CONTENT LOCATION AND DISTRIBUTION IN CONVERGED OVERLAY NETWORKS

Oren Unger[1,2] and Israel Cidon[1]
[1]*Department of Electrical Engineering, Technion - Israel Institute of Technology;* [2]*Zoran Microelectronics*

Abstract: A major challenge for organizations and application service providers (ASP) is to provide high quality network services to geographically dispersed consumers at a reasonable cost. Such providers employ content delivery networks (CDNs) and overlay networks to bring content and applications closer to their service consumers with better quality.

Overlay networks architecture should support high-performance and high-scalability at a low cost. For that end, in addition to the traditional unicast communication, multicast methodologies can be used to deliver content from regional servers to end users. Another important architectural problem is the efficient allocation of objects to servers to minimize storage and distribution costs.

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

An optimal object allocation algorithm for tree networks is presented with computational complexity of $O(N^2)$. The algorithm automatically selects, for each user, between multicast and unicast distribution. An approximation algorithm for general networks is also suggested. The model and algorithms can be easily extended to the cases where content is updated from multiple locations.

Keywords: Content Distribution; Location Problems; Hybrid networks; Overlay Networks; Tree Networks; Quality of Service

1. 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, organizations 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.

An overlay network is a collection of application servers that are interconnected through the general Internet Infrastructure. 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 popularity of multicast for distribution of such content is increasing with the introduction of real-time and multimedia applications that require high QoS (high bandwidth, low delay loss and jitter) and are delivered to large groups of consumers. Although multicast is efficient for large groups, its high deployment and management cost makes unicast a better solution for small groups, especially for a sparse spread or when data requirements are diverse.

Hybrid overlay networks are overlay networks that use both multicast and unicast as the transport protocol. The new approach suggested in this paper is to combine the replication used in CDNs with multicast/unicast based distribution and achieve better scalability of the service while maintaining a low cost of storage and communication. The novel hybrid approach for data distribution is based on the understanding that in some cases, it is more efficient to use unicast since it saves bandwidth or computational resources.

Our initial model is a tree graph that has a potential server located at each of its vertices. The vertices may also include local consumers. Each server is assigned with a storage cost and each edge is assigned with distribution communication costs. The distribution demands of the consumers are given. The consumers are served from servers using multicast and/or unicast communication. The costs can also be interpreted as QoS related costs or as loss of revenue resulted from reduced performance.

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. Each consumer is served by exactly one server for an object. There is an obvious tradeoff between the storage cost that increases with the number of copies and the distribution cost that decrease with that number.

In this work we present an optimal allocation algorithm for tree networks with computational complexity of $O(N^2)$. We solve the case where the mode of operation per consumer (multicast or unicast) is automatically optimized

by the algorithm itself. We also suggest an approximation algorithm for general networks. The model and algorithms can be easily extended to the case where server content is dynamic and needs to updated from media sources via multicast or unicast means[4].

1.1 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 for a single tree[5–8]. We focus on the way an existing overlay network should be partitioned to multiple regional multicast/unicast trees while optimizing the communication and storage cost. Our earlier work[9] presents an optimal allocation algorithm for the multicast only distribution on trees.

The object allocation problem, also referred as the file allocation problem in storage systems[10] or data management in distributed databases has been studied extensively. When looking only at the unicast distribution model, we end up with the classical "uncapacited plant location problem" (UPLP)[11] with facilities replacing servers and roads replacing communication lines. The problem has been proved to be NP-complete for general graphs[11]. It was solved for trees in polynomial time[12, 13]. The UPLP model was mapped to content delivery networks[14]. There are additional works that address the severs/replicas placement problem for the unicast only distribution model[15–17].

