Computer Networks 40 (2002) 205–218

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Optimal allocation of electronic content ☆

Israel Cidon [a], Shay Kutten [b,*], Ran Soffer [a]

[a] *Department of Electrical Engineering, Technion—IIT, Haifa 32000, Israel*

[b] *The William Davidson Faculty of Industrial Engineering and Management, Technion—IIT, Haifa 32000, Israel*

Received 17 December 2001; accepted 4 February 2002

Responsible Editor: I.F. Akyildiz

Abstract

The delivery of large files to individual users, such as video on demand or application programs to the envisioned network computers is expected by many to be one of the main tasks of broadband communication networks. This requires high bandwidth capacity as well as fast and dense storage servers. This motivates multimedia service providers to optimize the delivery network, as well as the electronic content allocation.

A hierarchical architecture for the distribution of multimedia content was introduced by Nussbaumer, Patel, Schaffa, and Sterbenz (INFOCOM 94). They addressed the trade-off between bandwidth and storage requirements that results from the placement of the content servers in the hierarchy tree. They presented a centralized algorithm to compute the best level of the hierarchy for the server location to minimize the combined cost of communication and storage.

In this work, we solve a general case where servers can be placed at different levels of the hierarchy. We develop a distributed optimal location algorithm that requires small nodal memory capacity and computational power. Previous results for related problems for caching system design are of higher complexity. Previous results for related classic operations research problems are limited to centralized algorithms, based on linear programming, that are not easy to convert into distributed algorithms. Instead, to obtain our results, we observed that the use of dynamic programming naturally lends itself to a distributed implementation.

For the specific problem at hand, we also managed to find a natural function (a generalization of the problem) that simplifies the combination operation used in the design of a dynamic program.

1. Introduction

Over the next few years, a tremendous development in multimedia services is expected. The future broadband network is expected to support multimedia services to mobile devices, homes and to workplaces through XDSL, cable, and wireless network providers. Extensive research on the transport of multimedia traffic through communication networks has been conducted in recent years for the communication network and server performance [3,5–8,10,12–15,19,21]. Media providers that operate on-line are trying to minimize the operational cost of their distribution services. These include distribution of application code and data to

☆ A preliminary version of this paper appeared the Proceedings of INFOCOM'01.

* Corresponding author.

E-mail addresses: cidon@ee.technion.ac.il (I. Cidon), kutten@ie.technion.ac.il (S. Kutten), rans@tx.technion.ac.il (R. Soffer).

network computers, computation services via Application Service Provider (ASP model), video on demand (VOD), network Web caching, and Storage Service Providers (SSP).

A hierarchical architecture for providing such multimedia content was introduced by Nussbaumer et al. [20]. They also introduced the trade-off between bandwidth and storage requirements resulted from the placement of the content servers in the hierarchy tree. They computed the best level of the hierarchy for the server location that minimize the combined cost of communication and storage, i.e., the cost of storage within servers combined with the cost of transferring information among and between servers to end-users. The expected competition among multimedia providers, may benefit the ones that can provide these services at the expected quality of service at the lowest price. For example, in the case of VOD, the servers can be owned or leased by the VOD providers. In either case, the provider pays for the storage of video copies in the servers and for the use of communication links for transferring information over the network to the content consumers. The problem of the VOD provider is to manage the storage of video copies within the servers, such that the overall (communication and storage) cost is minimized. The question is: how many copies should be allocated from each video and at which servers?

One abstraction of this problem [20] is as follows. Consider a graph whose nodes are servers of the distribution network and whose links are the communication links connecting them. Each node represents a server that may or may not contain multimedia programs. Consider a user attached to a given node that requests a particular program. The program is transferred to that end user over the network. Two extreme solutions come to mind when attempting to minimize the storage and communication cost. If the communication cost is negligible, the best solution is to have a single server (at a root) that stores all the video content and all the users are served directly from it. On the other hand, if storage cost in a server is negligible, the best solution is to copy all the content to all the servers which are the closest to the users. In other cases these two extreme solutions are not optimal and the optimal solution is to place the electronic content at several places in the network so that an optimal trade-off is achieved.

Note that our model assumes that users access the content on-demand. Therefore, in general, it is impossible to combine media transmissions (even of the same program), from the same server to any two different users. This is due to the fact that the users may start viewing the program (or access an application or file) at different times, or apply different controls over it. Since we assume that non-server nodes cannot store (or cache) content, when the second user accesses it, a second transmission from the server must start anew even if the previous user is still downloading a parts of the same content. The case that transmission can be combined for a video service (e.g. by requesting users to wait until they can be combined) was studied, e.g. in [18].

Similar models with different applications and variations have already been dealt with in the operations research and computer science areas. For example, the “multi-copy file allocation” [1] (NP-complete on a general graph) was explored in the context of theoretical computer science. The “uncapacitated plant location” [2,4] was investigated in the context of operations research. There are several other problems from operations research which can be mapped into networking, such as the p-median problem and the medi-centers problem [4,11].

In this work, we present a distributed algorithm for solving electronic content allocation over a (directed) distribution tree. This algorithm minimizes the overall cost (storage and communication) of the media provider. In a previous solution for a distribution of VOD over a tree [9,20] only a limited solution was considered. It was assumed that all the servers must be placed at the same depth on the tree. The solution was, therefore, limited to finding the optimal depth (which in turn dictated the number and locations of all servers). Our work allows optimal solution in which any number of servers to reside at different levels of the tree. Our cost model is also a generalization of the cost in [20]: we allow different costs at different servers for storing the same object.

In the classical operations research setting for similar problems (there is no previous solution for the directed tree problem) the solutions are centralized algorithms, based on linear programming, that are hard to convert into distributed algorithms. This seems to be adequate when dealing with trucks and roads. When dealing with communication networks, however, distributed algorithms are highly desirable. This is true in particular when all nodes are storage servers and files can be allocated there on demand. In this work we observe that the use of dynamic programming is natural for extending the algorithm towards distributed implementation.

Generally, in dynamic programming over a tree, results for some function computed for children nodes are combined to recursively yield the next result for the parent node. While it is not obvious how to combine the children results for any arbitrary function, we managed to find such function for the problem at hand.

Our algorithm can also be used as a centralized algorithm with sequential time complexity of $O(dN)$ for locating objects of one type (where d is the depth of the tree and N is the number of nodes; since $d < N$ the complexity is also $O(N^2)$). (It is conceivable that the algorithm of [2], with a higher complexity for undirected trees, can be "massaged" to yield a sequential time complexity similar to our centralized version for the directed case.) When the algorithm is used as a distributed algorithm, the message complexity is $O(N)$, the bit complexity is $O(d \log COST\, N)$, and the time complexity is $O(d)$. (The value of $COST$ is the largest cost reported by a child to its parent.)

