is $Dd_{i,j} = \sum_{e\in P_{i,j}} Ucd_e$, where $P_{i,j}$ is the set of edges that connect vertex i to vertex j. We define $P_{i,i} \equiv \emptyset$ and $Dd_{i,i} \equiv 0$. Since the tree is undirected, $P_{i,j} = P_{j,i}$.

The total distribution traffic demand produced by all the consumers connected to vertex i is Tdu_i ($Tdu_i \geq 0$).

The total unicast distribution traffic cost is $\sum_{i\in V} Tdu_i \cdot \min_{j\in\Phi} Dd_{i,j}$. (If $\exists j, k \in \Phi$ s.t. $Dd_{i,j} = Dd_{i,k}$ and $j < k$ then j, the smallest index, is taken).

E. Multicast update traffic cost

Denote the cost per update traffic unit at edge e_i as Ucu_i ($Ucu_i > 0$). Since $e_r = \emptyset$, $Ucu_r \equiv 0$.

The total multicast update traffic generated by all the media sources connected to vertex i is Tu_i ($Tu_i \geq 0$).

Denote $Umt_{i,\Phi}$ as the set of edges of the multicast update tree from vertex i to Φ.

The total update traffic cost is $\sum_{i\in V} Tu_i \cdot \left(\sum_{e\in Umt_{i,\Phi}} Ucu_e\right)$.

III. The Problem

The optimization problem is to find an object allocation that minimizes the total cost (storage and traffic):

$$\sum_{i\in\Phi} Sc_i + \sum_{i\in V} Tu_i \cdot \left(\sum_{e\in Umt_{i,\Phi}} Ucu_e\right) + \sum_{i\in\Phi} Td \cdot \left(\sum_{e\in Dmt_i} Ucd_e\right) + \sum_{i\in V} Tdu_i \cdot \min_{j\in\Phi} Dd_{i,j}$$

We developed an algorithm that solves the above optimization problem. The algorithm is called HDT (Hybrid multicast/unicast Distribution on Tree graphs). The HDT algorithm is presented in section VIII.

Based on the general optimization problem we derived additional novel problems which are solved in this paper:

1) Multicast only distribution - we omitted the unicast distribution traffic from the general problem (and its total cost $\sum_{i\in V} Tdu_i \cdot \min_{j\in\Phi} Dd_{i,j}$).
   For a tree graph, we developed an algorithm called MDT (Multicast Distribution on Tree graphs).
   For a general graph, we developed an approximation algorithm called MDG (Multicast Distribution on General graphs), which uses a variant of MDT.
2) Mutual Exclusive hybrid distribution - the algorithm automatically selects between multicast and unicast distribution to consumers. We replace the unicast distribution cost with $\sum_{i\in V_{uc}} Tdu_i \cdot \min_{j\in\Phi} Dd_{i,j}$, where V_{uc} the set of vertices which are served using unicast.
   For a tree graph, we developed an algorithm called MX-HDT (Mutual eXclusive Hybrid Distribution on Tree graphs).

IV. Optimal Allocation Properties

These properties are the fundamentals of our technique.

A. Per edge update traffic

As described in section II, a vertex i which is a root of a multicast update tree produces Tu_i update traffic through each edge $e \in Umt_{i,\Phi}$. The update traffic of such a tree is directed from i to Φ. Since the location of the media sources is known a-priory, when we look at a single edge, we can determine the update traffic that will pass through it in each direction, in case there are copies stored in the subtrees connected to it (in both ends of the edge).

[1] The reason for using the same traffic rate for all vertices in multicast is the fact that the server determines the transmission rate, not each customer as in the unicast case.

For each edge e_i we define Tu_i^{out} and Tu_i^{in}. Tu_i^{out} is the total update traffic that is outgoing via vertex i and edge e_i out of T_i, in case there is at least one copy stored outside T_i. Tu_i^{in} is the total update traffic that is incoming via vertex i and edge e_i into T_i, in case there is at least one copy stored in T_i.

$$Tu_i^{out} \leftarrow \sum_{j \in V_i} Tu_j = Tu_i + \sum_{c \in Ch_i} Tu_c^{out}$$
$$Tu_i^{in} \leftarrow \sum_{j \notin V_i} Tu_j = Tu_r^{out} - Tu_i^{out}$$

B. Distribution traffic properties

Lemma 1: In the optimal allocation, in case of unicast distribution, if vertex i is served from vertex j, which satisfies $\min_{j\in\Phi} Dd_{i,j}$, and i is served through vertex k (i.e. $P_{i,j}{=}P_{i,k}{\cup}P_{k,j}$), then k must also be served from j.

Proof: $P_{i,j}{=}P_{i,k}{\cup}P_{k,j} \Rightarrow Dd_{i,j}{=}Dd_{i,k}{+}Dd_{k,j}$. Suppose vertex k is not served from j, but from a different vertex l. Since the solution is optimal there must exist $Dd_{k,l}{<}Dd_{k,j}$.
In that case we get $Dd_{i,l}{=}Dd_{i,k}{+}Dd_{k,l}{<}Dd_{i,k}{+}Dd_{k,j} \Rightarrow$ a contradiction. ■

Lemma 2: In the optimal allocation, each vertex i can only belong to at most one multicast distribution tree.

Proof: Suppose a vertex i belongs to more than one multicast distribution tree, then by disconnecting it from the other trees and keeping it connected to only one multicast distribution tree we reduce the distribution traffic in contradiction to the optimality of the cost. ■

Lemma 3: In the optimal allocation, if vertex i is served through its neighbor k in T (either parent or child), then i and k are served from the same server.

Proof: The proof is a direct result of lemmas 1 , 2. ■

Corollary 1: The optimal allocation is composed of a subgraph of T which is a forest of unicast and multicast distribution subtrees. Each subtree is rooted at a vertex where a copy is located and its leaves are vertices were no copy is stored and there is distribution demand. Each edge and vertex in T can be part of at most one unicast and at most one multicast distribution subtree.

V. Tree based Algorithms Technique

The main idea behind the algorithms is the observation that in tree graphs, since there is only one edge from each vertex i to its parent, and due to lemma 3, if we consider the influence of the optimal allocation outside T_i on the optimal allocation within T_i, it is narrowed to a very small number of possibilities. We just have to consider the possibility that there are or not copies outside T_i (and if i is served from such an external copy, where is it located), there is or the isn't multicast distribution demand outside T_i and also consider the possibility that no copy is located within T_i (only when $i{\neq}r$). We define scenarios that are possible for each vertex pair i,j in unicast distribution and for vertex i in multicast distribution, which cover all these possible external influences on the optimal allocation within T_i. In addition, due to the same lemma 3, it is fairly easy and straight forward to calculate the optimal allocation for vertex i and T_i based on the optimal allocation calculated for each c and T_c, where $c \in Ch_i$.

As a result, our algorithms for tree graphs are recursive algorithms that find the optimal allocation for a new problem which is a subset of the original problem for vertex i and T_i, based on the optimal allocation computed by its children Ch_i for their subsets of the original problem. (There are different new problems for the multicast/hybrid distribution cases).

The algorithms are performed in two phases. The first phase is the cost calculation phase which starts at the leaves and ends at the root, while calculating the optimal allocation and its alternate cost for each vertex pair i,j in hybrid unicast/multicast distribution and each vertex i in multicast only distribution and for each scenario, based on the optimal allocations calculated by the children of vertex i for all their possible scenarios. The second phase is a backtrack phase which starts at the root and ends at the leaves where the algorithm selects the scenario which is active in the optimal allocation (in the optimal solution there can be only one actual scenario possible for each vertex) and allocates the copies in the relevant servers. The second phase is needed since only in the root it is possible to find the optimal allocation of the entire tree, and since the algorithm works in a recursive way, the root doesn't know the entire optimal allocation, but only the actual scenarios of itself and its children as well as the cost of the optimal allocation.

The algorithms calculate the optimal object allocation cost as well as the set of servers that will store the object.

VI. The MDT algorithm

The MDT algorithm was previously presented in [10], and this section provides a short reminder of that algorithm, since its the simplest algorithm to understand.

Besides omitting the unicast distribution traffic, the algorithm assumes that the multicast distribution demand is the same for all the vertices ($\forall i \in V, Tdm_i{=}Td$).

As described in section V, the algorithm is performed in two phases, the cost calculation phase and the backtrack phase. We will only remind the cost calculation phase.

For the new problem we define a new tree, which is a subtree of T constructed of T_i and e_i. We define 4 legal scenarios for a vertex i and T_i.

A. *The cost calculation phase*

For each vertex i the algorithm calculates for T_i 4 alternate costs, for the following possible scenarios:

Cxi_i - There is no copy located inside T_i ($i{\neq}r$). Edge e_i will carry incoming distribution and outgoing update traffic.

Cbi_i - Copies are located both inside and outside T_i but not all the internal consumers demand is supplied from copies in T_i. Edge e_i will carry incoming distribution and both incoming and outgoing update traffic.

Cbo_i - Copies are located both inside and outside T_i and all the internal consumers demand is supplied from copies in T_i. Edge e_i will carry both incoming and outgoing update (and maybe outgoing distribution) traffic.

Cio_i - All the copies of the object are located only inside T_i. Edge e_i will carry incoming update (and maybe outgoing distribution) traffic.

The algorithm calculates the costs as follows:

$$Cxi_i \leftarrow \begin{cases} Td \cdot Ucd_i + Tu_i^{out} \cdot Ucu_i + sum4, & \text{if } i \neq r \\ \infty, & \text{if } i = r \end{cases}$$

$$Cbi_i \leftarrow \begin{cases} Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + Tu_i^{out} \cdot Ucu_i + sum1\ , & \text{if } i \neq r \ \&\ Ch_i \neq \emptyset \\ \infty, & \text{if } i = r \parallel Ch_i = \emptyset \end{cases}$$

$$Cbo_i \leftarrow \begin{cases} \left(Tu_i^{in} + Tu_i^{out}\right) \cdot Ucu_i + \min\{min1, min2\}\ , & \text{if } i \neq r \\ \infty, & \text{if } i = r \end{cases}$$

$$Cio_i \leftarrow \begin{cases} Tu_i^{in} \cdot Ucu_i + \min\{min1, min2, min3\}\ , & \text{if } i \neq r \\ \min\{min1, min2, min3\}, & \text{if } i = r \end{cases}$$

where (various combinations of children scenarios):

$$\begin{aligned} sum1 &= \sum_{c \in Ch_i} \min\{Cxi_c, Cbo_c, Cbi_c\} \\ sum2 &= \sum_{k \in Ch_i, k \neq c} \min\{Cxi_k, Cbo_k, Cbi_k\} \\ sum3 &= \sum_{k \in Ch_i, k \neq c} Cxi_k \\ sum4 &= \sum_{c \in Ch_i} Cxi_c \\ min1 &= Sc_i + sum1 \\ min2 &= \min_{c \in Ch_i} \{Td \cdot Ucd_c + Cbo_c + sum2\} \\ min3 &= \min_{c \in Ch_i} \{Td \cdot Ucd_c + Cio_c + sum3\} \end{aligned}$$

note: $sum1, sum2, sum3, sum4$ equal 0 and $min2, min3$ equal ∞ if vertex i is a leaf ($Ch_i{=}\emptyset$).

The cost of the optimal allocation in T is Cio_r.

The computational complexity of MDT is $O(N)$. The explanation for the complexity calculation, as well as proof of optimality can be found in [10].

VII. The MDG approximation algorithm

The problem of finding the optimal allocation for multicast only distribution traffic in general graphs is NP-hard. Nevertheless, we can analyze the properties of the optimal solution and suggest an approximation algorithm for the optimization problem, based on the optimal allocation algorithm we developed for tree graphs.

A. Model changes for the General Graph

Besides omitting the unicast distribution traffic from the problem, we replace the tree graph (described in II-B) with a general graph:
Let $G = (V, E)$ be a connected graph that represents a communication network, where $V = \{1, \ldots, N\}$ is the set of vertices and E is the set of edges.

The notation of an edge e in the general graph is different from the one we defined in the tree graph. We define $e_{i,j}$ - the edge that connects vertices i and j.

In addition, we define the same distribution and update cost per traffic unit at edge $e_{i,j}$. I.e. $Ucd_e \equiv Ucu_e$. This is a reduction of the original traffic cost model, but it is still reasonable since in the real world usually the cost per traffic unit is the same for all kinds of traffic.

Figure 2. displays a graph network with various costs related to its vertices and edges.

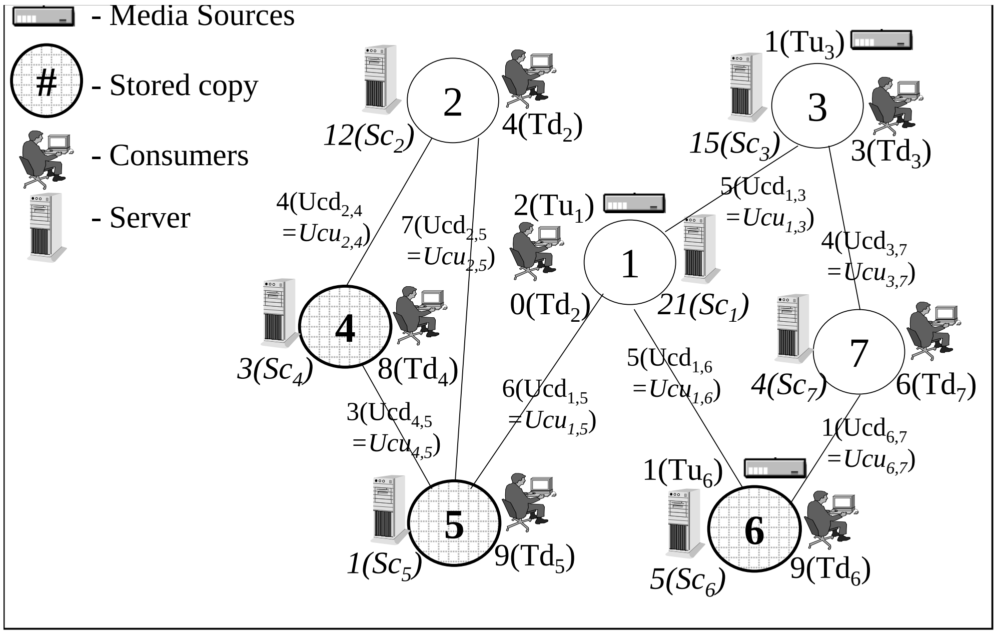

Fig. 2. An example of a graph network and various costs.

B. The optimal allocation properties in general graphs

The Steiner tree problem [18] is defined as follows: given an undirected graph with a specified subset of vertices called the terminals, find a tree with the minimum cost (lengths) of edges spanning all the terminals. According to this definition, an optimal multicast tree in a general graph is a Steiner tree. The Steiner tree problem is NP-hard on general graphs [19].

The property of lemma 2 is also valid in general graphs. Each vertex belongs to at most one multicast distribution tree. If we look at the optimal allocation, we can see again a forest of multicast trees that cover all the multicast consumers. Each media source is a root of a multicast update tree that connects it to all the vertices which store an object. The optimal solution in a general graph is a forest of Steiner trees that connect all the media sources to all the servers which store an object, and each server to the set of consumers which it serves. The total cost of the optimal allocation is constructed of the storage cost, the multicast update Steiner trees costs and the multicast distribution Steiner trees costs.