2. THE MODEL

2.1 Objects

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

2.2 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 represents a network switch and a potential storage place for copies of o. Each vertex is also an entry point of content consumers to the network. Distribution demands of consumers connected to vertex i are satisfied by the network from the closest vertex (or the closest multicast tree rooted at) j which stores a copy of o. Consumers

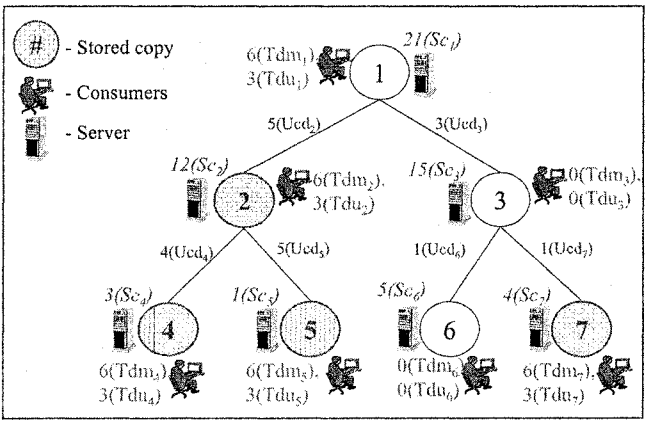


Figure 1. An example of a tree network and various costs

connected to a vertex are served by one of unicast or multicast. The selection is done automatically by the system in order to optimize the overall cost.

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

Figure 1 displays a tree network and costs related to its vertices and edges.

2.3 Storage cost

Let the storage cost of object o at vertex i be Sc_i. Sc_i represents resources, like disk space, computational power and relative maintenance cost. Denote Φ is the set of vertices that store o. The total storage cost of o is $\sum_{i \in \Phi} Sc_i$.

2.4 Distribution traffic cost

Denote the cost per traffic unit at edge e_i as Ucd_i ($Ucd_i > 0$). Since $e_r = \emptyset$, $Ucd_r \equiv 0$. Ucd_i represents the residual cost of traffic in a physical line or the relative cost of the connection to a public network. The cost per 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 graph is undirected, $P_{i,j} = P_{j,i}$.

The mutual exclusive hybrid model automatically selects between the unicast/muticast traffic 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). The effective bandwidth required by a unicast stream is smaller than a multicast stream.

The multicast 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. Td may be the bandwidth requirement, or other QoS related parameters[†]. Since unicast traffic require less bandwidth, the unicast traffic demand in vertex i, Tdu_i is $Tdu_i = q \cdot Tdm_i$, $0 < q < 1$.

Denote the set of vertices which are served using unicast (and are not served by unicast) as V_{uc}; the set of edges in the multicast tree rooted at vertex i as Dmt_i. If $i \notin \Phi$, or i does not serve by multicast, $Dmt_i = \emptyset$.

The total multicast traffic cost is $\sum_{i \in \Phi} Td \cdot (\sum_{e \in Dmt_i} Ucd_e)$. The total unicast traffic cost is $\sum_{i \in V_{uc}} Tdu_i \cdot \min_{j \in \Phi} Dd_{i,j}$. (If $\exists j, k \in \Phi$ s.t. $Dd_{i,j} = Dd_{i,k}$ and $j < k$ then j, the smallest, is taken).

3. THE PROBLEM

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

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

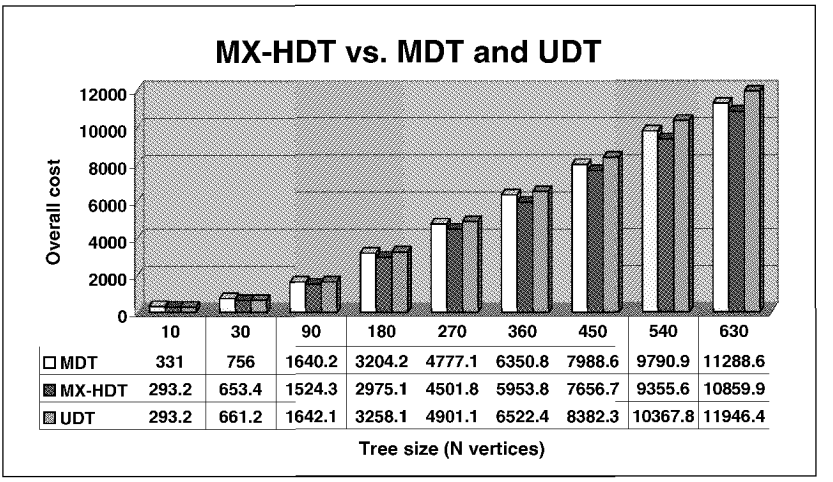
We developed an optimal algorithm called MX-HDT - Mutual eXclusive Hybrid Distribution for Trees, with computational complexity of $O(N^2)$.