Interesting and different abstractions of related problems were made in [22,23]. They assume that the number of servers, t, is known in advance. The functions they optimize are the times of delivery to the users (measured by the distance in hops from the server), versus the network capacity used. The problem dealt with in the current paper seems, at first glance, to be harder, since we do not know t, but rather compute the best t from the storage cost. One conclusion that may be derived from the current paper is that our problem is, probably, not harder, since the computational time complexities of the algorithms of [22,23] are higher (and depend also on t). In [22] they also analyze the more complex case that the servers (except for the one in the root) contain only a cache of a title, rather than a complete copy. Thus, a user addressing the title can sometimes be served from the cached copy, but some other times needs to be served from the root (that is, the "hit-ratio" is smaller than 1).

The paper of [24] deals with placing t servers (for a known t) on an undirected tree to minimize the sum of the distances from each user to the server closest to it.

In all the previous caching papers mentioned above, the distribution structure is a hierarchy, that is, a tree (possibly embedded in a more general network). Even when the network is not a physical tree, it is rather common to structure the logical distribution as a tree. See, for example, the cable TV networks, or the broadcasting trees for Multicast [32] including Internet Multicast [17,19]. This simplifies the control and is cheaper in resource utilization. The distribution in these trees is performed from the root towards the leaves. Consequently, this problem over a directed-tree-network is an important practical case. Moreover, methods exist that approximate good results in networks, given good results for trees, by covering the network by multiple trees (see e.g. [16]). The use of (overlapping) distribution trees (one per object) for Web caching is promoted in [27]. The idea is that the more popular is an object, the nearer to the leaves are its copies located. The use of a tree structure for video on demand distribution is also promoted or studied by [28,29,30,31], and others.

Recently, [26] dealt with the related problem of placing mirror sites on a general networks. In order to get a feasible algorithm they constrained the solutions (e.g. the number of mirrors). The evaluation method used there to measure algorithm performance is simulation.

Another practical problem we address is the Servers Allocation Problem over a distribution network. Consider the same model as the electronic content allocation with the addition of server cost. Each node that contains programs is considered a "server node" (a node with a server connected to it). Assume that the cost for connecting a server to Node i is known. The problem is to decide how many servers to allocate and where they should be placed in the distribution network. We show that the same algorithm can solve the Servers Allocation Problem and the electronic content allocation simultaneously. We also show that the sequential time complexity when the algorithm is used as a central algorithm for server allocation is $O(N^{K+1})$, where K is the number of different programs contained in the distribution network. In addition, when the algorithm is used as a distributed algorithm for server allocation, the message complexity is $O(N)$, the bit complexity is $O(N \log COST\, d^K)$, and the time complexity is $O(d)$.

This paper is organized as follows. Section 2 describes the network model and the cost functions. Section 3 describes the Optimal Location Algorithm (OLA). The pseudo-formal code appears in Appendix A. Section 3.3 presents the correctness and the computational complexity of OLA. An example of the execution of OLA is described in Appendix B. The last section deals with the Servers Allocation Problem and its solution based on OLA.

2. The model

A directed tree $T = (V, E)$ represents the communication network, where V is the set of nodes and E is the set of directed communication links. The tree is directed from a root toward the leaves. Link i is the incoming link of Node i as depicted in Fig. 1. Each node of the tree represents a communication switch and also a potential server that contains copies of objects. Intuitively, an object may be a program, a video movie, etc. The leaves of the directed tree represent the *head-ends*, i.e., the devices that connect the users to the network. A request of a user at Head-end u for an Object o is satisfied by the network (by communication from a Node v that is the nearest to u, among the servers that contain a copy of Object o, and are ancestors of u).

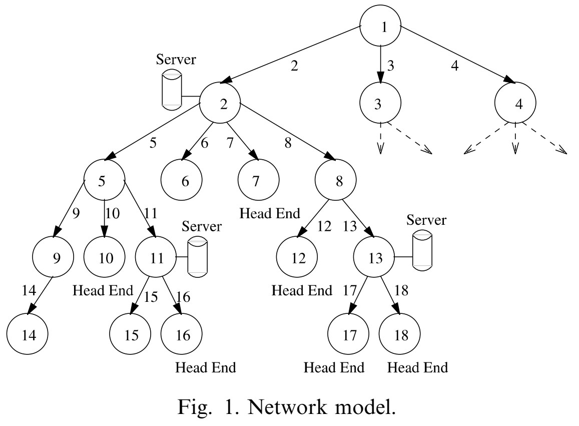

Fig. 1. Network model.

In this paper we assume that all the requests are given precisely, and in advance, and that the network does not change. Additional work is required in order to analyze how can our scheme be adapted to a dynamic environment such as the Internet.

Model assumptions

(1) The network must satisfy the head-end's requests for objects. That is, one of the nodes on the path from the root to any head-end must contain the object.
(2) The links capacities are sufficient to provide all the requested objects to all the head-ends.
(3) The links are bi-directional in terms of message exchange, but directed from the root to the leaves for the provision of objects to the head-ends.
(4) For simplicity and without loss of generality, we assume that each node has a unique identity. Before the algorithm starts, each node is aware of the identity of all its children and its distance (number of links) from the root of the tree.
(5) The incoming link of Node i is marked by the identity of Node i.
(6) Messages received at Node i include the identity of the sending node.

2.1. Storage cost

To be as general as possible, we do not define the storage cost to be depend only on the size of an object, or on any other specific parameter. We even assume that different servers may have different storage costs for the same object. Thus, if S_{Ci} is the storage cost of an object at Node i, the overall storage cost for this object is $\sum_{i \in \Lambda} S_{Ci}$ where Λ is the set of nodes that contain the object.

Since our paper deals with minimizing the total cost of the network, setting $S_{Ci} \leftarrow \infty$ means that there is no server at Node i. When there are different types of objects, $S_{Ci,k}$ is the storage cost of object type k at Node i and the overall storage cost for K types of objects is $\sum_k \sum_{i\in\Lambda_k} S_{Ci,k}$, where Λ_k is the set of nodes which contain the object type k.

2.2. Communication (bandwidth) cost

The communication cost over Link i, or the bandwidth requirement calculation for Link i: The request rate for object (type) k, by the users who are connected to head-end u is the bandwidth requirement from u to Object k. (Recall that we do not assume that the requirements of two different users can be combined, since the users may start viewing the object at different times.) If Node u does not contain Object k, the bandwidth requirement from Link u (the incoming link of Node u) is equal to the bandwidth requirement from Node u. Otherwise (Node u contains Object k), the bandwidth requirement from Link u is zero. This holds for each Link $i \in E$. The bandwidth requirement of an internal Node i is the sum of the bandwidth requirements of all its outgoing links.