Since finding a Steiner tree in a general graph is NP-hard, it is obvious that finding a forest of Steiner trees is NP-hard as well.

C. The approximation algorithm heuristics

Since the allocation problem in general graphs is NP-hard, we use our optimal algorithm for trees in order to find an approximation to the optimization problem.
We defined an efficient iterative algorithm that starts with a random or preset allocation, and converges to an allocation which is optimal in an approximated Steiner tree extracted from the general graph.
In the model described above, the multicast distribution demand is not the same for all the vertices in the graph (Tdm_i, is either Td or 0), so we use a variant of the MDT algorithm described in section VI to solve the optimization problem on the tree. We refer to this modified algorithm as XMDT (eXtended MDT). The detailed XMDT algorithm (e.g. cost calculation formulas as well as the pseudo code of that algorithm) can be found in appendix I. We also provide an algorithm for calculating the allocation cost in the general graph. We will refer to the cost calculation algorithm as $COST_{alloc}$.

1) The MDG algorithm steps: The approximation algorithm in the general graph is:

1) Start with a random allocation. The number of copies allocated is either a constant number or a percentage of N - the number of vertices in the graph (network size) .
2) Run $COST_{alloc}$ on the initial allocation, save the initial allocation and set min_{cost} to be the current cost.
3) Extract a Steiner tree from the general graph where the terminals are all the vertices with either distribution or update demand and the vertices which store the object.
4) Run XMDT on the extracted Steiner tree. The algorithm will allocate copies in vertices of the extracted tree.
5) Run $COST_{alloc}$ on the current allocation. If the cost is smaller than min_{cost} save the current allocation and update min_{cost} to be the current cost.
6) Repeat steps 3 to 5 till there is no improvement in the allocation cost.

At the end of the algorithm, the saved allocation is the suggested approximated allocation, and the saved cost (in min_{cost}) is the approximated cost.

Figure 3 describes the MDG algorithm flow chart.

2) Extracting a Steiner tree: The problem of finding a Steiner tree is NP-hard. There are several polynomial time approximation algorithms for the problem. We selected the approximation algorithm suggested by Zelikovsky [20], which has an approximation ratio of $11/6$ from the optimal solution. The Steiner points are the vertices that have multicast or unicast demand, as well as the vertices which currently store a copy as the terminals for the Steiner tree problem. We use Ucd_e ($\equiv Ucu_e$) as the edge length.

3) Running the XMDT optimal algorithm on the tree: While extracting the Steiner tree, we also set the storage cost and distribution and update demands for each vertex in the extracted tree according to the original values in G. We get a tree network on which the allocation problem can be solved using the optimal allocation algorithm for tree networks (XMDT). The algorithm finds a new allocation for the given tree. The resulting allocation may be the same as the previous one, especially if the extracted tree is similar to the previous one.

4) Calculating the cost for a given allocation: The total cost of the allocation is constructed of the storage cost, the multicast update Steiner trees costs and the multicast distribution Steiner trees costs.

Calculating the cost of a given allocation is an approximation algorithm by itself. Given the set of vertices with media sources and their update demand, the set of vertices which allocate the object and the set of vertices with consumers and their distribution demand, we need to find the forest of Steiner trees (which was not seen by XMDT, since it was running on a tree and not on the entire graph).

The storage cost of the allocation remains $\sum_{i\in\Phi} Sc_i$.

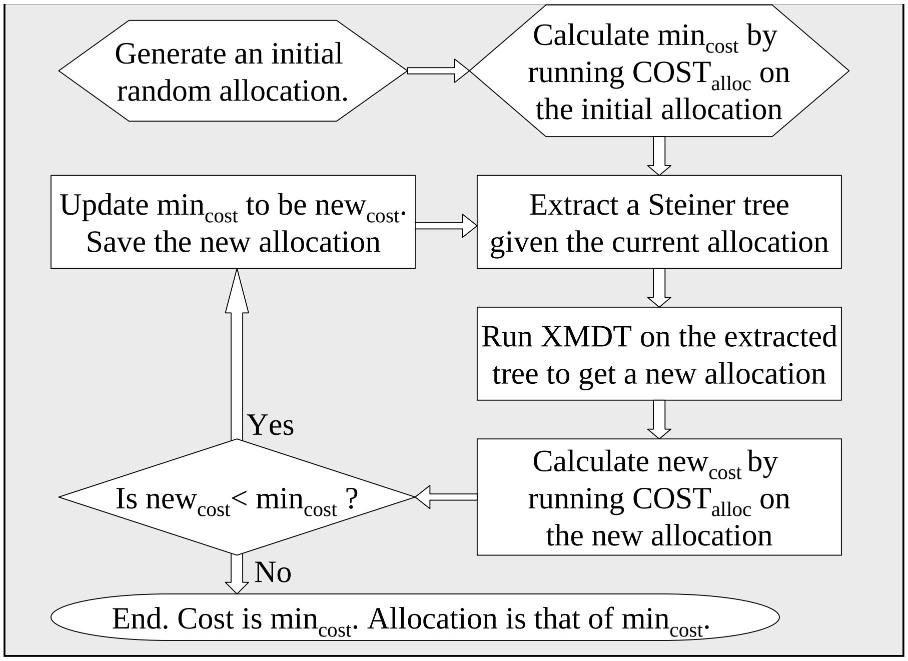
Generate an initial random allocation.
Calculate min$_{cost}$ by running COST$_{alloc}$ on the initial allocation
Extract a Steiner tree given the current allocation
Update min$_{cost}$ to be new$_{cost}$. Save the new allocation
Run XMDT on the extracted tree to get a new allocation
Calculate new$_{cost}$ by running COST$_{alloc}$ on the new allocation
Is new$_{cost}$< min$_{cost}$?
Yes
No
End. Cost is min$_{cost}$. Allocation is that of min$_{cost}$.

Fig. 3. The flow chart of the MDG algorithm.

Each multicast update tree is a Steiner tree and is fairly easy to find. For each vertex with a media source, we extract a Steiner tree that contains that vertex and all the vertices which store the object. The cost of such Steiner tree rooted at vertex i is the update traffic demand Tu_i multiplied by the total edge lengths of the Steiner tree, i.e. $Tu_i \cdot \left(\sum_{e \in Ust_{i,\Phi}} Ucu_e\right)$, where $Ust_{i,\Phi}$ is the set of edges in the update Steiner tree rooted at i.

It is much difficult to find the multicast distribution forest, since each vertex can be connected to at most one multicast tree. We use an approximation for that problem: extract a singe distribution-only Steiner tree that contains all the vertices which store the object and all the vertices with positive distribution demand. For each vertex that stores the copy, set the storage cost to 0 and for each vertex that doesn't store the object to ∞. Ignore the update demands of the media sources. Use the original distribution demand values and unit cost per edge. Run XMDT on that tree. The result is a forest of distribution trees. Please note that the extracted distribution-only Steiner tree may differ from the Steiner tree used by XMDT to find the current allocation, since it doesn't contain the vertices with only update demand (i.e. - don't have a distribution demand).

D. MDG Simulation results

We've generated general graphs using the Internet Model by Zegura et al. [21, 22]. We defined two Transit/Stub based models which differ in the way the network is partitioned to top level domains and local domains.

In all the graphs, the unit traffic cost per edge was taken from the Internet model, while the values of storage costs and update/distribution demands were randomly generated using the following guidelines: Uniform distribution of the storage cost in the range of $[10, 130]$ (average cost is 70); 25% of the total number of vertices have distribution demand (Td set to 4); A very small number (up to 3) of media sources with Tu set to 1.

We've run our approximation MDG algorithm on these graphs, and compared the results to several random allocations for each graph. The difference between the random allocations is the number of copies. RAND3, RAND5 - an allocation with an average number of 3, 5 copies respectively. RAND - an allocation with an average number of copies which follows the average number of copies by MDG.

In each Internet model and for each network size, we defined ten (10) random graphs and ran MDG on each such graph. We also generated the random allocations as described above. We've calculated the average cost and average number of copies for each allocation type and each network size. The average costs are presented in table I.

We also generated charts of the average number of copies and average cost vs. the number of vertices in each model. Each chart contains four (4) series - one of the MDG allocation and the other three are of the different random

Model 1

size	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
MDG	371	358	489	507	739	981	811	791	1137	1087	1304	1058	1486	1654	1522	1821
RAND	610	646	902	894	1350	1582	1392	1645	2044	2009	2308	1819	2545	2843	2644	3415
RAND3	521	586	874	909	1190	1468	1361	1414	1834	1814	1967	1804	2217	2509	2345	2737
RAND5	593	735	904	917	1209	1572	1326	1501	1759	1786	2038	1667	2276	2532	2339	3263
MDG/$\overline{\text{RND}}$		0.55	0.55	0.56	0.59	0.64	0.60	0.52	0.61	0.58	0.62	0.60	0.63	0.63	0.62	0.58

Model 2

net size	15	21	27	33	39	45	51	57	63	69	75	81	87	93
MDG cost	329	422	597	610	843	840	1113	1294	1190	1417	1289	1384	1664	2109
RAND cost	579	765	1017	1132	1473	1448	1992	2059	2135	2252	2523	2555	3044	3491
RAND3 cost	544	795	1030	1076	1422	1309	1701	1799	1885	2109	2180	2506	2495	3006
RAND5 cost	685	738	987	1074	1458	1310	1796	1993	1918	2196	2287	2450	2621	3075
MDG/$\overline{\text{RND}}$	0.55	0.55	0.59	0.56	0.58	0.62	0.61	0.66	0.60	0.65	0.55	0.55	0.61	0.66

TABLE I

THE AVERAGE COSTS OF ALLOCATIONS IN GENERAL GRAPHS

allocations. Figure 4 displays these charts.

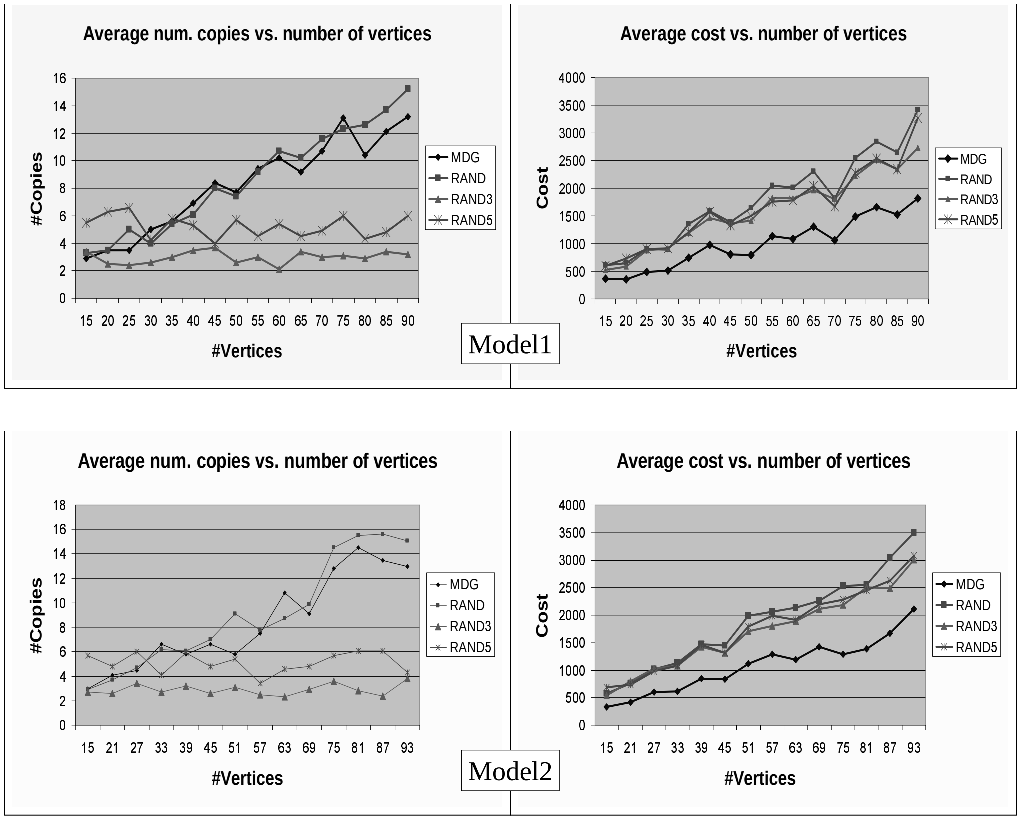

Fig. 4. The number of copies and cost of allocations in general graphs

Our conclusions from the results are:

1) The average costs of the MDG algorithm allocations are significantly better than of the random allocations. The average ratio between the MDG allocations costs and the random allocations costs is 0.6 for both models.

2) In both our Internet models, the average costs of the different random allocations produced almost similar costs. This indicates that the number of copies didn't have much affect on the overall random allocations costs. This can happen when the storage and update costs are relatively lower than the distribution costs (so the major part of the cost is the distribution traffic, which is very sensitive to the location of copies).
3) It can be seen, in both models, that the average number of copies and the average costs of the MDG allocations are increasing as the number of vertices grows. This behavior seems to be very reasonable - as there are more consumers in the network it is better to add more copies. The average cost increases since the storage and traffic costs increase as the network size and number of copies increase.

VIII. The HDT optimal algorithm

In this section we describe the general algorithm that solves the optimization problem defined in section III.

A. The algorithm

As described in section V, the algorithm is performed in two phases, the cost calculation phase and the backtrack phase.

For the new problem we define a new tree, $T_{i,j}$, which is a subtree of T constructed of T_i (the subtree of T rooted at i), and the additional set of edges e_i (connects vertex i to its parent) and $P_{i,j}$ (the string that connects vertex i to j), in case $j \notin T_i$.

The new optimization problem is defined as follows: Find the optimal allocation and its alternate cost in $T_{i,j}$, given the following assumptions:

1) There is a copy located at vertex j, $j \in V$. If $j \in V_i$ vertex i is served by unicast distribution from j. If $j \notin V_i$ and vertex i is served by unicast distribution from outside T_i it is served from j. When $j \notin V_i$, since vertex j is not part of $T_{i,j}$ (just the path to it), its storage cost is ignored. Note: the difference between the case of $j \in V_i$ and $j \notin V_i$, is that i has the entire data (including storage cost) regarding copies allocation inside T_i, but only the unicast distribution cost from copies located outside T_i. For $j \notin V_i$, i may decide that storing a copy in an internal vertex l and be served from that copy is less expensive than being served from j. This is only relevant for unicast distribution traffic.
2) There is or isn't a copy located inside T_i. Relevant for update/distribution multicast traffic. Also - the values of j must comply with this assumption.
3) There is or isn't a copy located outside T_i. Relevant for update/distribution multicast traffic. Also - the values of j must comply with this assumption.
4) When there are copies outside T_i, is there a need for incoming multicast distribution. I.e. are there consumers inside T_i that are connected to a multicast distribution tree through edge e_i. Only relevant for multicast distribution traffic.
5) When there are copies inside T_i, is there a need for outgoing multicast distribution. I.e. are there consumers outside T_i that are connected to a multicast distribution tree through edge e_i. Only relevant for multicast distribution traffic.