4. MX-HDT VS. MDT/UDT RESULTS

As stated in the sub-section 2.4, the MX-HDT algorithm attempts to leverage the advantages of both MDT (multicast only distribution in trees graphs[9]) and UDT (unicast only distribution, also termed UPLP, on trees[13]) by switching between multicast and unicast in order to minimize the overall communication and storage costs. The MX-HDT algorithm always performs better than UDT/MDT and the overall costs will be equal or smaller than the minimum between UDT/MDT.

Figure 2 presents a comparison between the average results of running MX-HDT, MDT, UDT on various trees in various shapes and sizes, while the ratio between the multicast bandwidth demands and the unicast bandwidth demand is 2 : 1 (i.e. $q = 0.5$). It can be seen that MX-HDT performs better than both algorithms in all cases.

[†]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.



	10	30	90	180	270	360	450	540	630
MDT	331	756	1640.2	3204.2	4777.1	6350.8	7988.6	9790.9	11288.6
MX-HDT	293.2	653.4	1524.3	2975.1	4501.8	5953.8	7656.7	9355.6	10859.9
UDT	293.2	661.2	1642.1	3258.1	4901.1	6522.4	8382.3	10367.8	11946.4

Figure 2. MX-HDT vs. MDT/UDT

5. OPTIMAL ALLOCATION PROPERTIES

LEMMA 1 *In case of unicast traffic, if vertex i is served by 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 vertex l is served by unicast through vertex k, l must also be served from j (k itself may not be served by unicast).*

PROOF $P_{i,j}=P_{i,k}\cup P_{k,j} \Rightarrow Dd_{i,j}=Dd_{i,k}+Dd_{k,j}$. *Suppose vertex l is not served by j, but from a different vertex m.* $P_{l,j}=P_{l,k}\cup P_{k,m} \Rightarrow Dd_{l,j}=Dd_{l,k}+Dd_{k,m}$. *Since the solution is optimal there must exist* $Dd_{k,j}=Dd_{k,m}$. *And if* $\exists j, m\in\Phi$ *s.t.* $Dd_{k,j}=Dd_{k,m}$ *and* $j<m$ *then j, the smallest, is taken. So j and m must be the same vertex.*

LEMMA 2 *Each vertex i can only belong to at most one multicast tree.*

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

LEMMA 3 *If vertex i is served through its neighbor k in T (either parent or child), then i and k are served by the same server.*

PROOF *A direct result of lemmas 1 , 2.*

LEMMA 4 *If there is multicast traffic through vertex i, then vertex i must belong to a multicast tree (this property is not correct for unicast).*

PROOF *Suppose there is multicast traffic through vertex i, and i is served by unicast. In this case i belongs to both kinds of trees, and this is a contradiction of the mutual exclusive traffic condition.* ‡

COROLLARY 5 *The optimal allocation is composed of a subgraph of T which is a forest of unicast and multicast subtrees. Each subtree is rooted at a vertex which stores a copy of o. Each edge and vertex in T can be part of at most one unicast and at most one multicast subtree. If a vertex belongs to a multicast tree, it may still pass unicast traffic through its uplink edge.*

6. THE MX-HDT OPTIMAL ALGORITHM

The algorithm calculates the optimal object allocation cost as well as the set of servers that will store the object o.

6.1 The technique

The main idea behind the algorithm 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 few possibilities, and 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, MX-HDT is a recursive algorithm that finds 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.

The algorithm is 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 for each scenario. The second phase is a backtrack phase which starts at the root and ends at the leaves where the algorithm selects the actual scenario in the optimal allocation and allocates the copies in the relevant servers.

The new optimization problem is defined as follows: Find the optimal allocation and its alternate cost in $T_{i,j}$, for the scenarios described in subsection 6.2 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 3. demonstrates the distribution forest (Described in corollary 5) with the different possible scenarios of the vertices and edges in T.

‡Note: the opposite is not a contradiction, i.e. even if there is unicast traffic through vertex i (i.e. - another vertex k is served by unicast through i), then vertex i may still be served by multicast.