Example 1. The network represented in Fig. 2 is a directed tree with 12 nodes labeled 1–12 and 11 directed links labeled 2–12 ($E = \{2, 3, \ldots, 12\}$). We assume that the network provides one type of object. The nodes that are marked with $\{*\}$ contain a copy of the object ($\Lambda = \{1, 3, 7, 12\}$). The users' bandwidth requirements are given under each head-end. Let us calculate the bandwidth requirements from the network links.

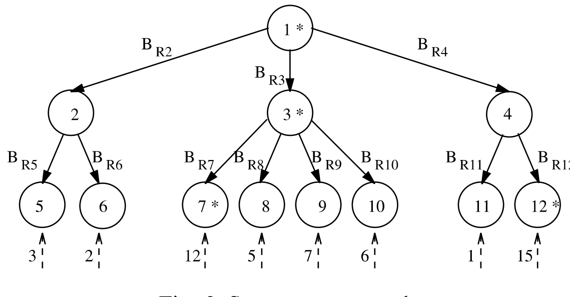

Fig. 2. System cost example.

$B_{R12} = 0$ {because Node 12 contains the object}; $B_{R11} = 1$ {the bandwidth requirement of the users that are connected to it}; $B_{R10} = 6$; $B_{R9} = 7$; $B_{R8} = 5$; $B_{R7} = 0$; $B_{R6} = 2$; $B_{R5} = 3$; $B_{R4} = B_{R12} + B_{R11} = 0 + 1 = 1$ {the bandwidth requirements from Link 4 is the sum of the bandwidth requirements of the links connecting Node 4 to its children, Links 11 and 12}; $B_{R3} = 0$ {Node 3 contains the object}; $B_{R2} = B_{R6} + B_{R5} = 2 + 3 = 5$.

The communication cost per bandwidth unit may be different between links. Let W_i be the cost per bandwidth unit on Link i and B_{Ri} the bandwidth requirement from Link i to a specific object. The communication cost over Link i to a specific object is $C_{Ci} = W_i \cdot B_{Ri}$. The overall communication cost is $\sum_{i\in E} C_{Ci} = \sum_{i\in E} W_i \cdot B_{Ri}$.

Example 2. Same as Example 1 and assuming that $W_i = 1\ \forall i$. Therefore, $C_{Ci} = B_{Ri}$, and the overall communication cost is $\sum_{i=2}^{12} B_{Ri} = 30$.

When there are different types of Objects, $B_{Ri,k}$ is the bandwidth requirements from Link i to Object type k. The overall communication cost is

$$\sum_{i\in E} C_{Ci,k} = \sum_{i\in E} \left(W_i \sum_k B_{Ri,k} \right).$$

2.3. Total cost

The total cost of the system is the sum of the overall cost of communication and storage. For a single object the total cost is

$$\sum_{i\in\Lambda} S_{Ci} + \sum_{i\in E} W_i \cdot B_{Ri}$$

where Λ is the set of nodes that contain a copy of the object. When there are different types of objects, the total cost is

$$\sum_k \left(\sum_{i\in\Lambda_k} S_{Ci,k} + \sum_{i\in E} C_{Ci,k} \right).$$

Example 3. Consider again Example 1 (see Fig. 2) with one object type. The set of nodes that contain the object is $\Lambda = \{1, 3, 7, 12\}$. Assuming that the storage cost for the object in each Node i is 10

($S_{Ci} = 10\ \forall i$), the overall storage cost is $\sum_A S_{Ci} = 40$, and the total cost of the system is $\sum_A S_{Ci} + \sum_E W_i \cdot B_{Ri} = 40 + 30 = 70$.

3. Optimal location algorithm

The OLA optimally allocates copies of a single object type to nodes in a directed tree by minimizing the total cost. This section presents the description of OLA (Section 3.1) and its Pseudo-code (Appendix A) at each node of the directed tree.

We define here a new problem, that may look somewhat artificial but it is one that lends itself easier to a solution by dynamic programming. The new problem is a generalization of the one we want to solve, and thus, its solution also solves the original problem.

Let T_i' be the subtree of T, rooted at Node i, that includes all the nodes which are descendants of i. To solve the problem of optimal allocation of objects in a tree $T = (V, E)$, note that one can view OLA as first solving, for every sub-tree T_i' the expanded problem of optimal allocation of objects in a somewhat different tree. That tree, T_i, is constructed from T_i' and an additional string connected to it as shown in Fig. 3. The string consists of the nodes $\{v_1', v_2', \ldots, v_j'\}$ and the edges $\{e_p' = v_{p-1}' = (v_i', v_{p-1}') \mid p = 1, 2, \ldots, j$, such that $v_0' = i$ is the original root of $T_i'\}$.The string length is j. At Node v_j' there is a server with a copy of the object. Note that this copy is at distance j from the root of tree T_i'. In the description of the algorithm (Section 3.1) the treatment of a sub-tree T_i' corresponds to the treatment of (its root) i, and a string of length j corresponds to the initial distance of the sub-tree from some server which contains a copy of the object.

The storage cost for an object in the string S_j is not included in the total cost of tree T_i', but the cost of communication over the links of the string is included. It is obvious that this is a generalization of the original problem: when the string length is zero and $T_i' = T$, the problem is reduced to the original one.

Given the construction above, the reader can construct the dynamic program herself/himself. It appears below, for the sake of completeness.

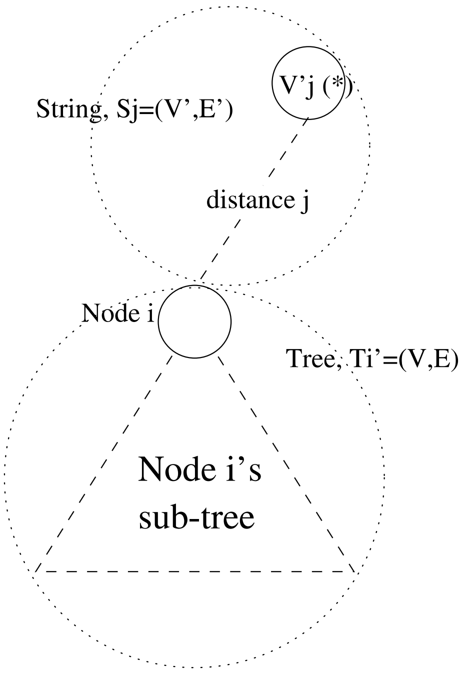

Fig. 3. Tree with a string.