Based on the above assumptions we define seven (7) scenarios that are possible for each vertex pair i, j. These scenarios cover all the possible external influences on the optimal allocation within T_i.

B. The cost calculation phase

For each vertex pair i, j the algorithm calculates for $T_{i,j}$, (vertex j is assumed to allocate a copy of the object), seven alternate costs, for the following seven possible scenarios:

$Cxn_{i,j}$- **C**ost of e**X**ternal only object allocation and **N**o incoming multicast distribution traffic. There is no copy located inside T_i ($i{\neq}r$) and there is no internal multicast demand. Edge e_i will only carry outgoing update traffic. Legal only when $j \notin V_i$.

$Cxi_{i,j}$ - **C**ost of e**X**ternal only object allocation and **I**ncoming multicast distribution traffic. There is no copy located inside T_i ($i{\neq}r$) and there is an internal multicast demand. Edge e_i will carry incoming multicast distribution and outgoing update traffic. Legal only when $j \notin V_i$.

$Cin_{i,j}$ - **C**ost of **I**nternal only object allocation and **N**o outgoing multicast distribution traffic. All the copies of the object are located only inside T_i and there is no external multicast demand. Edge e_i will carry incoming update traffic. Legal only when $j \in V_i$.

$Cio_{i,j}$ - **C**ost of **I**nternal only object allocation and **O**utgoing multicast distribution traffic. All the copies of the object are located only inside T_i and there is an external multicast demand. Edge e_i will carry incoming update and outgoing multicast distribution traffic. Legal only when $j \in V_i$.

$Cbn_{i,j}$ - **C**ost of **B**oth sides object allocation and **N**o multicast distribution traffic. Copies are located both inside and outside T_i and there is no multicast demand through edge e_i. Edge e_i will carry both incoming and outgoing update traffic.

$Cbi_{i,j}$ - **C**ost of **B**oth sides object allocation and **I**ncoming multicast distribution traffic. Copies are located both inside and outside T_i and there is an internal multicast demand through edge e_i. Edge e_i will carry incoming multicast distribution and both incoming and outgoing update traffic.

$Cbo_{i,j}$ - **C**ost of **B**oth sides object allocation and **O**utgoing multicast distribution traffic. Copies are located both inside and outside T_i and there is an external multicast demand through edge e_i. Edge e_i will carry both incoming and outgoing update and outgoing multicast distribution traffic.

The algorithm calculates the alternate costs as follows:

$$Cxn_{i,j} \leftarrow \begin{cases} \infty, & \text{if } j \in V_i \\ Tdu_i \cdot Dd_{i,j} + Tu_i^{out} \cdot Ucu_i + sum1\,, & \text{if } j \notin V_i \end{cases}$$

$$Cxi_{i,j} \leftarrow \begin{cases} \infty, & \text{if } j \in V_i \\ Td \cdot Ucd_i + Tdu_i \cdot Dd_{i,j} + Tu_i^{out} \cdot Ucu_i + sum2\,, & \text{if } j \notin V_i \end{cases}$$

$$Cin_{i,j} \leftarrow \begin{cases} Tdu_i \cdot Dd_{i,j} + Tu_i^{in} \cdot Ucu_i + \min\left\{sum4, sum5, sum6, sum8, min1\right\}, & \begin{array}{l}\text{if } j \in V_k,\\ \quad k \in Ch_i\end{array} \\ Tu_i^{in} \cdot Ucu_i + Sc_i + sum3, & \text{if } j = i \\ \infty, & \text{if } j \notin V_i \end{cases}$$

$$Cio_{i,j} \leftarrow \begin{cases} Td \cdot Ucd_i + Tdu_i \cdot Dd_{i,j} + Tu_i^{in} \cdot Ucu_i + \min\left\{sum5, sum8, min1\right\}, & \begin{array}{l}\text{if } j \in V_k,\\ \quad k \in Ch_i\end{array} \\ Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + Sc_i + sum3, & \text{if } j = i \\ \infty, & \text{if } j \notin V_i \end{cases}$$

$$Cbn_{i,j} \leftarrow \begin{cases} Tdu_i \cdot Dd_{i,j} + (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + \min\left\{sum6, sum8, min1\right\}, & \begin{array}{l}\text{if } j \in V_k,\\ \quad k \in Ch_i\end{array} \\ (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + Sc_i + sum3\,, & \text{if } j = i \\ \min\left\{\begin{array}{l}\min\limits_{l \in V_i} Cbn_{i,l}{}^{(*)}, \\ \qquad Tdu_i \cdot Dd_{i,j} + (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + \min\left\{min2, min4\right\}\end{array}\right\}, & \text{if } j \notin V_i \end{cases}$$

$$Cbi_{i,j} \leftarrow \begin{cases} Td \cdot Ucd_i + Tdu_i \cdot Dd_{i,j} + Tu_i^{in} \cdot Ucu_i + Tu_i^{out} \cdot Ucu_i + sum7\,, & \begin{array}{l}\text{if } j \in V_k,\\ \quad k \in Ch_i\end{array} \\ \infty, & \text{if } j = i \\ \min\left\{\begin{array}{l}\min\limits_{l \in V_i} Cbi_{i,l}{}^{(*)}, \\ \qquad Td \cdot Ucd_i + Tdu_i \cdot Dd_{i,j} + (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + min3\end{array}\right\}, & \text{if } j \notin V_i \end{cases}$$

$$Cbo_{i,j} \leftarrow \begin{cases} Td \cdot Ucd_i + Tdu_i \cdot Dd_{i,j} + (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + \min\left\{sum8, min1\right\}, & \begin{array}{l}\text{if } j \in V_k,\\ \quad k \in Ch_i\end{array} \\ Td \cdot Ucd_i + (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + Sc_i + sum3\,, & \text{if } j = i \\ \min\left\{\begin{array}{l}\min\limits_{l \in V_i} Cbo_{i,l}{}^{(*)}, \\ \qquad Td \cdot Ucd_i + Tdu_i \cdot Dd_{i,j} + (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + min4\end{array}\right\}, & \text{if } j \notin V_i \end{cases}$$

$(*)$ The minimum value can be calculated once during the calculation of each $Cb?_{i,j},\ j\in V_i$, given that these values are calculated prior to calculating any $Cb?_{i,j},\ j\notin V_i$

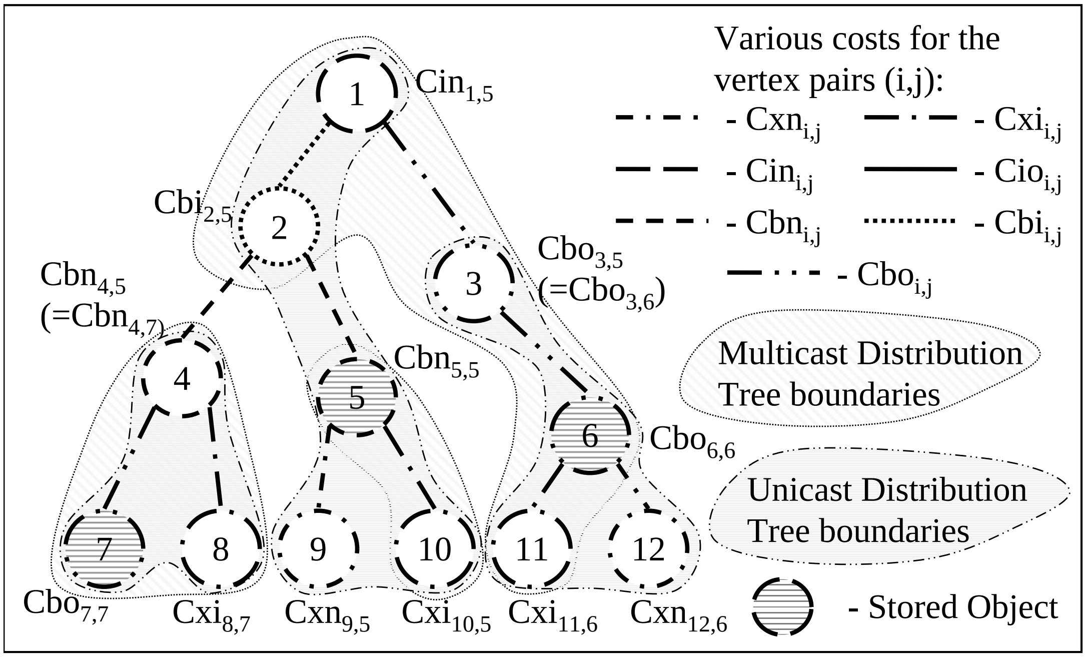

Fig. 5. An optimal allocation example, the actual scenarios and the distribution forest

where (various combinations of children scenarios):

$$sum1 = \sum_{k\in Ch_i} Cxn_{k,j}$$

$$sum2 = \sum_{k\in Ch_i} \min\{Cxn_{k,j}, Cxi_{k,j}\}$$

$$sum3 = \sum_{k\in Ch_i} \min\{Cxn_{k,j}, Cxi_{k,j}, Cbn_{k,j}, Cbi_{k,j}\}$$

$$sum4 = Cin_{k,j} + \sum_{l\in Ch_i, l\neq k} Cxn_{l,j}$$

$$sum5 = Cio_{k,j} + \sum_{l\in Ch_i, l\neq k} \min\{Cxn_{l,j}, Cxi_{l,j}\}$$

$$sum6 = Cbn_{k,j} + \sum_{l\in Ch_i, l\neq k} \min\{Cxn_{l,j}, Cbn_{l,j}\}$$

$$sum7 = \min\{Cbn_{k,j}, Cbi_{k,j}\} + \sum_{l\in Ch_i, l\neq k} \min\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\}$$

$$sum8 = Cbo_{k,j} + \sum_{l\in Ch_i, l\neq k} \min\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\}$$

$$min1 = \min\{Cbn_{k,j}, Cbi_{k,j}\} + \min_{\substack{m\in Ch_i,\\ m\neq k}} \left\{ Cbo_{m,j} + \sum_{\substack{l\in Ch_i,\\ l\neq k, l\neq m}} \min\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\} \right\}$$

$$min2 = \min_{k\in Ch_i} \left\{ Cbn_{k,j} + \sum_{l\in Ch_i, l\neq k} \min\{Cxn_{l,j}, Cbn_{l,j}\} \right\}$$

$$min3 = \min_{k\in Ch_i} \left\{ \min\{Cbn_{k,j}, Cbi_{k,j}\} + \sum_{l\in Ch_i, l\neq k} \min\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\} \right\}$$

$$min4 = \min_{k \in Ch_i} \left\{ Cbo_{k,j} + \sum_{l \in Ch_i, l \neq k} \min \left\{ Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j} \right\} \right\}$$

note: $sum1, sum2, sum3, sum4, sum5, sum6, sum7$ and $sum8$ equal 0, $min1, min2, min3$ and $min4$ equal ∞ if vertex i is a leaf ($Ch_i{=}\emptyset$). Also $sum1, sum4, sum6$ and $min2$ equal ∞ in case vertex i satisfies $Tdm_i > 0$

The cost of the optimal allocation in T is $min_{j \in V} Cin_{r,j}$.
The proof of optimality of the algorithm and a detailed explanation about the combinations are given in appendix II.

C. *Backtracking for content allocation*

While calculating the alternate costs for each vertex pair i, j, the algorithm remembers for each alternate cost (scenario), if a copy needs to be stored at vertex i and the relevant scenario of each child k that was used in the calculation (unless the scenario is $xn_{k,j}$ or $xi_{k,j}$, since it has no copy stored in its subtree). This is important for the backtracking phase, and allows accurate placement of the copies while backtracking.

The backtrack phase is recursive, starts at the root and ends at the leaves of T (can stop earlier if no child has a copy in V_k). For each vertex i, the algorithm determines the actual scenario in the optimal allocation, if a copy should be stored at i (will happen if (i, i) pair was selected for an actual scenario) and if it is necessary to keep advancing towards the leaves of T. The algorithm uses the backtrack information that was saved earlier..

Figure 5 demonstrates an optimal allocation, the various actual scenarios selected during the backtrack phase, and the distribution forest for that allocation.

D. *Computational complexity of HDT*

In the cost calculation phase, each vertex in the tree $i{\in}V$ the algorithm calculates up to $7{\cdot}N$ alternate costs. Each cost calculation requires $O(|Ch_i|{+}1)$. Therefore the total complexity of cost calculation for vertex i is $(7{\cdot}N){\cdot}O(|Ch_i|{+}1)$. The total complexity of the cost calculation phase for the entire tree is: $\sum_{i \in V}(7 \cdot N) \cdot O(|Ch_i| + 1)$.
The complexity of the backtrack phase for vertex i is $O(|Ch_i| + 1)$.
$|V|{=}N$ and the total number of children in the tree is $N{-}1$ (only the root r is not a child).
Therefore:

$$\begin{aligned} O_{HDT} = & \sum_{i \in V}(7 \cdot N + 1) \cdot O(|Ch_i| + 1) = O((7 \cdot N + 1) \cdot \sum_{i \in V}(|Ch_i| + 1)) = \\ & O((7 \cdot N + 1) \cdot (2 \cdot N - 1)) = O(N^2) \end{aligned}$$

The computational complexity of HDT is $O(N^2)$.

IX. The MX-HDT optimal algorithm

A variant of the original model in which the consumers connected to a vertex are served (distribution traffic) either by unicast or multicast but not both. The decision of which protocol to use is done by the network in order to reduce the total cost.

A. *Model changes*

The mutual exclusive hybrid model assumes that only one of unicast/muticast distribution traffic is provided to each vertex. The advantage of multicast over unicast is the aggregation of multiple streams into a single stream. On the other hand, unicast is much easier to control (in terms of flow control). We can say that the effective bandwidth requirements of a single unicast stream are smaller than a single multicast stream. Therefore, we modified the model as follows:

1) For each vertex i, both Tdu_i and Tdm_i are defined and satisfy: Tdm_i, is either Td or 0. $Tdu_i = q \cdot Tdm_i$. $0 < q < 1$. I.e. - unicast requires less bandwidth per stream.
2) A vertex can only be served either by unicast or by multicast. The selection is done automatically by the system in order to optimize the overall cost.

B. *Optimal solution properties*

Although the optimal solution properties presented in section IV are still valid here, the property described in lemma 1 is redefined to fit the current model this way: In the optimal allocation, in case of unicast distribution, if vertex i is served from vertex j, which satisfies $\min_{j \in \Phi} Dd_{i,j}$, and i is served through vertex k (i.e. $P_{i,j}=P_{i,k} \cup P_{k,j}$), then if another vertex l is served by unicast distribution through vertex k, l must also be served from j. The modification here implies that k itself may not be served by unicast distribution.

We also add an important property, to the specific case of mutual exclusive distribution traffic:

Lemma 4: In the optimal allocation, if there is multicast distribution traffic through vertex i, then vertex i must belong to a multicast distribution tree (this property is not correct for unicast distribution).