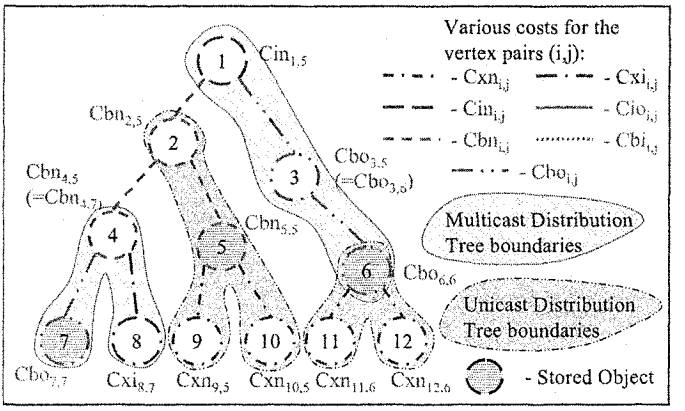


Figure 3. An allocation, scenarios and distribution forest example

6.2 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 o) alternate costs for the following scenarios:

$Cxn_{i,j}$ - e**X**ternal only allocation and **N**o incoming multicast traffic. No copy of o is located inside T_i ($i \neq r$) and edge e_i may only carry unicast traffic. Legal only when $j \notin V_i$.

$Cxi_{i,j}$ - e**X**ternal only allocation and **I**ncoming multicast traffic. No copy of o is located inside T_i ($i \neq r$) and there is distribution demand in T_i that is served by multicast through edge e_i. Legal only when $j \notin V_i$.

$Cin_{i,j}$ - **I**nternal only allocation and **N**o outgoing multicast traffic. All the copies of o are located only inside T_i. Edge e_i may only carry unicast traffic. Legal only when $j \in V_i$.

$Cio_{i,j}$ - **I**nternal only allocation and **O**utgoing multicast traffic. All the copies of o are located only inside T_i and there is distribution demand outside T_i that is served by multicast through edge e_i. Legal only when $j \in V_i$.

$Cbn_{i,j}$ - **B**oth sides allocation and **N**o multicast traffic. Copies are located both inside and outside T_i. Edge e_i may only carry unicast traffic.

$Cbi_{i,j}$ - **B**oth sides allocation and **I**ncoming multicast traffic. Copies are located both inside and outside T_i and there is distribution demand inside T_i that is served by multicast through edge e_i.

$Cbo_{i,j}$ - **B**oth sides allocation and **O**utgoing multicast traffic. Copies are located both inside and outside T_i and there is distribution demand outside T_i that is served by multicast through edge e_i.

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

The algorithm calculates the costs as follows:

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

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

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

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

$$Cbn_{i,j} \leftarrow \begin{cases} \min\{sum6, sum8, min1\}, & \text{if } j \in V_k, k \in Ch_i \\ Sc_i + sum3, & \text{if } j = i \\ \min\left\{\min\limits_{l \in V_i} Cbn_{i,l}{}^{\star}, \min\{min2, min4\}\right\}, & \text{if } j \notin V_i \end{cases}$$

$$Cbi_{i,j} \leftarrow \begin{cases} Td \cdot Ucd_i + sum7, & \text{if } j \in V_k, k \in Ch_i \\ \infty, & \text{if } j = i \\ \min\left\{\min\limits_{l \in V_i} Cbi_{i,l}{}^{\star}, Td \cdot Ucd_i + min3\right\}, & \text{if } j \notin V_i \end{cases}$$

$$Cbo_{i,j} \leftarrow \begin{cases} Td \cdot Ucd_i + \min\{sum8, min1\}, & \text{if } j \in V_k, k \in Ch_i \\ Td \cdot Ucd_i + Sc_i + sum3, & \text{if } j = i \\ \min\left\{\min\limits_{l \in V_i} Cbo_{i,l}{}^{\star}, Td \cdot Ucd_i + min4\right\}, & \text{if } j \notin V_i \end{cases}$$

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

$sum1$-$sum8$ and $min1$-$min4$ represent combinations of children scenarios ($sum1$-$sum8$ equal 0, $min1$-$min4$ equal ∞ if i is a leaf). A detailed explanation about the combinations and the proof of optimality exist in our TR[4].

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

6.3 Backtracking for the content allocation

While calculating the alternate costs for each vertex pair i, j, the algorithm remembers for each such 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.