3.1. OLA description

The (distributed version of the) OLA has two phases. In Phase 1, which begins at the leaves and ends at the root of the tree, each Node i calculates the optimal cost for the sub-tree rooted at it as a function of the distance from the nearest copy of the object up from Node i (that is, toward the root). Thus, Node i calculates the optimal cost for each distance possibility, and determines from where to obtain the object {*up, here or down*} depending on that possibility. Node i holds these costs and locations in a table as shown below.

j	$Cost_i$	$Location_i$
-1		
0		
1		
$\vdots$		
d_i		

In the table above, j (the same j as in Fig. 3) is the *initial distance* [1] to the object copy up the tree (i.e. an ancestor). The case $j = -1$ corresponds to the case that there is no copy of the object up the tree nor at the node itself; $j = 0$ is the case where the node may have a copy of the object; $j \geqslant 1$ is the case where the nearest copy of the object is at distance j up from Node i (at the end of a string at length j, see Fig. 3). *Cost* at line j (the j's line of the table) is the optimal cost of the sub-tree rooted at the node, assuming that the nearest object copy is at an initial distance j up the tree.

Location $\in$ {up, here, down} at line j is the decision of "from where to obtain the object when the initial distance is j".

- *location* = *up*, the head-ends will obtain the object from some node up the tree from the current node.
- *location* = *here*, the head-ends will obtain the object from the current node (a copy of the object is to be located at the node itself). That is, it is cheaper to copy the object to the current node, than to use a copy at distance j up the tree from it.
- *location* = *down*, head-ends located below the current node in the tree will obtain the object from nodes down from the current node (but, for each head end, from above the head-end itself).

Table calculation for Node i

The table is calculated in one of two ways, depending on whether Node i is a leaf or an internal node. A node will only start to calculate after all its children have completed their calculation.

(1) **For a Leaf** i
- For line $j = 0$, Leaf i has a copy of the object; the *cost* is set to the storage cost at Leaf i (S_{Ci}); *location* is set to *here* ($cost_i[0] = S_{Ci}$, $location_i[0] = here$).
- For all lines $j > 0$: If the local storage cost (S_{Ci}) is less than the *communication cost*, [2] then the *cost* is set to the storage cost and *location* is set to *here* ($cost_i[j] = S_{Ci}$, $location_i[j] = here$). Otherwise (it is cheaper to obtain the object from up the tree than to copy the object in Node i), the *cost* is set to the communication cost and *location* is set to *up* ($cost_i[j] = B_{Ri}\sum_{l\in\Phi} W_l$, $location_i[j] = up$).

(2) **For an internal Node** i
- For line $j = -1$: where there is no object copy up from Node i nor at Node i itself, the *cost* is the sum of all children *costs* at their line 0 (at the tables of the children) and *location* is set to *down* ($cost_i[-1] = \sum_{l\in\mathscr{CH}_i} cost_l[0]$, $location_i[-1] = down$, where $\mathscr{CH}_i$ is the set of Node i's children).
- For line $j = 0$: where Node i starts assuming it has a copy of the object, if the *cost* at line -1 is less then the sum of all children *costs* at their line 1 plus the storage cost at Node i ($cost_i[-1] < S_{Ci} + \sum_{l\in\mathscr{CH}_i} cost_l[1]$) then, the *cost* is set to the *cost* of line -1 and *location* is set to *down* ($cost_i[0] = cost_i[-1]$, $location_i[0] = down$). Otherwise, the *cost* is set to the sum above, and *location* is set to *here* ($cost_i[0] = S_{Ci} + \sum_{l\in\mathscr{CH}_i} cost_l[1]$, $location_i[0] = here$).
- For all lines $j \geqslant 1$: If the *cost* at line 0 is less than the sum of all children *costs* at their lines $j+1$ then the *cost* is set to the *cost* at line 0 and *location* is set to the *location* of line 0 ($cost_i[j] = cost_i[0]$, $location_i[j] = location_i[0]$). Otherwise, the *cost* is set to the sum of all children *cost* at their lines $j+1$, *location* is set to *up* ($cost_i[j] = \sum_{l\in\mathscr{CH}_i} cost_l[j+1]$, $location_i[j] = up$).

In Phase 2, which begins at the root and ends at the leaves, each node determines whether to copy

[1] Node i starts with the assumption that the nearest server which contains the object is at distance j up the tree. The node algorithm may change the distance to the nearest copy by copying the object to a nearer node—either the node itself or even to lower servers.

[2] The communication cost is the cost for obtaining the object from a node at distance j up from Leaf i ($B_{Ri}\sum_{l\in\Phi} W_l$, where Φ is the set of links between Leaf i and the node at distance j from Leaf i).

the object to itself. The root of the tree ends Phase 1 (after all the other nodes have completed their participation in Phase 1, this is known to the root since a parent always starts its participation in Phase 1 after its children finished) and starts Phase 2. The root determines the optimal cost of the tree by choosing the minimum *cost* between the two lines in its table ($j = -1$ or $j = 0$) and sends the message $place = m$ (where m is the index of the line having minimum *cost*) to all its children. (A copy of the object must be allocated to the root if and only if the *place* is zero.) A node which receives the message *place* performs: $place = place + 1$ and checks the *location* column at the line that equals *place*. If *location* is *here*, the node sets *place* to zero ($place = 0$), sends *place* messages to all children and copies the object to itself. If *location* is *down* it sets *place* to -1 ($place = -1$) and sends *place* messages to all children. If *location* is *up* it sends *place* messages to all children. When the message *place* is received at all the leaves, the algorithm ends and the problem of optimally allocating the object copies, in terms of minimizing the communication and storage cost, is solved. An example of the algorithm execution appears in Appendix B and the Pseudo-code appears in Appendix A.

3.2. Simple extensions

It is very simple to extend the algorithm to the case of K object types (just by using it for each object separately), and to the case that a user may be connected to an internal node by introducing "virtual" leaf nodes with zero communication cost to the internal node where users are desired.

3.3. Correctness

We present proof for the generalized problem (with the tree attached to a string) that is based on induction and the assumption that OLA has terminated. (Proof for the original problem follows immediately.) The proof that OLA terminates is trivial: the termination of Phase 1 can easily be shown by induction, starting with the leaves and ending with the root. The termination of Phase 2 is trivial by induction, starting with the root and ending with the leaves. Lemma 1 is used as the induction base.

Lemma 1. *For all string lengths, OLA optimally allocates the object copies in Tree T_i' when i is a leaf of T.*

Proof is easy using a case study. The full proof appears in Appendix C.

Lemma 2 is the induction step.

Lemma 2. *Assume that OLA optimally allocates object copies to servers in every tree that is rooted at a child of Node i ($T_1, T_2, \dots, T_{|\mathscr{CH}_i|}$, see Fig. 4), for all string lengths. OLA then allocates the object copies in the tree rooted at Node i optimally for all string lengths.*

The proof appears in Appendix C.