Proof: Suppose there is multicast distribution traffic through vertex i, and i is served by unicast distribution. In this case i belongs to both kinds of distribution trees, and this is a contradiction of the mutual exclusive traffic condition. Note: the opposite is not a contradiction, i.e. if there is unicast distribution traffic through vertex i (i.e. - another vertex k is served by unicast distribution through i), then vertex i may be served either by multicast or unicast distribution. ■

As suggested in corollary 1, and based on the above properties, the optimal allocation is composed of a subgraph of T which is a forest of distribution (multicast and/or unicast) subtrees. Each subtree is rooted at a vertex where a copy is located and its leaves are vertices were no copy of the object stored. An edge in T can be part of at most one multicast distribution tree. If a vertex belongs to a multicast distribution tree, it may still path unicast distribution through its edge.

C. *The algorithm*

As described in section V, the algorithm is performed in two phases, the cost calculation phase and the backtrack phase.

For the new problem we define a new tree, $T_{i,j}$, which is a subtree of T constructed of T_i (the subtree of T rooted at i), and the additional set of edges e_i (connects vertex i to its parent) and $P_{i,j}$ (the string that connects vertex i to j), in case $j \notin T_i$.

The new optimization problem is defined as follows: Find the optimal allocation and its alternate cost in $T_{i,j}$, given the following assumptions:

1) In case of unicast distribution, when a copy is located at vertex j, $j \in V$. If $j \in V_i$ vertex i is served by unicast distribution from j. If $j \notin V_i$ and vertex i is served by unicast distribution from outside T_i it is served from j. When $j \notin V_i$, since vertex j is not part of $T_{i,j}$ (just the path to it), its storage cost is ignored. Note: when $j \notin V_i$, i knows only the unicast distribution cost from copies located outside T_i. i may decide that storing a copy in an internal vertex l and be served from that copy is less expensive than being served from j.
2) There is or isn't a copy located inside T_i. Relevant for update/distribution multicast traffic. Also - the values of j must comply with this assumption.
3) There is or isn't a copy located outside T_i. Relevant for update/distribution multicast traffic. Also - the values of j must comply with this assumption.
4) In case of multicast distribution traffic, when there are copies outside T_i, are there consumers inside T_i that are connected to a multicast distribution tree through edge e_i.
5) In case of multicast distribution traffic, when there are copies inside T_i, are there consumers outside T_i that are connected to a multicast distribution tree through edge e_i.

Based on the above assumptions we define seven (7) scenarios that are possible for each vertex pair i, j. These scenarios cover all the possible external influences on the optimal allocation within T_i.

Figure 6. demonstrates the distribution forest with the different possible scenarios of the vertices and edges in T.

D. *The cost calculation phase*

For each vertex pair i, j the algorithm calculates for $T_{i,j}$, (vertex j is assumed to allocate a copy of the object), seven alternate costs, for the following seven possible scenarios:

$Cxn_{i,j}$- **C**ost of e**X**ternal only object allocation and **N**o incoming multicast distribution traffic. There is no copy located inside T_i ($i{\neq}r$) and there is no internal multicast demand. Edge e_i will only carry outgoing update traffic. Legal only when $j \notin V_i$.

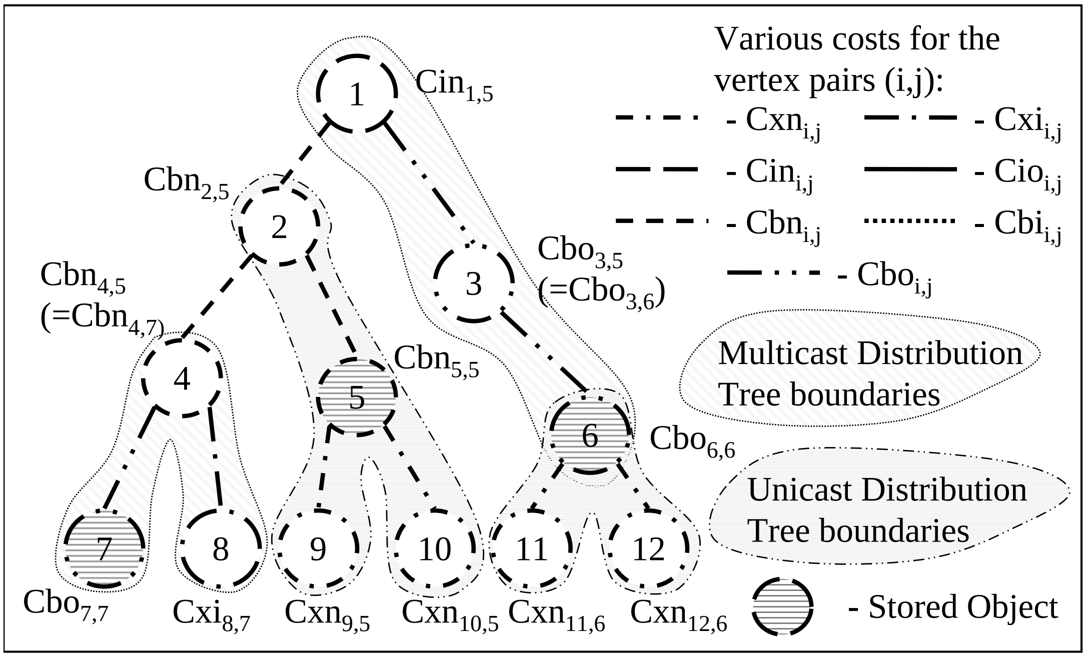

Fig. 6. An allocation, scenarios and distribution forest example

$Cxi_{i,j}$ - **C**ost of e**X**ternal only object allocation and **I**ncoming multicast distribution traffic. There is no copy located inside T_i ($i{\neq}r$) and there is an internal multicast demand. Edge e_i will carry incoming multicast distribution and outgoing update traffic. Legal only when $j \notin V_i$.

$Cin_{i,j}$ - **C**ost of **I**nternal only object allocation and **N**o outgoing multicast distribution traffic. All the copies of the object are located only inside T_i and there is no external multicast demand. Edge e_i will carry incoming update traffic. Legal only when $j \in V_i$.

$Cio_{i,j}$ - **C**ost of **I**nternal only object allocation and **O**utgoing multicast distribution traffic. All the copies of the object are located only inside T_i and there is an external multicast demand. Edge e_i will carry incoming update and outgoing multicast distribution traffic. Legal only when $j \in V_i$.

$Cbn_{i,j}$ - **C**ost of **B**oth sides object allocation and **N**o multicast distribution traffic. Copies are located both inside and outside T_i and there is no multicast demand through edge e_i. Edge e_i will carry both incoming and outgoing update traffic.

$Cbi_{i,j}$ - **C**ost of **B**oth sides object allocation and **I**ncoming multicast distribution traffic. Copies are located both inside and outside T_i and there is an internal multicast demand through edge e_i. Edge e_i will carry incoming multicast distribution and both incoming and outgoing update traffic.

$Cbo_{i,j}$ - **C**ost of **B**oth sides object allocation and **O**utgoing multicast distribution traffic. Copies are located both inside and outside T_i and there is an external multicast demand through edge e_i. Edge e_i will carry both incoming and outgoing update and outgoing multicast distribution traffic.

The result of the property described in lemma 4, is that for each scenario which contains multicast distribution through edge i (scenarios $xi_{i,j}, io_{i,j}, bi_{i,j}$ and $bo_{i,j}$), vertex i must be part of a multicast distribution tree and can't be served by unicast distribution. On the other hand, for each scenario which does not contain multicast distribution through edge i, vertex i may still belong to a multicast distribution tree (as a leaf) or may be served by unicast distribution.

The algorithm calculates the costs as follows:

$$Cxn_{i,j} \leftarrow \begin{cases} \infty, & \text{if } j \in V_i \\ Tdu_i \cdot Dd_{i,j} + Tu_i^{out} \cdot Ucu_i + sum1, & \text{if } j \notin V_i \end{cases}$$

$$Cxi_{i,j} \leftarrow \begin{cases} \infty, & \text{if } j \in V_i \\ Td \cdot Ucd_i + Tu_i^{out} \cdot Ucu_i + sum2, & \text{if } j \notin V_i \end{cases}$$

$$Cin_{i,j} \leftarrow \begin{cases} Tu_i^{in} \cdot Ucu_i + \min\left\{sum4, sum5, sum6, sum8, min1\right\}, & \text{if } j \in V_k, \\ & \quad k \in Ch_i \\ Tu_i^{in} \cdot Ucu_i + Sc_i + sum3, & \text{if } j = i \\ \infty, & \text{if } j \notin V_i \end{cases}$$

$$Cio_{i,j} \leftarrow \begin{cases} Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + \min\left\{sum5, sum8, min1\right\}, & \text{if } j \in V_k, \\ & \quad k \in Ch_i \\ Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + Sc_i + sum3, & \text{if } j = i \\ \infty, & \text{if } j \notin V_i \end{cases}$$

$$Cbn_{i,j} \leftarrow \begin{cases} (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + \min\left\{sum6, sum8, min1\right\}, & \text{if } j \in V_k, \\ & \quad k \in Ch_i \\ (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + Sc_i + sum3, & \text{if } j = i \\ \min\left\{\min\limits_{l \in V_i} Cbn_{i,l}{}^{(*)}, \right. & \\ \qquad \left. (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + \min\left\{min2, min4\right\}\right\}, & \text{if } j \notin V_i \end{cases}$$

$$Cbi_{i,j} \leftarrow \begin{cases} Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + Tu_i^{out} \cdot Ucu_i + sum7, & \text{if } j \in V_k, \\ & \quad k \in Ch_i \\ \infty, & \text{if } j = i \\ \min\left\{\min\limits_{l \in V_i} Cbi_{i,l}{}^{(*)}, \right. & \\ \qquad \left. Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + Tu_i^{out} \cdot Ucu_i + min3\right\}, & \text{if } j \notin V_i \end{cases}$$

$$Cbo_{i,j} \leftarrow \begin{cases} Td \cdot Ucd_i + (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + \min\left\{sum8, min1\right\}, & \text{if } j \in V_k, \\ & \quad k \in Ch_i \\ Td \cdot Ucd_i + (Tu_i^{in} + Tu_i^{out}) \cdot Ucu_i + Sc_i + sum3, & \text{if } j = i \\ \min\left\{\min\limits_{l \in V_i} Cbo_{i,l}{}^{(*)}, \right. & \\ \qquad \left. Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + Tu_i^{out} \cdot Ucu_i + min4\right\}, & \text{if } j \notin V_i \end{cases}$$

$(*)$ The minimum value should be calculated efficiently if $Cb?_{i,j}$, $j{\in}V_i$ are calculated prior to calculating any $Cb?_{i,j}$, $j{\notin}V_i$

where (various combinations of children scenarios):

$$sum1 = \sum_{k \in Ch_i} Cxn_{k,j}$$

$$sum2 = \sum_{k \in Ch_i} \min\left\{Cxn_{k,j}, Cxi_{k,j}\right\}$$

$$sum3 = \sum_{k \in Ch_i} \min\left\{Cxn_{k,j}, Cxi_{k,j}, Cbn_{k,j}, Cbi_{k,j}\right\}$$

$$sum4 = Tdu_i \cdot Dd_{i,j} + Cin_{k,j} + \sum_{l \in Ch_i, l \neq k} Cxn_{l,j}$$

$$sum5 = Cio_{k,j} + \sum_{l \in Ch_i, l \neq k} \min\left\{Cxn_{l,j}, Cxi_{l,j}\right\}$$

$$sum6 = Tdu_i \cdot Dd_{i,j} + Cbn_{k,j} + \sum_{l \in Ch_i, l \neq k} \min\left\{Cxn_{l,j}, Cbn_{l,j}\right\}$$

$$sum7 = \min\left\{Cbn_{k,j}, Cbi_{k,j}\right\} + \sum_{l \in Ch_i, l \neq k} \min\left\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\right\}$$

$$sum8 = Cbo_{k,j} + \sum_{l \in Ch_i, l \neq k} \min\left\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\right\}$$

$$min1 = \min\left\{Cbn_{k,j}, Cbi_{k,j}\right\} + \min_{m \in Ch_i, m \neq k}\left\{Cbo_{m,j} + \sum_{\substack{l \in Ch_i, \\ l \neq k, l \neq m}} \min\left\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\right\}\right\}$$

$$min2 = Tdu_i \cdot Dd_{i,j} + \min_{k \in Ch_i}\left\{Cbn_{k,j} + \sum_{l \in Ch_i, l \neq k} \min\left\{Cxn_{l,j}, Cbn_{l,j}\right\}\right\}$$

$$min3 = \min_{k \in Ch_i}\left\{\min\left\{Cbn_{k,j}, Cbi_{k,j}\right\} + \sum_{l \in Ch_i, l \neq k} \min\left\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\right\}\right\}$$

$$min4 = \min_{k \in Ch_i}\left\{Cbo_{k,j} + \sum_{l \in Ch_i, l \neq k} \min\left\{Cxn_{l,j}, Cxi_{l,j}, Cbn_{l,j}, Cbi_{l,j}\right\}\right\}$$

note: $sum1, sum2, sum3, sum4, sum5, sum6, sum7$ and $sum8$ equal 0, $min1, min2, min3$ and $min4$ equal ∞ if vertex i is a leaf ($Ch_i{=}\emptyset$).

The cost of the optimal allocation in T is $min_{j \in V} Cin_{r,j}$.

The proof of optimality of the algorithm and a detailed explanation about the combinations are given in appendix III.

E. Backtracking for content allocation

The backtracking phase of MX-HDT is similar to that of HDT (described in subsection VIII-C).

Figure 6 demonstrates an optimal allocation, the various actual scenarios selected during the backtrack phase, and the distribution forest for that allocation.

The pseudo code and backtrack details of the algorithm are given in appendix III.

F. Computational complexity of MX-HDT

The computational complexity of MX-HDT is $O(N^2)$
The calculation of the complexity of MX-HDT is the same as of HDT (described in subsection VIII-D).

X. Conclusions

In this work, we addressed a content location problem in overlay networks with update from multiple media sources and content distribution to users that employ multicast transport.

We developed optimal content allocation algorithms for tree networks with computational complexity of $O(N)$ for multicast only distribution traffic and $O(N^2)$ for hybrid multicast and unicast distribution traffic. The algorithms are recursive and are based on dynamic programming. These algorithms can easily be converted to distributed algorithms due to the independent calculations at each vertex (which are only based on information from its neighbors) and due to the hierarchical data flow

In addition to the optimal algorithms on tree networks, we presented an approximation algorithm for the multicast only distribution traffic problem on general graph. The approximation algorithm is based on the optimal algorithm for tree networks, while the extraction of trees from the general graph is done using Steiner tree approximation. We ran our algorithm on a synthetic Internet based network and compared it to various random placements. The placements generated by our approximation algorithm achieved significantly better overall costs (a ratio of 60% compared to the random placements costs).

Appendix I
The XMDT optimal algorithm

In this appendix we present, without a proof of optimality, the XMDT algorithm used in the MDG algorithm flow and its pseudo code.

In the algorithm description (subsection I-A) we provide only the cost calculation formulas, while in the pseudo code (subsection I-B) we provide both the cost calculation and backtrack phases.

A. The cost calculation phase

For each vertex i the algorithm calculates for T_i seven alternate costs, for the following possible scenarios:

Cxn_i - **C**ost of e**X**ternal only object allocation and **N**o incoming multicast distribution traffic. There is no copy located inside T_i ($i{\neq}r$) and there is no internal multicast demand. Edge e_i will only carry outgoing update traffic.