The backtrack phase starts at the root and ends at the leaves of T. 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. The pseudo code and backtrack details of the algorithm are given in our TR[4].

6.4 Computational complexity of MX-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 $O(|Ch_i| + 1)$. $|V| = N$ and the total number of children in the tree is $N-1$ (only the root r is not a child). The complexity of the backtrack phase for vertex i is $O(1)$. The computational complexity of the algorithm is:

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

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

7. THE MX-HDG APPROXIMATION ALGORITHM

We extended the model to general graphs. Figure 4 depicts a network with the costs related to its vertices and edges.

7.1 The optimal allocation properties

The well known Steiner tree problem[18] is defined as follows: given a (edge) weighted undirected graph and a given subset of vertices termed terminals, find a minimal weight tree spanning all terminals. Consequently, 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 for general graphs. The optimal solution in a general graph is a forest of Steiner trees for multicast traffic and

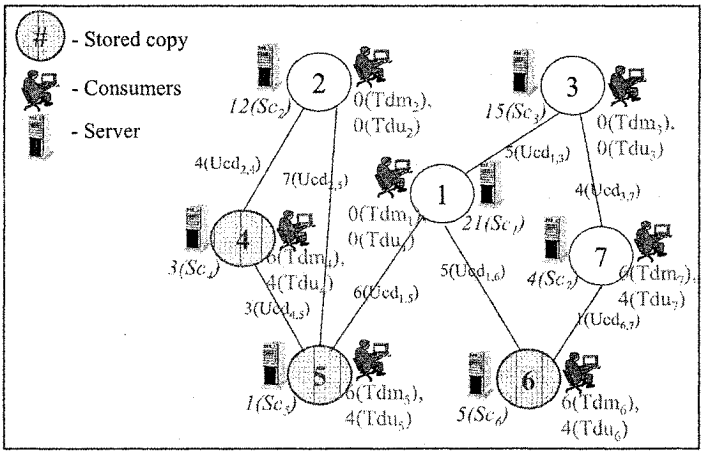


Figure 4. An example network and costs.

additional unicast paths where unicast is used. The total cost of the optimal allocation is constructed of the storage cost, the multicast Steiner trees costs and the unicast traffic cost.

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.

7.2 The approximation algorithm

As the allocation problem in general graphs is NP-hard, we use our optimal MX-HDT algorithm for trees to develop an approximation to that problem. This is an iterative algorithm that starts with an initial random or arbitrary allocation, and converges to an allocation which is optimal in an approximated Steiner tree computed for the general graph.

7.2.1 The MX-HDG algorithm steps

1 Start with a random allocation and set min_{cost} to ∞.

2 Compute a Steiner tree for the graph where the terminals are all the vertices with distribution demand and the vertices which store the object.

3 Run MX-HDT on the extracted tree. For the $Dd_{i,j}$ values use the shortest path between i and j in the graph. The result of the algorithm is a new set of vertices which store the object and a new approximated cost.

4 If the new cost is smaller than min_{cost} save the new allocation and update min_{cost} to be the new cost.

5 Repeat steps 2 to 4 till there is no improvement in the allocation cost.

At the end of the execution, the last saved allocation and min_{cost} are the approximated allocation and cost.

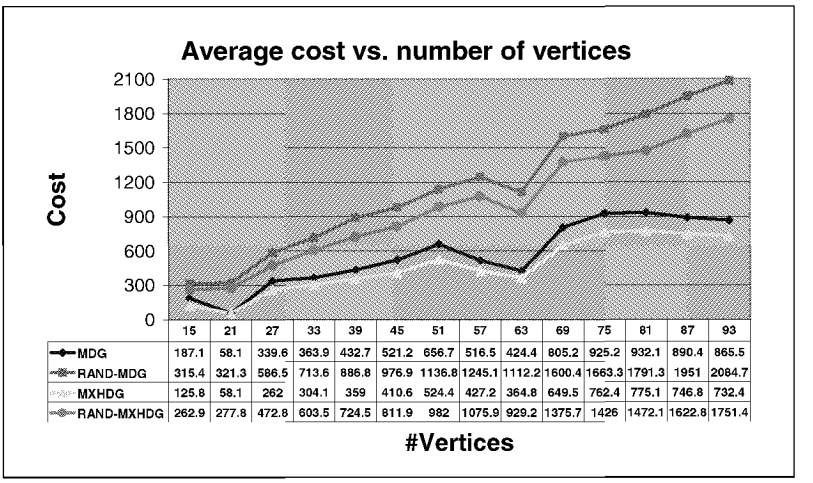
7.2.2 Computing a Steiner tree

The problem of finding a Steiner tree is NP-hard. There are several polynomial time approximations for the problem. We selected the approximation suggested by Zelikovsky[20], which has an approximation ratio of $11/6$.