Theorem 1. *When the algorithm terminates, the allocation of object copies over the network servers is optimal.*

Proof. Proof is conducted by induction where Lemma 1 is the base and Lemma 2 is the step. □

3.4. Complexity

3.4.1. Computational

Since Node i's computation consists of just $d_i + 1$ sums of $|\mathscr{CH}_i|$ elements, clearly the complexity for Node i is

$$(d_i + 1) \cdot |\mathscr{CH}_i|$$

where d_i is Node i's distance from the root and $|\mathscr{CH}_i|$ is the number of Node i's children. The overall complexity of all nodes is

$$\sum_{i\in V} (d_i + 1) \cdot |\mathscr{CH}_i|$$

For $|V| = N$, it is clear that $d_i \leqslant N - 1\ \forall i$. Therefore

$$\sum_{i\in V} (d_i + 1) \cdot |\mathscr{CH}_i| \leqslant N \cdot \sum_{i\in V} |\mathscr{CH}_i| = N(N-1)$$

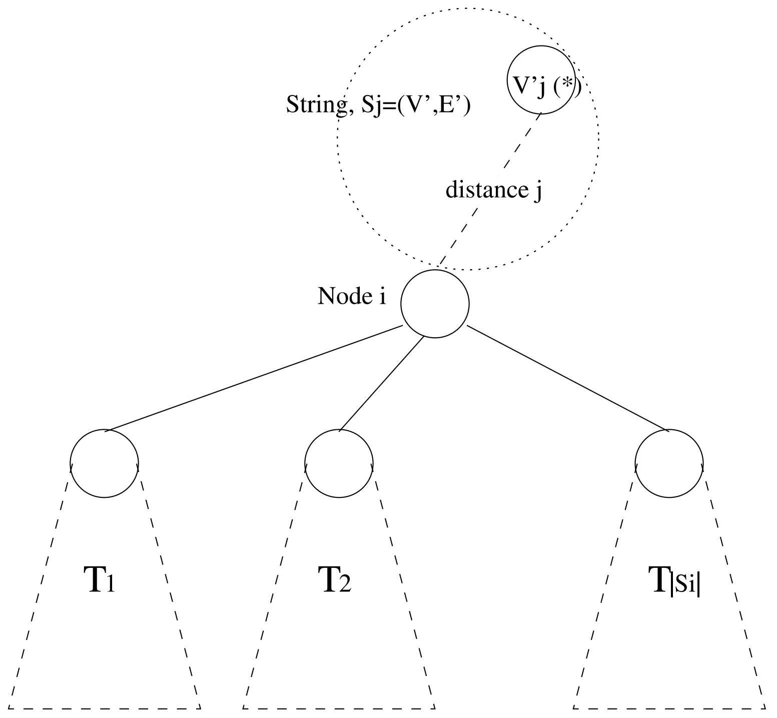

Fig. 4. Example for Lemma 2.

The last equality holds because the total number of children in a tree equals $N - 1$. Hence, the algorithm complexity is $O(dN) = O(N^2)$.

3.4.2. Message

In OLA there are two types of messages. The first is sent by each node to its parent and consists of the cost vector calculated by the node. The second type, *place*, is sent by each node to its children. There are N links and each link passes two messages, one of each type. The message complexity is, therefore, $O(N)$. The longest message is the one with the longest cost vector. The length of the cost vector is determined by the distance (number of hops) from the node to the root. Let $d = \max\{d_i\}$ and $COST = \max_{i,j}\{cost_i[j]\}$. The longest message is d long, and the bit complexity is $O(d \cdot N \cdot \log COST)$.

3.4.3. Time

Assuming that the time to send a message over a link is 1 *time unit*, the running time of the algorithm is $2d$ time units. The time complexity is $O(d)$, where d is the longest distance between the root to a node in the tree.

4. Optimal allocation of servers

OLA optimally allocates copies of an object to a given server location. OLA (with some changes) can also be used to find the optimal allocation of servers to some of the nodes of the network. The idea is to use k strings of nodes attached to the root of a subtree T_i', instead of one string (as was used above). Each string represents the distance of the nearest copy of a different type of object. The details are left to the reader. (For the sake of completeness, one can see the full details in [25].)

5. Conclusion

Further work should include finding a dynamic distributed algorithm in a model that allows a dynamic (i.e., changing over time) request rate for objects. Another issue is that of more general networks. Most of the related problems are NP-hard for general networks. Still, solutions (or good approximations) for networks that are somewhat more general than trees may be useful. In particular, it may be useful to try to superimpose multiple trees on a network (rather than one tree, as is common in practice today), and use the solution for the trees to help with the network problem. There has been a lot of research using this direction for other problems [16].

Another direction is to solve the problem with constraints, e.g. limited capacity at the servers, limited capacity on the links, and bounded distance between a user to a server.

Appendix A. OLA pseudo-code

- $cost_i[j]$—The cost of the sub-tree rooted at Node i.
- $location_i[j]$—The location from which leaves in i's sub-tree obtain the object, assuming i's ancestors are a chain of length j; starting with a server.
- *place*—A message from a parent to its children containing its distance from a server.
- S_{Ci}—The cost of storing one object copy at Node i.
- B_{Ri}—The bandwidth requirements from Link i.
- W_i—The cost per bandwidth unit at Link i.
- d_i—The distance between Node i and the tree root.
- Φ—The set of links between i and the node at distance j.

Algorithm for Leaf (head-end) i

```
* For receive START
  $location_i$ = array[0 : $d_i$] ← here
  $cost_i$ = array[0 : $d_i$] ← $S_{Ci}$ {storage cost at Leaf $i$}
  Do $j \leftarrow 1$ to $d_i$ while ($\sum_{l\in\Phi} W_l \cdot B_{Ri} < S_{Ci}$)
  {while communication cost is less than storage cost}
    $cost_i[j] \leftarrow \sum_{l\in\Phi} W_l \cdot B_{Ri}$
    {comm. cost: getting object from distance $j$}
    $location_i[j]$ ← up
  send $cost_i$ to parent

* For receive place
  {from parent, the distance of the object copy above parent}
  If $location_i[place + 1]$ = here then
    Copy the object at Leaf $i$ {Leaf $i$ prefers a local object copy}
```