Cxi_i - **C**ost of e**X**ternal only object allocation and **I**ncoming multicast distribution traffic. There is no copy located inside T_i ($i{\neq}r$) and there is an internal multicast demand. Edge e_i will carry incoming multicast distribution and outgoing update traffic.

Cin_i - **C**ost of **I**nternal only object allocation and **N**o outgoing multicast distribution traffic. All the copies of the object are located only inside T_i and there is no external multicast demand. Edge e_i will carry incoming update traffic.

Cio_i - **C**ost of **I**nternal only object allocation and **O**utgoing multicast distribution traffic. All the copies of the object are located only inside T_i and there is an external multicast demand. Edge e_i will carry incoming update and outgoing multicast distribution traffic.

Cbn_i - **C**ost of **B**oth sides object allocation and **N**o multicast distribution traffic. Copies are located both inside and outside T_i and there is no multicast demand through edge e_i. Edge e_i will carry both incoming and outgoing update traffic.

Cbi_i - **C**ost of **B**oth sides object allocation and **I**ncoming multicast distribution traffic. Copies are located both inside and outside T_i and there is an internal multicast demand through edge e_i. Edge e_i will carry incoming multicast distribution and both incoming and outgoing update traffic.

Cbo_i - **C**ost of **B**oth sides object allocation and **O**utgoing multicast distribution traffic. Copies are located both inside and outside T_i and there is an external multicast demand through edge e_i. Edge e_i will carry both incoming and outgoing update and outgoing multicast distribution traffic.

The algorithm calculates the alternate costs as follows:

$$Cxn_i \leftarrow \begin{cases} Td \cdot Ucd_i + Tu_i^{out} \cdot Ucu_i + sum1, & \text{if } i \neq r \\ \infty, & \text{if } i = r \end{cases}$$

$$Cxi_i \leftarrow \begin{cases} Td \cdot Ucd_i + Tu_i^{out} \cdot Ucu_i + sum2, & \text{if } i \neq r \\ \infty, & \text{if } i = r \end{cases}$$

$$Cin_i \leftarrow \begin{cases} Tu_i^{in} \cdot Ucu_i + \min\{min1, min2, min3, min4, min5\}, & \text{if } i \neq r \\ \min\{min1, min2, min3, min4, min5\}, & \text{if } i = r \end{cases}$$

$$Cio_i \leftarrow \begin{cases} Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + \min\{min1, min3, min5\}, & \text{if } i \neq r \\ \infty, & \text{if } i = r \end{cases}$$

$$Cbn_i \leftarrow \begin{cases} Tu_i^{in} \cdot Ucu_i + Tu_i^{out} \cdot Ucu_i + \min\{min1, min2, min3\}, & \text{if } i \neq r \\ \infty, & \text{if } i = r \end{cases}$$

$$Cbi_i \leftarrow \begin{cases} Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + Tu_i^{out} \cdot Ucu_i + sum3, & \text{if } i \neq r \;\&\; Ch_i \neq \emptyset \\ \infty, & \text{if } i = r \;\|\; Ch_i = \emptyset \end{cases}$$

$$Cbo_i \leftarrow \begin{cases} Td \cdot Ucd_i + \left(Tu_i^{in} + Tu_i^{out}\right) \cdot Ucu_i + \min\{min1, min3\}, & \text{if } i \neq r \\ \infty, & \text{if } i = r \end{cases}$$

where (various combinations of children scenarios):

$$sum1 = \sum_{c \in Ch_i} Cxn_c$$

$$sum2 = \sum_{c \in Ch_i} \min\{Cxn_c, Cxi_c\}$$

$$sum3 = \sum_{c \in Ch_i} \min\{Cxn_c, Cxi_c, Cbn_c, Cbi_c\}$$

$$\begin{aligned}
min1 &= Sc_i + sum3 \\
min2 &= \min_{c \in Ch_i} \left\{ Cbn_c + \sum_{k \in Ch_i, k \neq c} \min\left\{Cxn_k, Cbn_k\right\} \right\} \\
min3 &= \min_{c \in Ch_i} \left\{ Cbo_c + \sum_{k \in Ch_i, k \neq c} \min\left\{Cxn_k, Cxi_k, Cbn_k, Cbi_k\right\} \right\} \\
min4 &= \min_{c \in Ch_i} \left\{ Cin_c + \sum_{k \in Ch_i, k \neq c} Cxn_k \right\} \\
min5 &= \min_{c \in Ch_i} \left\{ Cio_c + \sum_{k \in Ch_i, k \neq c} \min\left\{Cxn_k, Cxi_k\right\} \right\}
\end{aligned}$$

note: $sum1, sum2$ and $sum3$ equal 0, $min2, min3, min4$ and $min5$ equal ∞ if vertex i is a leaf (Ch_i=$\emptyset$). Also $sum1, min2$ and $min4$ equal ∞ in case vertex i satisfies $Tdm_i > 0$

The cost of the optimal allocation in T is Cin_r.

B. Partial Pseudo code of XMDT

We assume that the vertices are ordered by breadth first ordering. Vertex 1 is the root and n must be a leaf.
We also assume the ∞ is the maximal number that exists in the computer.
Variables starting with BT are used for the backtrack process, and store a vertex number or the cost/vertex data.

The algorithm is performed in two phases. The first one is for calculating the optimal cost and the backtrack info for later.
We only provide part of the first phase due to the large number of scenarios and combinations. The main idea in the pseudo code provided is to show the combinations of children scenarios and to demonstrate the generation of backtrack info.

Cost calculation phase

```
for $i = n, n-1, n-2, ..., 2, 1$ do
    if $Ch_i = \emptyset$ then /* a leaf */
        if $Tdm_i = 0$ then /* No multicast demand */
            $Cxn_i \leftarrow \infty$
        else
            $Cxn_i \leftarrow Tu_i^{out} \cdot Ucu_i$
        end if
        $Cxi_i \leftarrow Td \cdot Ucd_i + Tu_i^{out} \cdot Ucu_i$
        $Cin_i \leftarrow Tu_i^{in} \cdot Ucu_i + Sc_i$ ; $BT\text{-}Cin_i \leftarrow (i, ''local'')$
        $Cio_i \leftarrow Td \cdot Ucd_i + Tu_i^{in} \cdot Ucu_i + Sc_i$ ; $BT\text{-}Cio_i \leftarrow (i, ''local'')$
        $Cbn_i \leftarrow \left(Tu_i^{in} + Tu_i^{out}\right) \cdot Ucu_i + Sc_i$ ; $BT\text{-}Cbn_i \leftarrow (i, ''local'')$
        $Cbi_i \leftarrow \infty$ ; $BT\text{-}Cbi_i \leftarrow \emptyset$
        $Cbo_i \leftarrow Td \cdot Ucd_i + \left(Tu_i^{in} + Tu_i^{out}\right) \cdot Ucu_i + Sc_i$ ; $BT\text{-}Cbo_i \leftarrow (i, ''local'')$
    else
        /* calculate sum1, sum2, sum3 (and sum4) */
        $sum1 \leftarrow 0$ ; $sum2 \leftarrow 0$
        $sum3 \leftarrow 0$ ; $BT\text{-}sum3 \leftarrow \emptyset$
        $sum4 \leftarrow 0$ ; $BT\text{-}sum4 \leftarrow \emptyset$
        foreach $c$ in $Ch_i$ do
            $sum1 \leftarrow sum1 + Cxn_c$
            /* update $\min\{Cxn_c, Cxi_c\}$ */
            $Cminx_c \leftarrow Cxn_c$
            /* update $\min\{Cxn_c, Cbn_c\}$ */
            $Cminn_c \leftarrow Cxn_c$ ; $Cminn_{c,type} \leftarrow ”none”$
            /* update $\min\{Cxn_c, Cxi_c, Cbn_c, Cbi_c\}$ */
            $Cmin_c \leftarrow Cxn_c$ ; $Cmin_{c,type} \leftarrow ”none”$
            if ($Cxi_c < Cmin_c$) then
                $Cminx_c \leftarrow Cxi_c$
                $Cmin_c \leftarrow Cxi_c$ ; $Cmin_{c,type} \leftarrow ”none”$
            end if
            if ($Cbn_c < Cmin_c$) then
                $Cminn_c \leftarrow Cbn_c$ ; $Cminn_{c,type} \leftarrow ”bn”$
                $Cmin_c \leftarrow Cbn_c$ ; $Cmin_{c,type} \leftarrow ”bn”$
            else if ($Cbn_c < Cminn_c$) then
                $Cminn_c \leftarrow Cbn_c$ ; $Cminn_{c,type} \leftarrow ”bn”$
            end if
            if ($Cbi_c < Cmin_c$) then
                $Cmin_c \leftarrow Cbi_c$ ; $Cmin_{c,type} \leftarrow ”bi”$
            end if
            $sum2 \leftarrow sum2 + Cminx_c$
            $sum3 \leftarrow sum1 + Cmin_c$ ; $BT\text{-}sum3 \leftarrow BT\text{-}sum3 \cup (c, Cmin_{c,type})$
            $sum4 \leftarrow sum4 + Cminn_c$ ; $BT\text{-}sum4 \leftarrow BT\text{-}sum4 \cup (c, Cminn_{c,type})$
        end do
        /* calculate min1, min2, min3, min4, min5 */
        $min1 \leftarrow Sc_i + sum3$ ; $BT\text{-}min1 \leftarrow BT\text{-}sum3 \cup (i, ”local”)$
        $min2 \leftarrow \infty$ ; $BT\text{-}min2 \leftarrow \emptyset$
        $min3 \leftarrow \infty$ ; $BT\text{-}min3 \leftarrow \emptyset$
        $min4 \leftarrow \infty$ ; $BT\text{-}min4 \leftarrow \emptyset$
        $min5 \leftarrow \infty$ ; $BT\text{-}min5 \leftarrow \emptyset$
        foreach $c$ in $Ch_i$ do
            $tmp \leftarrow Cbn_c + sum4 - Cminn_c$
            if ($tmp < min2$ & $Tdm_i > 0$) then
                $min2 \leftarrow tmp$ ; $BT\text{-}min2 \leftarrow (c, ”bn”) \cup \Big(BT\text{-}sum4 \setminus (c, Cminn_{c,type})\Big)$
            end if
            $tmp \leftarrow Cbo_c + sum3 - Cmin_c$
            if ($tmp < min3$) then
                $min3 \leftarrow tmp$ ; $BT\text{-}min3 \leftarrow (c, ”bo”) \cup \Big(BT\text{-}sum3 \setminus (c, Cmin_{c,type})\Big)$
            end if
            $tmp \leftarrow Cin_c + sum1 - Cxn_c$
            if ($tmp < min4$ & $Tdm_i > 0$) then
                $min4 \leftarrow tmp$ ; $BT\text{-}min4 \leftarrow (c, ”in”)$
            end if
            $tmp \leftarrow Cio_c + sum2 - Cminx_c$
            if ($tmp < min5$) then
                $min5 \leftarrow tmp$ ; $BT\text{-}min5 \leftarrow (c, ”io”)$
            end if
        end do
        /* calculate the optimal costs */
        if $i \neq 1$ then /* not root */
            if $Tdm_i = 0$ then /* No multicast demand */
                $Cxn_i \leftarrow \infty$
            else
                $Cxn_i \leftarrow Td \cdot Ucd_i + Tu_i^{out} \cdot Ucu_i + sum1$
```

```
            end if
            $Cxi_i \leftarrow Td \cdot Ucd_i + Tu_i^{out} \cdot Ucu_i + sum2$
            ...
            $Cbi_i \leftarrow Td \cdot Ucd_i + \left(Tu_i^{in} + Tu_i^{out}\right) \cdot Ucu_i + sum3$ ; $BT\text{-}Cbi_i \leftarrow BT\text{-}sum3$
        end if
        /* calculate optimal $Cin_i$ cost and BT data */
        $ind = argmin\{min1, min2, min3, min4, min5\}$
        $Cin_i \leftarrow Tu_i^{in} \cdot Ucu_i + min_{ind}$ ; $BT\text{-}Cin_i \leftarrow BT\text{-}min_{ind}$
    end if
end do
```

The optimal cost is Cin_1.

Backtrack phase

The backtrack phase for allocation of copies is recursive and can easily be described using a recursive function. The recursion starts by calling **allocate**$(1, ”in”)$.

```
proc **allocate** $(i, type)$ {
    if $type = ”io”$ then
        foreach $(c, ctype)$ in $BT\text{-}Cio_i$ do
            call **allocate**$(c, ctype)$
        end do
    else if $type = ”in”$ then
        foreach $(c, ctype)$ in $BT\text{-}Cin_i$ do
            call **allocate**$(c, ctype)$
        end do
    else if $type = ”bn”$ then
        foreach $(c, ctype)$ in $BT\text{-}Cbn_i$ do
            call **allocate**$(c, ctype)$
        end do
    else if $type = ”bo”$ then
        foreach $(c, ctype)$ in $BT\text{-}Cbo_i$ do
            call **allocate**$(c, ctype)$
        end do
    else if $type = ”bi”$ then
        foreach $(c, ctype)$ in $BT\text{-}Cbi_i$ do
            call **allocate**$(c, ctype)$
        end do
    else if $type = ”local”$ then
        allocate a copy at $i$
    end if
    return
}
```

APPENDIX II
MORE ON HDT - PROOF AND PSEUDO-CODE

This appendix contains additional details on the HDT algorithm that were not included in section VIII.

A. Proof of HDT Optimality

The proof is based on induction. Lemma 5 is the induction base.

Lemma 5: For all scenarios, and for all vertices $j \in V$ the algorithm optimally allocates the object in $T_{i,j}$, when i is a leaf of T

Proof: According to the definition of the new optimization problem, either one of the following possibilities holds.

1) $j{=}i$ (no string is connected), the algorithm allocates the object at vertex i. There is a copy located at T_i therefore $Cxn_{i,i}$ and $Cxi_{i,i}$ can't exist (set to ∞). $Cbi_{i,i}$ is also impossible, since there is no need for incoming multicast distribution traffic when i stores a copy. The valid possible scenarios are $in_{i,i}, io_{i,i}, bn_{i,i}$ and $bo_{i,i}$ which differ in the assumptions that i is or isn't the only vertex which stores a copy and there is or isn't outgoing multicast distribution traffic from T_i. Since i stores a copy, there must be incoming update traffic through edge e_i. The optimal cost is constructed from the storage cost, the incoming update traffic through e_i, and the additional outgoing update and/or distribution traffic through $e_i$when appropriate ($Cio_{i,i}$, $Cbn_{i,i}$, $Cbo_{i,i}$).
2) $j{\neq}i$ ($j{\notin}V_i$). A string $P_{j,i}$ is connected to T_i. According to the definition of the problem, there's a copy of the object located at vertex j and if i is served from outside T_i, i is served (unicast distribution) from that vertex j. There is a copy located outside T_i therefore $Cin_{i,j}$ and $Cio_{i,j}$ can't exist (set to ∞). The scenarios $xn_{i,j}$ and $xi_{i,j}$ assume that it costs more to store an object in i (and to keep it updated) than to be served from j. The scenario $xn_{i,j}$ is valid only if $Tdm_i = 0$ (else $Cxn_{i,j} = \infty$ since $sum1 = \infty$ when $Tdm_i > 0$). The optimal cost is the outgoing update traffic through e_i and the incoming unicast (from j to i) and/or multicast distribution traffic through e_i. The scenarios $bn_{i,j}, bi_{i,j}$ and $bo_{i,j}$ assume that it is cheaper to store a copy in i although j has a copy. Since i stores a copy, the costs calculated for $bn_{i,i}, bi_{i,i}$ and $bo_{i,i}$ must be used accordingly (achieved by setting $min2, min3$ and $min4$ to ∞ for a leaf vertex).