7.3 MX-HDG Simulation results

We've generated general graphs using the Internet Model suggested by Zegura et al.[21, 22]. We've generated graphs with various node counts.

We've run our approximation MX-HDG algorithm on these graphs, and compared the results to random allocations and the multicast only (MDG) algorithm[4]. Figure 5 displays these charts.



	15	21	27	33	39	45	51	57	63	69	75	81	87	93
MDG	187.1	58.1	339.6	363.9	432.7	521.2	656.7	516.5	424.4	805.2	925.2	932.1	890.4	865.5
RAND-MDG	315.4	321.3	586.5	713.6	886.8	976.9	1136.8	1245.1	1112.2	1600.4	1663.3	1791.3	1951	2084.7
MXHDG	125.8	58.1	262	304.1	359	410.6	524.4	427.2	364.8	649.5	762.4	775.1	746.8	732.4
RAND-MXHDG	262.9	277.8	472.8	603.5	724.5	811.9	982	1075.9	929.2	1375.7	1426	1472.1	1622.8	1751.4

Figure 5. The number of copies and cost of allocations in general graphs

As can be seen from the results, the average costs of MX-HDG are better than MDG (typically by 20%) and significantly better than of the results of the random allocations (the differences between RAND-MDG and RAND-MXHDG, is due the distribution model - multicast vs. hybrid).

8. CONCLUSIONS AND FUTURE WORK

In this work, we addressed the content location problem in hybrid overlay networks, while optimizing the storage and communication costs in the context of QoS provisioning.

We developed an optimal content allocation algorithm for tree networks with computational complexity of $O(N^2)$. The algorithm is recursive and is based on dynamic programming. The algorithm can easily be specified as a distributed algorithm due to the independent calculations at each vertex (only based on information from its neighbors) and due to the hierarchical data flow.

In addition to the optimal algorithm we suggested an approximation for general networks, which requires a small number of iterations, and is based on our optimal algorithm for tree networks.

In our extended work[4], that is not presented here because of space limitations, we address a more general problem where additional media sources are added and additional update communication from the media sources to the servers is considered in the optimization problem.

REFERENCES

1. Cisco, http://www.cisco.com/
2. Akamai, http://www.akamai.com/
3. Digital Fountain, http://www.digitalfountain.com/
4. O. Unger and I. Cidon, Content location in multicast based overlay networks with content updates, CCIT Report #432, Technion Press, June 2003.
5. S. Shi and J. Turner. Routing in Overlay Multicast Networks. In Proc. of IEEE INFOCOM, June 2002.
6. P. Francis. "Yoid: Extending the Internet multicast architecture", April 2000.
7. D. Helder and S. Jamin. Banana tree protocol, an end-host multicast protocol. Technical Report CSE-TR-429-00, University of Michigan, 2000.
8. J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek and J. O'Toole. 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. I. Cidon and O. Unger, Optimal content location in IP multicast based overlay networks, In Proceedings of the 23rd ICDCS workshops, May 2003.
10. L. W. Dowdy and D. V. Foster. Comparative Models of the File Assignment Problem. ACM Computing Surveys, 14(2) pp. 287-313, 1982.
11. Pitu B. Mirchandani, Richard L. Francis. Discrete Location Theory, John Wiley & Sons, Inc. 1990.
12. Kolen A., "Solving covering problems and the uncapacited plant location problem on trees", European Journal of Operational Research, Vol. 12, pp. 266-278, 1983.

13. Alain Billionnet, Marie-Christine Costa, "Solving the Uncapacited Plant Location Problem on Trees", Discrete Applied Mathematics, Vol. 49(1-3), pp. 51-59, 1994.
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