Algorithm for non-leaf, non-root Node i

```
  $location_i$ = array[0 : $d_i$] ← down
  $cost_i$ = array[−1 : $d_i$] ← 0, $cost_i[0] \leftarrow S_{Ci}$
  $\mathcal{CH}_i$ ← set of Node $i$'s children

* For receive $cost_c$ from child $c$
  $\mathcal{CH}_i = \mathcal{CH}_i - \{c\}$
  Do $j \leftarrow -1$ to $d_i$ {$i$'s cost = $\sum$(children's costs) + possibly, $S_{Ci}$}
    $cost_i[j] \leftarrow cost_i[j] + cost_c[j+1]$
  If $\mathcal{CH}_i = \emptyset$ then {costs of all children arrived}
    If $cost_i[-1] < cost_i[0]$ then {rely on objects below Node $i$}
      $cost_i[0] \leftarrow cost_i[-1]$,
      $cost\_bound_i \leftarrow cost_i[-1]$,
      $temp_i$ ← down
    Else {better rely on object here}
      $cost\_bound_i \leftarrow cost_i[0]$, $temp_i$ ← here
    Do $j \leftarrow 0$ to $d_i$ {use local server if cheaper than remote}
      If $cost_i[j] \geqslant cost\_bound_i$ then
        $cost_i[j] \leftarrow cost\_bound_i$,
        $location_i[j] \leftarrow temp_i$
      Else $location_i[j]$ ← up
    send $cost_i$ to parent

* For receive place {from parent}
  place ← place + 1
  If $location_i[place]$ = here then
    place ← 0
    Copy the object at Node $i$ {prefering a local copy}
If $location_i[place]$ = down then
    place ← −1 {rely on object below Node $i$}
  send place to all nodes in set of Node $i$'s children
```

Algorithm for the root Node r

```
  $location_r$ = array[−1 : 0] ← down
  $cost_r$ = array[−1 : 0] ← 0, $cost_r[0] \leftarrow S_{Cr}$
  $\mathcal{CH}_r$ ← set of Node $r$'s children
```

```
* For receive $cost_c$ from child c
  $\mathscr{CH}_r = \mathscr{CH}_r - \{c\}$
  Do $j \leftarrow -1$ to 0
    $cost_r[j] \leftarrow cost_r[j] + cost_c[j+1]$
    {$r$'s cost = sum (children's costs) + possibly, $S_{Cr}$}
  If $\mathscr{CH}_r = \emptyset$ then {costs of all children arrived}
    If $cost_r[-1] < cost_r[0]$ then
    {rely on objects below Node $r$}
      $place \leftarrow -1$
    Else
      $place \leftarrow 0$ {Node $r$ prefers a local object copy}
      Copy the object at Node $r$
    send $place$ to all nodes in set of Node $i$'s children
```

Appendix B. Execution example of the OLA

The example is based on the network of Fig. 2, assuming $(\forall i, S_{Ci} = 10, W_i = 1)$.

B.1. Phase 1

In Phase 1 each node calculates the optimal cost for the sub-tree rooted at it, for all distance possibilities for the nearest object copy up the tree. Let us begin with the leaves' tables.

j	0	1	2
$Cost_5$	10	3	6
$Location_5$	Here	Up	Up
$Cost_6$	10	2	4
$Location_6$	Here	Up	Up
$Cost_7$	10	10	10
$Location_7$	Here	Here	Here
$Cost_8$	10	5	10
$Location_8$	Here	Up	Here
$Cost_9$	10	7	10
$Location_9$	Here	Up	Here
$Cost_{10}$	10	6	10
$Location_{10}$	Here	Up	Here
$Cost_{11}$	10	1	2
$Location_{11}$	Here	Up	Up
$Cost_{12}$	10	10	10
$Location_{12}$	Here	Here	Here

Table 1
The parent's table

j	−1	0	1
$Cost_2$	20	15	10
$Location_2$	Down	Here	Up
$Cost_3$	40	38	38
$Location_3$	Down	Here	Here
$Cost_4$	20	20	12
$Location_4$	Down	Down	Up

Table 2
The root's table

j	−1	0
$Cost_1$	73	70
$Location_1$	Down	Here

Each leaf sends its *cost* vector to its parent. The parent receives the *cost* vectors from its children and calculates its own table (see Table 1).

The root ends Phase 1 and starts Phase 2 of the algorithm. The root determines that the optimal cost of the tree is 70 and copies the object to itself because in line 0 *location* is *here* (see Table 2).

B.2. Phase 2

The root sets *place* to zero ($place \leftarrow 0$) and sends it to all its children (Nodes 2–4).

Node 2 receives $place = 0$, increments it by one ($place = 1$), and checks the *location* at line 1. *Location* is *up*, therefore it sends *place* to all its children.

Node 3 receives $place = 0$, increments it by one, and checks the *location* at line 1. *Location* is *here*, therefore it sets *place* to zero ($place \leftarrow 0$), copies the object to itself, and sends *place* to its children.

Node 4 receives $place = 0$, increments it by one, and checks the *location* at line 1. *Location* is *up*, therefore it sends *place* to all its children.

Nodes 5, 6, 11 receive $place = 1$, increment it by one, and check the *location* at line 2. *Location* is *up*, and these nodes are leaves, therefore they stop.

Node 7 receives $place = 0$, increments it by one, and checks the *location* at line 1. *Location* is *here*, and the node is a leaf, therefore it copies the object to itself and stops.

Nodes 8, 9, 10 receive *place* $= 0$, increment it by one, and check the *location* at line 1. *Location* is *up*, and these nodes are leaves, therefore they stop.

Node 12 receives *place* $= 1$, increments it by one, and checks the *location* at line 2. *Location* is *here*, and the node is a leaf, therefore it copies the object to itself and stops.

When the algorithm terminates, the optimal allocation of the object copies is $\Lambda = \{1, 3, 7, 12\}$ and the cost of the optimal location is $\sum_{i\in\Lambda} S_{\mathrm{C}i} + \sum_{i\in E} W_i \cdot B_{\mathrm{R}i} = 40 + 30 = 70$.

Appendix C. Proofs of Lemma 1 and Lemma 2

Proof of Lemma 1. Either one of two possibilities holds.

(1) There is no string connected. OLA allocates the object at Node *i*, which is the only possibility, and thus the optimum.
(2) A string with length *j* is connected to Tree T_i'. If the communication cost is less than the storage cost ($\sum_{l\in\Phi} W_l \cdot B_{\mathrm{R}i} < S_{\mathrm{C}i}$), it is clearly better to use (as OLA does alocate) the copy from the string (*cost* is $\sum_{l\in\Phi} W_l \cdot B_{\mathrm{R}i}$ and *location* is *up*). Otherwise, OLA allocates the object at Node *i* (*cost* is $S_{\mathrm{C}i}$ and *location* is *here*), which is obviously optimal. □

Proof of Lemma 2. We prove by contradiction. Assume that the lemma does not hold, thus, if $COST^{ALG}$ is the cost of the solution $SOLUTION^{ALG}$ calculated by the algorithm, then, for the optimal cost $COST^{OPT}$ of an optimal solution the assumption is that $COST^{OPT} < COST^{ALG}$. For each case, in the following case analysis, we use $SOLUTION^{OPT}$ to calculate solutions for the children of Node *i*. We show that at least one of these solutions has a lower cost than the solution found for the child by OLA. This contradicts the assumption in the lemma that the solutions calculated for the children are optimal.

Either 1 there is no string connected to Node *i*, or 2 there is a string connected.

(1) If there is no string connected, then OLA's allocation (of the value $cost_i[0]$) is the minimum between: 1(a) not allocating an object copy at Node *i*, and 1(b) allocating an object copy at Node *i*. In each of these cases, we consider separately the case that $SOLUTION^{OPT}$ allocates a server at Node *i*, and the case that $SOLUTION^{OPT}$ does not.

(a) Consider the case that the minimum is obtained by OLA by not allocating an object copy at Node *i*,—The *cost* calculated by OLA is, thus, the sum of the *cost*s of the trees that are rooted at the children of Node *i* ($T_1, T_2, \ldots, T_{|\mathrm{S}_i|}$) with no string connected to them ($\sum_{l\in\mathscr{CH}_i} cost_l[0]$). If $SOLUTION^{OPT}$ does not allocate a copy at Node *i*, then, by definition, $SOLUTION^{OPT}$ too is the sum of the costs calculated (say, by some optimal algorithm) for the children. Thus, for $COST^{OPT}$ to be smaller than $COST^{ALG}$, the cost of the solutions for at least one of the children in $SOLUTION^{OPT}$ must be smaller than in $SOLUTION^{ALG}$. A contradiction.

If $SOLUTION^{OPT}$ does allocate a copy at Node *i*, then consider the solutions for the children of Node *i* in $SOLUTION^{OPT}$. Each such child solution is a correct solution for the problem of allocating an object copy to the child's tree, with a string of length 1 attached to the child (since each such solution for a child of Node *i* assumes that there is a copy of the object at the parent Node *i*). The sum $CC(OPT)$ of the costs of these child solutions is smaller than the sum $CC(ALG)$ of the costs of the child solutions calculated by algorithm OLA for the same children with the same string of length 1 attached. (These follows from the facts that $COST^{OPT} = CC(OPT) + S_{\mathrm{C}i}$, and $COST^{ALG} \leqslant CC(ALG) + S_{\mathrm{C}i}$ (since OLA found that the minimum is obtained without assigning an object copy at Node *i*) and the assumption that $COST^{OPT} < COST^{ALG}$.) Thus, at least one of the child solutions computed by OLA was not optimal. A contradiction.

(b) Consider the case that the minimum is obtained by OLA by allocating an object copy at Node *i*,—The *cost* calculated by OLA is, thus, the sum of the *cost*s of the trees that are rooted at the children of Node *i* (T_1,

$T_2, \ldots, T_{|S_i|}$) with a string of length one (S_1) connected to them, and the storage cost at Node i ($\sum_{l \in \mathscr{CH}_i} cost_l[1] + S_{Ci}$). The rest of the proof is very similar to the proof in case 1(a).

(2) If a string of length $j > 0$ is connected, OLA's allocation (of the value $cost_i[j]$) is the minimum between

(a) Using the object copy from the string—The *cost* is the sum of the *cost*s of the trees which are rooted at the children of Node i with a string of length $j+1$ (S_{j+1}) connected to them ($\sum_{l \in \mathscr{CH}_i} cost_l[j+1]$).

(b) Allocating an object copy at Node i (see 1(a)), ($\sum_{l \in \mathscr{CH}_i} cost_l[1] + S_{Ci}$).

(c) Not allocating an object copy at Node i, and not using the object copy from the string (see 1(b)), ($\sum_{l \in \mathscr{CH}_i} cost_l[0]$).

These are the only three possibilities when there is a string of length j connected to Node i. Thus, there are three cases for the minimum calculated by the algorithm, and for each such case, there are three cases for the decisions of the optimal algorithm assumed. For each of the nine cases, assuming that $COST^{OPT} < COST^{ALG}$ implies that the solution of OLA for the children was not optimal, a contradiction to the assumption of the lemma. The proof in each of the nine cases is very similar to the proof of Part 1 above, and we leave the details to the reader. □

Hence, the allocation of the object copies is optimal for the tree T' rooted at Node i, for every length of a string connected to it.

References

[1] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
[2] A. Kolen, Solving covering problems and the uncapacited plant location problem on trees, European Journal of Operational Research 12 (1983) 266–278.
[3] C.N. Judice, E.J. Addeo, M.I. Eige, H.L. Lemberg, Video on demand: a wideband service a myth?, in: Proceedings of the 1986 International Conference on Communication (ICC'86), 1986, pp. 1735–1739.
[4] P.B. Mirchandani, R.L. Francis, Discrete Location Theory, Wiley, New York, 1990.
[5] A.D. Gelman, S. Halfin, Analysis of resource sharing in information providing services, in: Proceedings of 1990 GLOBECOM,1990, pp. 312–316.
[6] S.L. Hardt-Kornacki, L.A., Ness, Optimization model for the delivery of interactive multimedia documents, GLOBECOM 1991.
[7] A. Lombardo, S. Palazzo, G. Schembra, A model for multimedia service creation and activation, IEEE ICC 1994, New Orleans, LO, May 1995.
[8] J.P. Nussbaumer, B.V. Patel, F. Schaffa, Multimedia delivery on demand: capacity analysis and implication, in: Proceedings of the 1994 Local Computer Networks Conference, 1994.
[9] J.P. Nussbaumer, F. Schaffa, On bandwidth and storage tradeoffs in multimedia distribution networks, IEEE INFOCOM 95, Boston MA, April 1995.
[10] I. Korpeoglu, T.-H. Wu, Distributed interactive video system design and analysis, IEEE Communication Magazine, March 1997.
[11] O. Berman, E.K. Yang, Medi-center location problem, Journal of Operational Research Society 42 (4) (1991) 313–322.
[12] A. Karmouch, N.D. Georganas, Multimedia segment delivery scheme and its performance for real-time synchronization control, IEEE ICC 1994.
[13] C. Parris, H. Zhang, D. Ferrari, A dynamic scheme for real-time connection, IEEE INFOCOM 1994.
[14] G. Orphanos, D. Kanellopoulos, S. Koubias, G. Papadopoulos, An integrated application/service platform to support multimedia applications, IEEE ICC 1994.
[15] N. Shacham, Preemption-based admission control in multimedia multiparty communications, IEEE INFOCOM 1995.
[16] B. Awerbuch, D. Peleg, Sparse partitions, IEEE FOCS, 1990, pp. 503–513.
[17] S.E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, L. Wei, An architecture for wide-area multicast routing, ACM SIGCOMM, 1994, pp. 126–135.
[18] S. Aggarwal, J.A. Garay, A. Herzberg, Adaptive video on demand, ESA 1995, Corfu, Greece, pp. 538–553.
[19] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, Protocol Independent Multicast, Internet RFC 2362.
[20] J.P. Nussbaumer, B.V. Patel, F. Schaffa, J.P.G. Sterbenz, Networking requirements for interactive video on demand, IEEE Journal on Selected Areas in Communications 13 (5) (1995) (Also presented at INFOCOM 94).
[21] L.De. Giovanni, A.M. Langellotti, L.M. Patitucci, L. Petrini, Dimensioning of hierarchical storage for video on demand services, IEEE ICC 1994, pp. 1739–1743.
[22] P. Krishnan, Danny Raz, Yuval Shavitt, The cache location problem, IEEE/ACM Transactions on Networking 8(5) (2000) 568–582.
[23] B. Li, M.J. Galin, F. Italiano, X. Deng, K. Sohraby, On the optimal placement of web proxies in the Internet, IEEE INFOCOM'99, pp. 1282, 1290.
[24] A. Tamir, An O(PN^2) algorithm for the P median and related problems on tree graphs, Operation Research Letters 19 (1996) 59–64.