■

Lemma 6 constructs the induction step for the recursive proof of optimality.

Lemma 6: Assume that the algorithm optimally allocates the object to servers in every subtree rooted at vertex c which is a child of i (T_c, $c{\in}Ch_i$) for all scenarios and for all vertices $j{\in}V$, then the algorithm optimally allocates the object in T_i for all the scenarios and for all vertices $j{\in}V$.

Proof: According to the definition of the new optimization problem, either one of the following possibilities holds.

1) $j{=}i$ (no string is connected), the algorithm allocates the object at vertex i. There is a copy located at T_i therefore $Cxn_{i,i}$ and $Cxi_{i,i}$ can't exist (set to ∞). $Cbi_{i,i}$ is also impossible, since there is no need for incoming multicast distribution traffic when i stores a copy. The valid possible scenarios are $in_{i,i}, io_{i,i}, bn_{i,i}$ and $bo_{i,i}$ which differ in the assumptions that i is or isn't the only vertex which stores a copy and there is or isn't outgoing multicast distribution traffic from T_i. The vertices in each subtree T_k, $k{\in}Ch_i$ may be served by unicast and/or multicast either from vertex i or from copies located internally in the subtree. (The minimum of the following legal scenarios for each $k{\in}Ch_i$: $Cxn_{k,j}, Cxi_{k,j}, Cbn_{k,j}, Cbi_{k,j} \Rightarrow sum3$). Since i stores a copy, there must be incoming update traffic through edge e_i. The optimal cost is constructed from the storage cost at i, the optimal costs calculated by the children ($sum3$), the incoming update traffic through e_i, and the additional outgoing update and/or distribution traffic through e_i when appropriate ($Cio_{i,i}$, $Cbn_{i,i}$, $Cbo_{i,i}$).
2) $j{\in}V_k$, $k{\in}Ch_i$ (no string is connected), the algorithm allocates a copy of the object at vertex j. There is a copy located at T_i therefore $Cxn_{i,j}$ and $Cxi_{i,j}$ can't exist (set to ∞), and there must be incoming update traffic through edge e_i.

   For each of the scenarios where no copy of the object is allocated outside T_i ($in_{i,j}$ and $io_{i,j}$) or copies are also located outside T_i ($bn_{i,j}, bi_{i,j}$ and $bo_{i,j}$), one the following children combinations hold:

   a) There are copies allocated only inside T_k (at least at vertex j), and there is no outgoing multicast distribution traffic from T_k (scenario $in_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ must fulfil scenario $xn_{l,j}$ ($sum4$).

      This combination is valid only when $Tdm_i = 0$ (else $sum4$ is set to ∞) and only for scenario $in_{i,j}$, since there can be no outgoing multicast distribution traffic from T_i.

   b) There are copies allocated only inside T_k (at least at vertex j), and there is outgoing multicast distribution traffic from T_k (scenario $io_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ must not allocate a copy in T_l (scenarios $xn_{l,j}$ or $xi_{l,j}$) ($sum5$).

      This combination is valid only for scenarios $in_{i,j}$ and $io_{i,j}$.

   c) There are copies allocated both inside T_k (at least at vertex j) and outside T_k, but there is no multicast distribution through edge e_k (scenario $bn_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate

a copy in T_l but must not be served by incoming multicast distribution through i (through edge e_l) (scenarios $xn_{l,j}$ or $bn_{l,j}$) ($sum6$).
This combination is valid only when $Tdm_i = 0$ (else $sum6$ is set to ∞) and only for scenarios $in_{i,j}$ and $bn_{i,j}$, since there can be no outgoing multicast distribution traffic from T_i.

d) This combination is only valid for scenario $bi_{i,j}$. In this scenario there is incoming multicast distribution through edges e_i.
There are copies allocated both inside T_k (at least at vertex j) and outside T_k, and e_k. Vertex i is part of a multicast distribution tree, through edge e_i (one of scenarios $bn_{k,j}, bi_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, and may be served by incoming multicast distribution traffic through i (and e_l) ($sum7$).

e) There are copies allocated both inside T_k (at least at vertex j) and outside T_k, and there is outgoing multicast distribution through edge e_k (scenario $bo_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, and may be served by incoming multicast distribution traffic through vertex i (and edge e_k) ($sum8$).
This combination is not valid for scenario $bi_{i,j}$, due to lemma 2.

f) There are copies allocated both inside T_k (at least at vertex j) and outside T_k (scenarios $bn_{k,j}$ or $bi_{k,j}$), but there is outgoing multicast distribution through some edge e_l, $l{\in}Ch_i$, $l{\neq}k$ (scenario $bo_{l,j}$). In this case the rest of the children $m{\in}Ch_i$, $m{\neq}k$, $m{\neq}l$ may allocate a copy in T_m, and may be served by incoming multicast distribution traffic through i (and edge e_l) ($min1$).
This combination is not valid for scenario $bi_{i,j}$, due to lemma 2.

For each such scenario the minimal (optimal) valid children combination is selected, and the optimal cost is constructed from that combination cost (min between $sum4$, $sum5$, $sum6$, $sum7$, $sum8$ or $min1$), the unicast distribution cost from j to i, the cost of update traffic through e_i (incoming only for $Cin_{i,j}, Cio_{i,j}$, both incoming and outgoing for $Cbn_{i,j}, Cbi_{i,j}, Cbo_{i,j}$) and the cost of multicast distribution traffic through e_i (outgoing for $Cio_{i,j}, Cbo_{i,j}$, incoming for $Cbi_{i,j}$).

3) $j{\notin}V_i$. A string $P_{j,i}$ is connected to T_i. According to the definition of the problem, there's a copy of the object located at vertex j and if i is served from outside T_i, i is served (unicast distribution) from that vertex. There is a copy located outside T_i therefore $Cin_{i,j}$ and $Cio_{i,j}$ can't exist (set to ∞), and there must be outgoing update traffic through edge e_i.
For the scenario $xn_{i,j}$, which means that no copy is located in T_i and there is no incoming multicast distribution traffic to T_i, only one children combination is possible - all the children $k{\in}Ch_i$ must fulfil scenario $xn_{k,j}$ ($sum1$). This scenario is valid only when $Tdm_i = 0$ (else $sum1$ is set to ∞).
For the scenario $xi_{i,j}$, which means that no copy is located in T_i but there is incoming multicast distribution traffic to T_i, all the children $k{\in}Ch_i$ must fulfil scenario $xn_{k,j}$ or $xi_{k,j}$ ($sum2$).
For each of the scenarios where copies are also located inside T_i ($bn_{i,j}, bi_{i,j}$ and $bo_{i,j}$), the unicast distribution source of vertex i, may be an internal vertex (i.e. i may not be served from j by unicast distribution). In such a case, the unicast distribution source l,$l{\in}V_i$ and the corresponding cost $Cb?_{i,l}$ will be used (expression $\min_{l \in V_i} Cb?_{i,l}$). In case i is served from j by unicast distribution, it must not store an object and one the following children combinations hold:

a) This combination is only valid for scenario $bn_{i,j}$. In this scenario there is no multicast distribution through edge e_i. There is at least one vertex k, $k{\in}Ch_i$ which has at least one copy allocated in T_k, and does not produce outgoing multicast distribution traffic (scenario $bn_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, but must not be served by incoming multicast distribution through i (and edge e_l) (scenarios $xn_{l,j}$ or $bn_{l,j}$) ($min2$).
This combination is valid only when $Tdm_i = 0$ (else $min2$ is set to ∞)

b) This combination is only valid for scenario $bi_{i,j}$. In this scenario there is incoming multicast distribution through edge e_i. There is at least one vertex k, $k{\in}Ch_i$ which has at least one copy allocated in T_k, and may or may not be served by incoming multicast distribution through i (scenarios $bn_{k,j}$ or $bi_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, and may be served by incoming multicast distribution traffic through i (and e_l) ($min3$).

c) There is outgoing multicast distribution through some vertex k, $k{\in}Ch_i$ (scenario $bo_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, and may be served by incoming multicast distribution traffic through vertex i (and k) ($min4$).

This combination is not valid for scenario $bi_{i,j}$, due to lemma 2.

For each such scenario the minimal (optimal) valid children combination is selected, and the optimal cost is constructed from that combination cost (min between $sum1, sum2, min2, min3$ or $min4$), the unicast distribution cost from j to i, the cost of update traffic through e_i (outgoing only for $Cxn_{i,j}, Cxi_{i,j}$, both incoming and outgoing for $Cbn_{i,j}, Cbi_{i,j}, Cbo_{i,j}$) and the cost of multicast distribution traffic through e_i (outgoing for $Cbo_{i,j}$, incoming for $Cxi_{i,j}$, $Cbi_{i,j}$).

■

Theorem 1: When the algorithm ends, $min_{j\in V}Cin_{r,j}$ holds the optimal allocation cost and the allocation of copies is optimal.

Proof: The proof is conducted by the induction where lemma 5 is the base and lemma 6 is the step. For each $j{\in}V$, $Cin_{r,j}$ represents an optimal allocation of the objects where r is served from j. The minimal $Cin_{r,j}$ is the optimal cost of the original optimization problem. In addition, the costs $Cxn_{r,j}, Cxi_{r,j}, Cbn_{r,j}, Cbi_{r,j}$ and $Cbo_{r,j}$ are illegal since there can't be copies allocated outside $T_r \equiv T$ and $Cio_{r,j}$ is illegal there can't be distribution traffic through r outside of T. ■

B. Partial Pseudo code of HDT

We assume that the vertices are ordered by breadth first ordering. Vertex 1 is the root and n must be a leaf.
We also assume the ∞ is the maximal number that exists in the computer.
Variables starting with BT are used for the backtrack process, and store a vertex number or the cost/vertex data.

The algorithm is performed in two phases. The first one is for calculating the optimal cost and the backtrack info for later.
We only provide part of the first phase due to the large number of scenarios and combinations. The main idea in the pseudo code provided is to show the combinations of children scenarios and to demonstrate the generation of backtrack info.

Cost calculation phase

```
for $i = n, n-1, n-2, ..., 2, 1$ do
    /* calculate costs for $i, i$ */
    $sum3 \leftarrow 0$ ; $BT\text{-}sum3 \leftarrow \emptyset$
    foreach $k$ in $Ch_i$ do
        if $Cxn_{k,i} <= Cxi_{k,i}$ & $Cxn_{k,i} <= Cbn_{k,i}$ & $Cxn_{k,i} <= Cbi_{k,i}$ then
            $sum3 \leftarrow sum3 + Cxn_{k,i}$
        else if $Cxi_{k,i} <= Cbn_{k,i}$ & $Cxi_{k,i} <= Cbi_{k,i}$ then
            $sum3 \leftarrow sum3 + Cxi_{k,i}$
        else if $Cbn_{k,i} <= Cbi_{k,i}$ then
            $sum3 \leftarrow sum3 + Cbn_{k,i}$ ; $BT\text{-}sum3 \leftarrow BT\text{-}sum3 \cup (k, i, ''bn'')$
        else
            $sum3 \leftarrow sum3 + Cbi_{k,i}$ ; $BT\text{-}sum3 \leftarrow BT\text{-}sum3 \cup (k, i, ''bi'')$
        end if
    end do
    $Cxn_{i,i} \leftarrow \infty$ ; $Cxi_{i,i} \leftarrow \infty$ ; $Cbi_{i,i} \leftarrow \infty$
    $Cin_{i,i} \leftarrow Tu_i^{in} \cdot Ucu_i + Sc_i + sum3$ ; $BT\text{-}Cin_{i,i} \leftarrow BT\text{-}sum3 \cup (i, i, ''local'')$
    $Cio_{i,i} \leftarrow Td \cdot Ucd_i + Cin_{i,i}$ ; $BT\text{-}Cio_{i,i} \leftarrow BT\text{-}Cin_{i,i}$
    $Cbn_{i,i} \leftarrow Cin_{i,i} + Tu_i^{out} \cdot Ucu_i$ ; $BT\text{-}Cbn_{i,i} \leftarrow BT\text{-}Cin_{i,i}$
    $Cbo_{i,i} \leftarrow Cio_{i,i} + Tu_i^{out} \cdot Ucu_i$ ; $BT\text{-}Cbo_{i,i} \leftarrow BT\text{-}Cio_{i,i}$
    /* update $\min_{l\in V_i} Cb?_{i,l}$ */
    $j_{minn} \leftarrow i$ ; $j_{mini} \leftarrow i$ ; $j_{mino} \leftarrow i$
    /* calculate costs for $i, j$ where $j \in V_k, k \in Ch_i$ */
    foreach $k$ in $Ch_i$ do
        foreach $j$ in $V_k$ do
```

```
$sum4 \leftarrow Cin_{k,j}$ ; $BT\text{-}sum4 \leftarrow (k, j, ''in'')$
$sum5 \leftarrow Cio_{k,j}$ ; $BT\text{-}sum5 \leftarrow (k, j, ''io'')$
$sum6 \leftarrow Cbn_{k,j}$ ; $BT\text{-}sum6 \leftarrow (k, j, ''bn'')$
if $Cbn_{k,j} <= Cbi_{k,j}$ then
    $sum7 \leftarrow Cbn_{k,j}$ ; $BT\text{-}sum7 \leftarrow (k, j, ''bn'')$
else
    $sum7 \leftarrow Cbi_{k,j}$ ; $BT\text{-}sum7 \leftarrow (k, j, ''bi'')$
end if
$sum8 \leftarrow Cbo_{k,j}$ ; $BT\text{-}sum8 \leftarrow (k, j, ''bo'')$
foreach $l$ in $Ch_i \setminus k$ do
    $sum4 \leftarrow sum4 + Cxn_{l,j}$
    if $Cxn_{l,j} <= Cxi_{l,j}$ then
        $sum5 \leftarrow sum5 + Cxn_{l,j}$
    else
        $sum5 \leftarrow sum5 + Cxi_{l,j}$
    end if
    if $Cxn_{l,j} <= Cbn_{l,j}$ then
        $sum6 \leftarrow sum6 + Cxn_{l,j}$
    else
        $sum6 \leftarrow sum6 + Cbn_{l,j}$ ; $BT\text{-}sum6 \leftarrow BT\text{-}sum6 \cup (l, j, ''bn'')$
    end if
    $Cmin_l \leftarrow Cxn_{l,j}$ ; $Cmin_{l,type} \leftarrow ''none''$
    if $Cxi_{l,j} < Cmin_l$ then
        $Cmin_l \leftarrow Cxi_{l,j}$ ; $Cmin_{l,type} \leftarrow ''none''$
    end if
    if $Cbn_{l,j} < Cmin_l$ then
        $Cmin_l \leftarrow Cbn_{l,j}$ ; $Cmin_{l,type} \leftarrow ''bn''$
    end if
    if $Cbi_{l,j} < Cmin_l$ then
        $Cmin_l \leftarrow Cbi_{l,j}$ ; $Cmin_{l,type} \leftarrow ''bi''$
    end if
    $sum7 \leftarrow sum7 + Cmin_l$ ; $BT\text{-}sum7 \leftarrow BT\text{-}sum7 \cup (l, j, Cmin_{l,type})$
    $sum8 \leftarrow sum8 + Cmin_l$ ; $BT\text{-}sum8 \leftarrow BT\text{-}sum8 \cup (l, j, Cmin_{l,type})$
end do
/* calculate min1 (derived from sum7) */
$min1 \leftarrow \infty$ ; $BT\text{-}min1 \leftarrow \emptyset$
foreach $l$ in $Ch_i \setminus k$ do
    $tmp \leftarrow Cbo_{l,j} + sum7 - Cmin_l$
    if ($tmp < min1$) then
        $min1 \leftarrow tmp$ ; $BT\text{-}min1 \leftarrow (l, j, ''bo'') \cup \Big(BT\text{-}sum7 \setminus (l, j, Cmin_{l,type})\Big)$
    end if
end do
if $Tdm_i > 0$ then /* There is multicast demand */
    $sum4 \leftarrow \infty$ ; $BT\text{-}sum4 \leftarrow \emptyset$ ; $sum6 \leftarrow \infty$ ; $BT\text{-}sum6 \leftarrow \emptyset$
end if
$Cxn_{i,j} \leftarrow \infty$ ; $Cxi_{i,j} \leftarrow \infty$
/* calculate optimal $Cin_{i,j}$ cost and BT data */
$min_{val} = min\{sum4, sum5, sum6, sum8, min1\}$
$min_{type} = argmin\{sum4, sum5, sum6, sum8, min1\}$
$Cin_{i,j} \leftarrow Tdu_i \cdot Dd_{i,j} + Tu_i^{in} \cdot Ucu_i + min_{val}$ ; $BT\text{-}Cin_{i,j} \leftarrow BT\text{-}min_{type}$
...
```

```
        $Cbi_{i,j} \leftarrow Td \cdot Ucd_i + Tdu_i \cdot Dd_{i,j} + \left(Tu_i^{in} + Tu_i^{out}\right) \cdot Ucu_i + sum7$ ; $BT\text{-}Cbi_{i,j} \leftarrow BT\text{-}sum7$
        /* update $\min_{l \in V_i} Cb?_{i,l}$ */
        if $Cbn_{i,j} < Cbn_{i,j_{minn}}$ then
            $j_{minn} \leftarrow j$
        end if
        if $Cbi_{i,j} < Cbi_{i,j_{min}}$ then
            $j_{mini} \leftarrow j$
        end if
        if $Cbo_{i,j} < Cbo_{i,j_{mino}}$ then
            $j_{mino} \leftarrow j$
        end if
    end do
end do
/* calculate costs for $i, j$ where $j \notin V_i$ */
foreach $j$ in $V \setminus V_i$ do
    $sum1 \leftarrow 0$ ; $sum2 \leftarrow 0$
    /* min2, min3, min4 are derived from sum6, sum8, sum8 */
    $sum6 \leftarrow 0$ ; $BT\text{-}sum6 \leftarrow \emptyset$ ; $sum8 \leftarrow 0$ ; $BT\text{-}sum8 \leftarrow \emptyset$
    foreach $k$ in $Ch_i$ do
        $sum1 \leftarrow sum1 + Cxn_{k,j}$
        if $Cxn_{k,j} <= Cxi_{k,j}$ then
            $sum2 \leftarrow sum2 + Cxn_{k,j}$
        else
            $sum2 \leftarrow sum2 + Cxi_{k,j}$
        end if
        $Cminn_k \leftarrow Cxn_{k,j}$ ; $Cmin_{k,type} \leftarrow ''none''$
        if $Cbn_{k,j} < Cminn_k$ then
            $Cminn_k \leftarrow Cbi_{k,j}$ ; $Cminn_{k,type} \leftarrow ''bn''$
        end if
        $sum6 \leftarrow sum6 + Cminn_k$ ; $BT\text{-}sum6 \leftarrow BT\text{-}sum6 \cup (k, j, Cminn_{k,type})$
        $Cminb_k \leftarrow Cbn_{k,j}$ ; $Cminb_{k,type} \leftarrow ''bn''$
        if $Cbi_{k,j} < Cminb_k$ then
            $Cminb_k \leftarrow Cbi_{k,j}$ ; $Cminb_{k,type} \leftarrow ''bi''$
        end if
        $Cmin_k \leftarrow Cxn_{k,j}$ ; $Cmin_{k,type} \leftarrow ''none''$
        if $Cxi_{k,j} < Cmin_k$ then
            $Cmin_k \leftarrow Cxi_{k,j}$ ; $Cmin_{k,type} \leftarrow ''none''$
        end if
        if $Cbn_{k,j} < Cmin_k$ then
            $Cmin_k \leftarrow Cbn_{k,j}$ ; $Cmin_{k,type} \leftarrow ''bn''$
        end if
        if $Cbi_{k,j} < Cmin_k$ then
            $Cmin_k \leftarrow Cbi_{k,j}$ ; $Cmin_{k,type} \leftarrow ''bi''$
        end if
        $sum8 \leftarrow sum8 + Cmin_k$ ; $BT\text{-}sum8 \leftarrow BT\text{-}sum8 \cup (k, j, Cmin_{k,type})$
    end do
    /* calculate min2, min3, min4 (derived from sum6,sum8) */
    $min2 \leftarrow \infty$ ; $BT\text{-}min2 \leftarrow \emptyset$ ; $min3 \leftarrow \infty$ ; $BT\text{-}min3 \leftarrow \emptyset$ ; $min4 \leftarrow \infty$ ; $BT\text{-}min4 \leftarrow \emptyset$
    foreach $k$ in $Ch_i$ do
        $tmp \leftarrow Cbn_{k,j} + sum6 - Cminn_k$
        if ($tmp < min2$) then
        $min2 \leftarrow tmp$ ; $BT\text{-}min2 \leftarrow (k, j, ”bn”) \cup \Big(BT\text{-}sum6 \setminus (k, j, Cminn_{k,type})\Big)$
      end if
      $tmp \leftarrow Cminb_k + sum8 - Cmin_k$
      if ($tmp < min3$) then
        $min3 \leftarrow tmp$ ; $BT\text{-}min3 \leftarrow (k, j, Cminb_{k,type}) \cup \Big(BT\text{-}sum8 \setminus (k, j, Cmin_{k,type})\Big)$
      end if
      $tmp \leftarrow Cbo_{k,j} + sum8 - Cmin_k$
      if ($tmp < min4$) then
        $min4 \leftarrow tmp$ ; $BT\text{-}min4 \leftarrow (k, j, ”bo”) \cup \Big(BT\text{-}sum8 \setminus (k, j, Cmin_{k,type})\Big)$
      end if
    end do
    if $Tdm_i > 0$ then /* There is multicast demand */
      $sum1 \leftarrow \infty$ ; $min2 \leftarrow \infty$ ; $BT\text{-}min2 \leftarrow \emptyset$
    end if
    $Cxn_{i,j} \leftarrow Tdu_i \cdot Dd_{i,j} + Tu_i^{out} \cdot Ucu_i + sum1$
    ...
    $Cbo_{i,j} \leftarrow Td \cdot Ucd_i + Tdu_i \cdot Dd_{i,j} + \Big(Tu_i^{in} + Tu_i^{out}\Big) \cdot Ucu_i + min4$ ; $BT\text{-}Cbo_{i,j} \leftarrow BT\text{-}min4$
    if $Cbo_{i,j_{mino}} < Cbo_{i,j}$ then
      $Cbo_{i,j} \leftarrow Cbo_{i,j_{mino}}$ ; $BT\text{-}Cbo_{i,j} \leftarrow BT\text{-}Cbo_{i,j_{mino}}$
    end if
  end do
end do
/* find the optimal cost */
$j_{min} \leftarrow 1$
for $j = n, n-1, n-2, ..., 2$ do
  if $Cin_{1,j} < Cin_{1,j_{min}}$ then
    $j_{min} \leftarrow j$
  end if
end do
The optimal cost is $Cin_{1,j_{min}}$.
```

Backtrack phase

The backtrack phase for allocation of copies is recursive and can easily be described using a recursive function. The recursion starts by calling **allocate**$(1, j_{min}, ”in”)$.

```
proc allocate $(i, j, type)$ {
  if $type = ”io”$ then
    foreach $(k, l, ktype)$ in $BT\text{-}Cio_{i,j}$ do
      call allocate$(k, l, ktype)$
    end do
  else if $type = ”in”$ then
    foreach $(k, l, ktype)$ in $BT\text{-}Cin_{i,j}$ do
      call allocate$(k, l, ktype)$
    end do
  else if $type = ”bn”$ then
    foreach $(k, l, ktype)$ in $BT\text{-}Cbn_{i,j}$ do
      call allocate$(k, l, ktype)$
    end do
  else if $type = ”bo”$ then
    foreach $(k, l, ktype)$ in $BT\text{-}Cbo_{i,j}$ do
      call allocate$(k, l, ktype)$
```

```
        end do
    else if $type = "bi"$ then
        foreach $(k, l, ktype)$ in $BT\text{-}Cbi_{i,j}$ do
            call **allocate**$(k, l, ktype)$
        end do
    else if $type = "local"$ then
        allocate a copy at $i$
    end if
    return
}
```

Appendix III
More on MX-HDT - Proof and Pseudo-Code

This appendix contains additional details on the HDT algorithm that were not included in section IX.

A. *Proof of MX-HDT Optimality*

The proof of the MX-HDT optimality is very similar to the proof of the HDT optimality. The difference in the proof is the justification of selecting either unicast or multicast distribution for each vertex (but not both), which was not relevant for the HDT algorithm.

The proof is based on induction. Lemma 7 is the induction base.

Lemma 7: For all scenarios, and for all vertices $j{\in}V$ the algorithm optimally allocates the object in $T_{i,j}$, when i is a leaf of T

Proof: According to the definition of the new optimization problem, either one of the following possibilities holds.

1) $j{=}i$ (no string is connected), the algorithm allocates the object at vertex i. There is a copy located at T_i therefore $Cxn_{i,i}$ and $Cxi_{i,i}$ can't exist (set to ∞). $Cbi_{i,i}$ is also impossible, since there is no need for incoming multicast distribution traffic when i stores a copy. The valid possible scenarios are $in_{i,i}, io_{i,i}, bn_{i,i}$ and $bo_{i,i}$ which differ in the assumptions that i is or isn't the only vertex which stores a copy and there is or isn't outgoing multicast distribution traffic from T_i. Since i stores a copy, there must be incoming update traffic through edge e_i. The optimal cost is constructed from the storage cost, the incoming update traffic through e_i, and the additional outgoing update and/or multicast distribution traffic through e_i when appropriate ($Cio_{i,i}$, $Cbn_{i,i}$, $Cbo_{i,i}$).
2) $j{\neq}i$ ($j{\notin}V_i$). A string $P_{j,i}$ is connected to T_i. According to the definition of the problem, there's a copy of the object located at vertex j and i is served (unicast distribution) from that vertex (if i is served by unicast distribution from outside T_i). There is a copy located outside T_i therefore $Cin_{i,j}$ and $Cio_{i,j}$ can't exist (set to ∞). The scenarios $xn_{i,j}$ and $xi_{i,j}$ assume that it costs more to store an object in i (and to keep it updated) than to be served from j. The scenario $xn_{i,j}$ implies that i is served by unicast distribution. The optimal cost is the outgoing update traffic through e_i and the incoming unicast (from j to i) and/or multicast distribution traffic through e_i. The scenarios $bn_{i,j}, bi_{i,j}$ and $bo_{i,j}$ assume that it is cheaper to store a copy in i although j has a copy. Since i stores a copy, the costs calculated for $bn_{i,i}, bi_{i,i}$ and $bo_{i,i}$ must be used accordingly (achieved by setting $min2, min3$ and $min4$ to ∞ for a leaf vertex).

■

Lemma 8 constructs the induction step for the recursive proof of optimality.

Lemma 8: Assume that the algorithm optimally allocates the object to servers in every subtree rooted at vertex c which is a child of i (T_c, $c{\in}Ch_i$) for all scenarios and for all vertices $j{\in}V$, then the algorithm optimally allocates the object in T_i for all the scenarios and for all vertices $j{\in}V$.

Proof: According to the definition of the new optimization problem, either one of the following possibilities holds.

1) $j{=}i$ (no string is connected), the algorithm allocates the object at vertex i. There is a copy located at T_i therefore $Cxn_{i,i}$ and $Cxi_{i,i}$ can't exist (set to ∞). $Cbi_{i,i}$ is also impossible, since there is no need for incoming multicast distribution traffic when i stores a copy. The valid possible scenarios are $in_{i,i}, io_{i,i}, bn_{i,i}$ and $bo_{i,i}$ which differ

in the assumptions that i is or isn't the only vertex which stores a copy and there is or isn't outgoing multicast distribution traffic from T_i. The vertices in each subtree T_k, $k{\in}Ch_i$ may be served by unicast and/or multicast either from vertex i or from copies located internally in the subtree. (The minimum of the following legal scenarios for each $k{\in}Ch_i$: $Cxn_{k,j}, Cxi_{k,j}, Cbn_{k,j}, Cbi_{k,j} \Rightarrow sum3$). Since i stores a copy, there must be incoming update traffic through edge e_i. The optimal cost is constructed from the storage cost at i, the optimal costs calculated by the children ($sum3$), the incoming update traffic through e_i, and the additional outgoing update and/or multicast distribution traffic through e_i when appropriate ($Cio_{i,i}$, $Cbn_{i,i}$, $Cbo_{i,i}$).

2) $j{\in}V_k$, $k{\in}Ch_i$ (no string is connected), the algorithm allocates a copy of the object at vertex j. There is a copy located at T_i therefore $Cxn_{i,j}$ and $Cxi_{i,j}$ can't exist (set to ∞), and there must be incoming update traffic through edge e_i.

   For each of the scenarios where no copy of the object is allocated outside T_i ($in_{i,j}$ and $io_{i,j}$) or copies are also located outside T_i ($bn_{i,j}, bi_{i,j}$ and $bo_{i,j}$), one the following children combinations hold:

   a) There are copies allocated only inside T_k (at least at vertex j), and there is no outgoing multicast distribution traffic from T_k (scenario $in_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ must fulfil scenario $xn_{l,j}$. Vertex i must be served using unicast distribution from vertex j ($sum4$).

   b) There are copies allocated only inside T_k (at least at vertex j), and there is outgoing multicast distribution traffic from T_k (scenario $io_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ must not allocate a copy in T_l (scenarios $xn_{l,j}$ or $xi_{l,j}$) ($sum5$). Vertex i is part of a multicast distribution tree, so it must be served by multicast distribution.