[25] I. Cidon, S. Kutten, R. Soffer, Optimal Allocation of Electronic content, (an earlier draft with additional examples) http://iew3.technion.ac.il:8080/kutten/cks-long.ps.

[26] S. Jamin, C. Jin, A. Kurc, D. Raz, Y. Shavitt, Constrained mirror placement on the Internet, IEEE INFOCOM'01, Anchorage, AK, April 2001.

[27] L. Zhang, S. Floyd, V. Jacobson, Adaptive web caching, The 2nd Web Caching Workshop, Boulder, CO, June 1997. http://ircache.nlanr.net/Cache/Workshop97/Papers/Floyd/floyd.ps.

[28] C. Vassilakis, M. Paterakis, On the distributed delivery of broadband multimedia services to residential/business customers on demand, Technical Report, ECE Department, Tech. Univ. of Crete, 1997.

[29] M.M. Buddhikot, Project MARS: scalable, high performance, Web based multimedia-on-demand (MOD) services and servers, Ph.D. Thesis, Department of Computer Science, Sever Institute of Technology, Washington University, St. Louis, MO, USA, August 1998.

[30] C. Bisdikian, B. Patel, Issues on movie allocation in distributed video-on-demand systems, IEEE International Communications Conference, 1995, pp. 250–255.

[31] C. Bisdikian et al., MLAP: A MAC level access protocol for the HFC 802.14 network, IBM Research Report RC20284, December 1995.

[32] J. Auerbach, M. Gopal, M. Kaplan, S. Kutten, Multicast group membership management in high speed wide area networks, in: Proceedings of the 11th International Conference on Distributed Computing Systems (IC-DCS), Arlington, TX, May 1991.

Israel Cidon is a professor in the Faculty of Electrical Engineering at the Technion. He holds a D.Sc. and a B.Sc. in Electrical Engineering from the Technion (1980 and 1984, respectively). His research interests are in the field of high-speed networks, network architecture, network control, quality of service and distributed algorithms. In 1994 and 1995, he was manager of High-Speed Networking at Sun Microsystems Labs, CA. There, he founded and lead Sun's first high-speed networking research group. Between 1985 and 1994, he was with by IBM Thomas J. Watson Research Center, NY where he served as a Research Staff Member and a Manager of the Network Architectures and Algorithms Group. He was responsible for projects in high-speed networks, switch design, network control and wireless networks. In particular he lead the implementation of PARIS the world first integrated packetized network to carry data, voice and video and the first spatial reuse optical ring. Israel was also instrumental in the technology development of Micronet Ltd. (1981), a vendor of data entry terminals, Viola Networks (1997), a provider of distributed network performance verification and fault isolation software and VersEdge Technologies (1999) a provider of a software solution that accelerate file sharing over enterprise Intranets/Extranets.

He was a founding editor for the IEEE/ACM Transactions on Networking, Editor for Network Algorithms for the IEEE Transactions on Communications and a guest editor for Algorithmica. He is a recipient of the IBM Outstanding Innovation Award for his work on the PARIS project and topology update algorithms (1989 and 1993, respectively). He has authored over 100 papers in leading journals and conferences and 15 US patents.

Shay Kutten received his Master degree (on scheduling of radio broadcasts) and his Ph.D. (on distributed network algorithms, under the supervision of Shlomo Moran) in Computer Science from the Technion, Israel, in 1984 and 1987 correspondingly. From 1987 to 1996 he was with IBM T.J. Watson Research Center, as a post doctoral fellow, as a project leader, as the manager of the Network Architecture and Algorithms group, and as a Research Staff Member. He led the network security project which developed the security architecture for several IBM products, and developed algorithms for network control, security, and distributed processing control, that were later used in IBM's products. Many of his contributions were patented.

He received (in 1993) the IBM Corporation Outstanding Innovation Award (OIA) for his work on distributed control protocols that were basis to the distributed control of IBM Broad Band Networking Services architecture (NBBS, IBM's ATM). In 1994 he received the IBM Outstanding Innovation Award (OIA) for his work on authentication protocols and his contributions to IBM's network security and NetSP (by itself an award winning product). The same authentication protocols influenced the later development of the Internet payment protocol, so IBM had to grant a no-fee licence (see Internet RFC 1822 in the RFC Editor home page) for the patent granted for some of them, so that the Internet can adopt this payment protocol. Shay Kutten also contributed to the theory of distributed computing, mostly by introducing new theoretical subjects, many of them inspired by his work on practical issues, and by giving well founded solutions to practical problems.

Currently he is an associate professor of Information Systems, and Area Head of the Information Systems Area, at the Davidson Faculty of IE & M at the Technion. He is a senior member of the IEEE, an area editor (for security, reliability, and availability) of the ACM's journal on Selected Topics in Mobile Networks and Applications (MONET), and was a member of the editorial board of the ACM's Wireless Networks journal. He served on program committees for several conferences and workshops, was the chairman of the program committee of the 1998 DISC conference (the 12th annual international symposium on Distributed Computing, formerly WDAG), and was awarded several additional awards and research grants.

Ran Soffer. Ran graduated his M.Sc.-EE at the Technion (Israel institute of Technology) at 1998. He specialized with algorithms for networking communication. Since 1998 Ran held several marketing positions in IC communication vendors.