      This combination is valid only for scenarios $in_{i,j}$ and $io_{i,j}$.

   c) There are copies allocated both inside T_k (at least at vertex j) and outside T_k, but there is no multicast distribution through edge e_k (scenario $bn_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l but must not be served by incoming multicast distribution through i (through edge e_l) (scenarios $xn_{l,j}$ or $bn_{l,j}$). Vertex i must be served using unicast distribution from vertex j ($sum6$).

   d) This combination is only valid for scenario $bi_{i,j}$. In this scenario there is incoming multicast distribution through edges e_i.

      There are copies allocated both inside T_k (at least at vertex j) and outside T_k, and e_k. Vertex i is part of a multicast distribution tree, through edge e_i (one of scenarios $bn_{k,j}, bi_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, and may be served by incoming multicast distribution traffic through i (and e_l) ($sum7$).

   e) There are copies allocated both inside T_k (at least at vertex j) and outside T_k, and there is outgoing multicast distribution through edge e_k (scenario $bo_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, and may be served by incoming multicast distribution traffic through vertex i (and edge e_k) ($sum8$). Vertex i is part of a multicast distribution tree, so it must be served by multicast distribution.

      This combination is not valid for scenario $bi_{i,j}$, due to lemma 2.

   f) There are copies allocated both inside T_k (at least at vertex j) and outside T_k (scenarios $bn_{k,j}$ or $bi_{k,j}$), but there is outgoing multicast distribution through some edge e_l, $l{\in}Ch_i$, $l{\neq}k$ (scenario $bo_{l,j}$). In this case the rest of the children $m{\in}Ch_i$, $m{\neq}k$, $m{\neq}l$ may allocate a copy in T_m, and may be served by incoming multicast distribution traffic through i (and edge e_l) ($min1$).

      This combination is not valid for scenario $bi_{i,j}$, due to lemma 2.

   For each such scenario the minimal (optimal) valid children combination is selected, and the optimal cost is constructed from that combination cost (min between $sum4, sum5, sum6, sum7$, $sum8$ or $min1$), the cost of update traffic through e_i (incoming only for $Cin_{i,j}, Cio_{i,j}$, both incoming and outgoing for $Cbn_{i,j}, Cbi_{i,j}, Cbo_{i,j}$) and the cost of either unicast distribution from j to i, or multicast distribution traffic through edge i (outgoing for $Cio_{i,j}, Cbo_{i,j}$, incoming for $Cbi_{i,j}$).

3) $j{\notin}V_i$. A string $P_{j,i}$ is connected to T_i. According to the definition of the problem, there's a copy of the object located at vertex j and i or any vertex $v \in V_i$ may be served (unicast distribution) from that vertex (if they are served by unicast from outside T_i). There is a copy located outside T_i therefore $Cin_{i,j}$ and $Cio_{i,j}$ can't exist (set to ∞), and there must be outgoing update traffic through edge e_i.

   For the scenario $xn_{i,j}$, which means that no copy is located in T_i and there is no incoming multicast distribution traffic to T_i, only one children combination is possible - all the children $k{\in}Ch_i$ must fulfil scenario $xn_{k,j}$. Vertex

i must be served using unicast distribution from vertex j ($sum1$).
For the scenario $xi_{i,j}$, which means that no copy is located in T_i but there is incoming multicast distribution traffic to T_i (i is part of a multicast distribution tree), all the children $k{\in}Ch_i$ must fulfil scenario $xn_{k,j}$ or $xi_{k,j}$ ($sum2$).
For each of the scenarios where copies are also located inside T_i ($bn_{i,j}, bi_{i,j}$ and $bo_{i,j}$), the unicast distribution source of vertex i (if i is served or passes unicast distribution traffic), may be an internal vertex (i.e. j may not be the unicast distribution source of i when there is unicast distribution traffic through i). In such a case, the unicast distribution source l,$l{\in}V_i$ and the corresponding cost $Cb?_{i,l}$ will be used (expression $\min_{l \in V_i} Cb?_{i,l}$). In case i is served from j by unicast distribution, it must not store an object and one the following children combinations hold:

a) This combination is only valid for scenario $bn_{i,j}$. In this scenario there is no multicast distribution through edge e_i. There is at least one vertex k, $k{\in}Ch_i$ which has at least one copy allocated in T_k, and does not produce outgoing multicast distribution traffic (scenario $bn_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, but must not be served by incoming multicast distribution through i (and edge e_l) (scenarios $xn_{l,j}$ or $bn_{l,j}$) ($min2$).

b) This combination is only valid for scenario $bi_{i,j}$. In this scenario there is incoming multicast distribution through edge e_i. There is at least one vertex k, $k{\in}Ch_i$ which has at least one copy allocated in T_k, and may or may not be served by incoming multicast distribution through i (scenarios $bn_{k,j}$ or $bi_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, and may be served by incoming multicast distribution traffic through i (and e_l) ($min3$).

c) There is outgoing multicast distribution through some vertex k, $k{\in}Ch_i$ (scenario $bo_{k,j}$). In this case the rest of the children $l{\in}Ch_i$, $l{\neq}k$ may allocate a copy in T_l, and may be served by incoming multicast distribution traffic through vertex i (and k) ($min4$).
This combination is not valid for scenario $bi_{i,j}$, due to lemma 2.

For each such scenario the minimal (optimal) valid children combination is selected, and the optimal cost is constructed from that combination cost (min between $sum1, sum2, min2, min3$ or $min4$), the cost of update traffic through e_i (outgoing only for $Cxn_{i,j}, Cxi_{i,j}$, both incoming and outgoing for $Cbn_{i,j}, Cbi_{i,j}, Cbo_{i,j}$) and the cost of either unicast distribution from j to i, or multicast distribution traffic through edge i (outgoing for $Cbo_{i,j}$, incoming for $Cxi_{i,j}$, $Cbi_{i,j}$).

■

Theorem 2: When the algorithm ends, $min_{j \in V} Cin_{r,j}$ holds the optimal allocation cost and the allocation of copies is optimal.

Proof: The proof is conducted by the induction where lemma 7 is the base and lemma 8 is the step. For each $j{\in}V$, $Cin_{r,j}$ represents an optimal allocation of the objects where r is served from j. The minimal $Cin_{r,j}$ is the optimal cost of the original optimization problem. In addition, the costs $Cxn_{r,j}, Cxi_{r,j}, Cbn_{r,j}, Cbi_{r,j}$ and $Cbo_{r,j}$ are illegal since there can't be copies allocated outside $T_r \equiv T$ and $Cio_{r,j}$ is illegal there can't be distribution traffic through r outside of T. ■

B. Partial Pseudo code of MX-HDT

We assume that the vertices are ordered by breadth first ordering. Vertex 1 is the root and n must be a leaf.
We also assume the ∞ is the maximal number that exists in the computer.
Variables starting with BT are used for the backtrack process, and store a vertex number or the cost/vertex data.

The algorithm is performed in two phases. The first one is for calculating the optimal cost and the backtrack info for later.
We only provide part of the first phase due to the large number of scenarios and combinations. The main idea in the pseudo code provided is to show the combinations of children scenarios and to demonstrate the generation of backtrack info.

The cost calculation phase of MX-HDT is almost similar to the one of HDT (described in subsection II-B). In the following pseudo code, we only highlighted some of the differences.

Cost calculation phase

```
for $i = n, n-1, n-2, ..., 2, 1$ do
```

```
    /* calculate costs for $i, i$ */
    $sum3 \leftarrow 0$ ; $BT\text{-}sum3 \leftarrow \emptyset$
    ...
    $Cxn_{i,i} \leftarrow \infty$ ; $Cxi_{i,i} \leftarrow \infty$ ; $Cbi_{i,i} \leftarrow \infty$
    $Cin_{i,i} \leftarrow Tu_i^{in} \cdot Ucu_i + Sc_i + sum3$ ; $BT\text{-}Cin_{i,i} \leftarrow BT\text{-}sum3 \cup (i, i, "local")$
    ...
    $Cbo_{i,i} \leftarrow Cio_{i,i} + Tu_i^{out} \cdot Ucu_i$ ; $BT\text{-}Cbo_{i,i} \leftarrow BT\text{-}Cio_{i,i}$
    /* update $\min_{l \in V_i} Cb?_{i,l}$ */
    $j_{minn} \leftarrow i$ ; $j_{mini} \leftarrow i$ ; $j_{mino} \leftarrow i$
    /* calculate costs for $i, j$ where $j \in V_k, k \in Ch_i$ */
    foreach $k$ in $Ch_i$ do
        foreach $j$ in $V_k$ do
            $\boxed{sum4 \leftarrow Tdu_i \cdot Dd_{i,j} + Cin_{k,j} \text{ ; } BT\text{-}sum4 \leftarrow (k, j, "in")}$
            $sum5 \leftarrow Cio_{k,j}$ ; $BT\text{-}sum5 \leftarrow (k, j, "io")$
            $\boxed{sum6 \leftarrow Tdu_i \cdot Dd_{i,j} + Cbn_{k,j} \text{ ; } BT\text{-}sum6 \leftarrow (k, j, "bn")}$
            ...
            ~~if $Tdm_i > 0$ then /* There is multicast demand */~~
                ~~$sum4 \leftarrow \infty$ ; $BT\text{-}sum4 \leftarrow \emptyset$ ; $sum6 \leftarrow \infty$ ; $BT\text{-}sum6$~~ $\leftarrow \emptyset$
            ~~end if~~
            ...
            $\boxed{Cbi_{i,j} \leftarrow Td \cdot Ucd_i + \left(Tu_i^{in} + Tu_i^{out}\right) \cdot Ucu_i + sum7 \text{ ; } BT\text{-}Cbi_{i,j} \leftarrow BT\text{-}sum7}$
            /* update $\min_{l \in V_i} Cb?_{i,l}$ */
            ...
        end do
    end do
    /* calculate costs for $i, j$ where $j \notin V_i$ */
    foreach $j$ in $V \setminus V_i$ do
        ...
        /* calculate min2, min3, min4 (derived from sum6,sum8) */
        $min2 \leftarrow \infty$ ; $BT\text{-}min2 \leftarrow \emptyset$ ; $min3 \leftarrow \infty$ ; $BT\text{-}min3 \leftarrow \emptyset$ ; $min4 \leftarrow \infty$ ; $BT\text{-}min4 \leftarrow \emptyset$
        foreach $k$ in $Ch_i$ do
            $\boxed{tmp \leftarrow Tdu_i \cdot Dd_{i,j} + Cbn_{k,j} + sum6 - Cminn_k}$
            if ($tmp < min2$) then
                $min2 \leftarrow tmp$ ; $BT\text{-}min2 \leftarrow (k, j, "bn") \cup \left(BT\text{-}sum6 \setminus (k, j, Cminn_{k,type})\right)$
            end if
            ...
        end do
        ~~if $Tdm_i > 0$ then /* There is multicast demand */~~
            ~~$sum1 \leftarrow \infty$ ; $min2 \leftarrow \infty$ ; $BT\text{-}min2 \leftarrow$~~ $\emptyset$
        ~~end if~~
        $Cxn_{i,j} \leftarrow Tdu_i \cdot Dd_{i,j} + Tu_i^{out} \cdot Ucu_i + sum1$
        ...
        $\boxed{Cbo_{i,j} \leftarrow Td \cdot Ucd_i + \left(Tu_i^{in} + Tu_i^{out}\right) \cdot Ucu_i + min4 \text{ ; } BT\text{-}Cbo_{i,j} \leftarrow BT\text{-}min4}$
        if $Cbo_{i,j_{mino}} < Cbo_{i,j}$ then
            $Cbo_{i,j} \leftarrow Cbo_{i,j_{mino}}$ ; $BT\text{-}Cbo_{i,j} \leftarrow BT\text{-}Cbo_{i,j_{mino}}$
        end if
    end do
end do
```

```
/* find the optimal cost */
$j_{min} \leftarrow 1$
for $j = n, n-1, n-2, ..., 2$ do
    if $Cin_{1,j} < Cin_{1,j_{min}}$ then
        $j_{min} \leftarrow j$
    end if
end do
The optimal cost is $Cin_{1,j_{min}}$.
```

Backtrack phase

The backtrack phase of MX-HDT is similar to the backtrack phase of HDT described in appendix II, subsection II-B.

REFERENCES

[1] Cisco, http://www.cisco.com/

[2] Akamai, http://www.akamai.com/

[3] Digital Fountain, http://www.digitalfountain.com/

[4] WebDAV, http://www.webdav.org/

[5] Scale8, http://www.scale8.com/

[6] S. Shi and J. Turner. Routing in Overlay Multicast Networks. In Proc. of IEEE INFOCOM, June 2002.

[7] P. Francis. ”Yoid: Extending the Internet multicast architecture”, April 2000.

[8] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek and J. OToole. Overcast: Reliable multicasting with an overlay network. In Proceedings of the Fourth Symposium on Operating Systems Design and Implementation, pp. 197-212, October 2000.

[9] D. Helder and S. Jamin. Banana tree protocol, an end-host multicast protocol. Technical Report CSE-TR-429-00, University of Michigan, 2000.

[10] I. Cidon and O. Unger, Optimal content location in IP multicast based overlay networks, In Proceedings of the 23rd ICDCS workshops, May 2003.

[11] L. W. Dowdy and D. V. Foster. Comparative Models of the File Assignment Problem. ACM Computing Surveys, 14(2) pp. 287-313, 1982.

[12] K. Kalpakis, K. Dasgupta, and O. Wolfson. Optimal Placement of Replicas in Trees with Read, Write, and Storage Costs. IEEE Transactions on Parallel and Distributed Systems, 12(6) pp. 628-637, June 2001.

[13] C. Krick, H. Räcke, and M. Westermann. Approximation Algorithms for Data Management in Networks. In Proceedings of the Symposium on Parallel Algorithms and Architecture, pp. 237-246, July 2001.

[14] I. Cidon, S. Kutten, and R. Sofer. Optimal allocation of electronic content. In Proceedings of IEEE Infocom, Anchorage, AK, April 22-26, 2001.

[15] L. Qiu, V. N. Padmanabham, and G. M. Voelker. On the placement of web server replicas. In Proc. 20th IEEE INFOCOM, 2001.

[16] Sugih Jamin, Cheng Jin, Anthony R. Kurc, Danny Raz and Yuval Shavitt. Constrained Mirror Placement on the Internet, IEEE Infocom 2001.

[17] J. Kangasharju and J. Roberts and K. Ross. Object Replication Strategies in Content Distribution Networks, In Proceedings of WCW’01: Web Caching and Content Distribution Workshop, Boston, MA, June 2001.

[18] S. L. Hakimi. Steiner’s problem in graphs and its implications. Networks, Vol. 1 (1971), pp. 113-133.

[19] M. R. Garey, R. L. Graham and D.S. Johnson. The complexity of computing Steiner minimal trees. SIAM J. Appl. Math., 32 (1977) pp. 835-859.

[20] A. Z. Zelikovsky. The 11/6–approximation algorithm for the Steiner problem on networks. Algorithmica 9 (1993) 463-470.

[21] Ellen W. Zegura, Ken Calvert and S. Bhattacharjee. How to Model an Internetwork. Proceedings of IEEE Infocom ’96, San Francisco, CA.

[22] GT-ITM: Georgia Tech Internetwork Topology Models, http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz