

ON THE VIRTUAL TOPOLOGY OF LOOSELY COUPLED MULTI-COMPUTERS

French Investigators

Cyril Gavoille (PI)

Laboratoire Bordelais de Recherche en Informatique (LaBRI), Université de Bordeaux.

Pierre Fraigniaud

Laboratoire d'Informatique Algorithmique (LIAFA), Université Paris Diderot (Paris 7)

Emmanuelle Lebhar

Laboratoire d'Informatique Algorithmique (LIAFA), Université Paris Diderot (Paris 7)

Amos Korman

Laboratoire d'Informatique Algorithmique (LIAFA), Université Paris Diderot (Paris 7)

Israeli Investigators

David Peleg (PI)

Department of Computer Science, Weizmann Institute

Shay Kutten

Faculty of Industrial Engineering, Technion

Abstract

The proposed research addresses the use of networked computers as a large-scale multicomputer, that is, a tool for parallel and distributed computing. Informally speaking, it deals with realizing the vision “The network is the computer” (McNealy, Sun Microsystems). We intend to address a broad definition, whereby a multicomputer is a generalization of a single computer, namely, a general purpose machine that performs scientific computations but can also be used as part of an information system, or as a communication unit, or for control and alerting (when certain events happen), or for obtaining, managing and presenting video data, and so on. Just as parallelism is an essential prerequisite without which supercomputing becomes infeasible, we argue that the use of both parallelism and distribution enables multicomputers to perform tasks that would be infeasible or impossible using a single traditional (say, multi-terminal) computer. We intend to address all principal components of the multicomputer architecture, including storage, computing, data flow, and communication. To handle the topology of a large-scale distributed multicomputer, we borrow and adapt a rather recent concept from the field of networking, namely, *overlay networks*, which are *virtual* higher layer networks. The design of the multicomputer, then, involves the design of the virtual topology of its overlay network, including placing resources (e.g., specific files), distributed assignment of tasks (scheduling, load balancing), defining virtual links (e.g., for routing the multicomputation tasks), and adapting to topological changes (e.g., components joining and leaving). We intend to design a collection of algorithms for these tasks, that will complement each other cooperatively, so as to support the creation of a distributed multicomputer.

1 Introduction

Motivation: This research addresses the use of networked computers as a large-scale multicomputer, that is, a tool for parallel and distributed computing. Informally speaking, it deals with realizing the vision “The network is the computer” (McNealy, Sun Microsystems).

A narrow approach to parallel computing may focus on addressing one specific application of such computers. A common such example involves a parallel machine intended to be used as a supercomputer for computation intensive tasks, e.g., weather forecasts. We intend to address a broader definition, whereby a multicomputer is a generalization of a single computer, namely, a general purpose machine that performs scientific computations but can also be used as part of an information system, or as a communication unit, or for control and alerting (when certain events happen), or for obtaining, managing and presenting video data, and so on. Just as parallelism is an essential prerequisite without which supercomputing becomes infeasible, we argue that the use of both parallelism and distribution enables multicomputers to perform tasks that would be infeasible or impossible using a single traditional (say, multi-terminal) computer.

Let us briefly review and discuss some aspects of such a generalized computer, focusing on the impact of topology considerations.

Storage: In a traditional computer, certain decisions must be made as to where data should be located. For instance, data can be located in a fast cache, or in the main memory, or in some secondary memory, etc. Even when the data is located on a slower disk storage, it is still necessary to reach further decisions as to whether to place certain pieces of data near each other or spread them around, or as to whether a certain file should be stored in consecutive memory locations, etc. For a networked multicomputer, we shall investigate the analogous question of data placement in the network storage. Here, one can decide on questions such as which of the actual machines that compose the multicomputer should host the data, when should the data be moved or replicated, and when should a replica be destroyed, and so on. Note that locations in the network are really a part of the network’s topology.

Computing: For a traditional computer, decisions are made (e.g., by the operating system) on when to allocate the processor to which task. In our context, it is required to make scheduling decisions on an entire distributed system. Such scheduling decisions are heavily affected by the topology of the system. For example, the distance of components from each other affects the decision on whether a task should be moved from its current location, and if so, where to. Clearly, this is also affected by the location of the data (and data servers). Hence, it is impacted by the topology.

Data flow and communication: The performance of a traditional computer is strongly impacted by the internal communication performed among its various components. In the generalized computer addressed here, there are more components, and they are further away from each other, hence the issue of communication efficiency becomes even more important. The topological issues here involve, for example, decisions on which pairs of components can communicate directly with each other.

There are various other differences between the design of a multicomputer and that of traditional one. For example, in a networked multicomputer, the topology can be much more dynamic, as components can be added and rearranged in software, just by changing some decisions online (for example, by establishing another communication connection).

To handle the topology of a large-scale distributed multicomputer, we borrow and adapt a rather recent concept from the field of networking, namely, *overlay networks*, which are *virtual* higher layer networks. For example, a *virtual* point-to-point link ℓ in an overlay network can be mapped into some collection of services in the *underlying* communication network. Typically, the link ℓ will be mapped into an end-to-end connection over a path consisting of multiple links in the underlying network. However, there are also other, more involved, scenarios where communication over one virtual link is mapped into multiple paths, or into a multicast service of the underlying network. Hereafter, such virtual links are termed *overlay links*, and similarly we use the terms *overlay node*, *overlay message* etc. Analogously, an

entity of the underlying network is termed an *underlying entity*. An overlay node typically resides in an underlying node, but may sometimes be relocated, or changed in some other way (e.g., its storage may be increased by mapping it to actual storage in two or more underlying nodes). An overlay message is typically translated to multiple underlying messages, and so on.

The task of designing overlay networks is inherently different from that of designing a single physical machine (or even a tightly coupled physical parallel machine), as well as from the task of designing the underlying networks. First, the *available tools* are different. For example, in the context of an underlying network, or even a rigid (tightly coupled) parallel machine, the addition of an edge may be impossible, or at least is very expensive, slow and seldom performed. The decision on such a change is done not by automatically operating distributed protocols but rather by people (aided by centralized optimization algorithms) who must consider also the extra expense. Thus, this design task has only a very indirect interaction with, say, the task of load balancing in the parallel machine. On the other hand, in overlay networks, establishing a new edge can be considered a local action of a node, and performed often and rather quickly. Thus, it can (and should) be optimized together with other on-going tasks, such as load balancing or routing. Similarly, consider the operation of algorithms for learning the topology (e.g., for the purpose of making informed decisions on implementing connection changes or message routing). The knowledge available to such an algorithm operating in an overlay network may be different in nature than that available to an algorithm operating in the underlying network, or the internal network or switch of a parallel machine. In the physical context, the algorithm may measure the actual traffic and sense the physical device to decide whether it is operative. It can also force a message to follow the route of a previous message, to find out whether the previous route is still operational. Such methods of knowledge acquisition may not be available at the application layer, where the overlay network of the multicomputer is typically defined. Moreover, the owners of the underlying networks may be different from those of the overlay networks, and may not wish to share their knowledge about the state of the underlying system.

In addition, the *design goals* relevant to underlying networks, or computer hardware, may be very different from those related to a multicomputer overlay. As an illustrative analogy, note that the task faced by a group of high schools kids who design a messengers “network” for home distribution of newspapers (corresponding to an overlay network) is inherently different from the task of designing a physical network of streets and highways (corresponding to the underlying network). Similarly, designing a distributed multicomputer for the parallel / distributed computing task of a publish-subscribe system (cf. [1]) involves decisions that are different in nature from those taken in the design of the underlying networks or parallel computers.

Multicomputer Overlay Networks: Design and Update: The task of topology design involves the selection of components and connections between them. In particular, it is necessary to decide how many different servers are needed for a given computational task and where those servers should be allocated. It is also necessary to allocate the contents and the tasks among the various servers (or grid components). Then, the connectivity between the components needs to be defined, including its type (unicast, multicast, VPN), pattern (which node is connected to which), fault tolerance (e.g., redundancy), and QoS (required bandwidth, delay etc.). Finally, one must define how to connect the consumers to the service infrastructure and by what means.

These parameters can sometimes be defined statically, but can also be changed adaptively according to (a) the traffic and load pattern changes, (b) component failures and recoveries, (c) the growth or shrinkage of the network, e.g., due to consumers joining or leaving it. Thus, the task of topology design is complemented by that of *topology update*. This involves sharing information about environment and topology changes (either as a result of events occurring in the environment or as a result of corrective actions taken by the dynamic topology design process). We would like to investigate these two complementing tasks, as well as the feedback processes between them. We term the combination of these tasks *topology control*. Algorithmic issues also arise when considering the interaction and feedback between the

topology control and other tasks performed by the network.

Some of the above aspects have been investigated in a more limited setting (for underlying networks), as detailed below.

Background: A problem related to topology design is that of *object allocation*. This problem has been studied extensively outside the context of multicomputers or even of overlay networks. In the context of content delivery overlay networks, the problem was studied in [22]. Additional studies address the server location problem given a fixed number of servers [23, 24, 25, 26] and objects [27].

An important part of topology design involves deciding on the addition of new edges to a given network. Many variants of this problem were studied in the context of underlying networks (cf. [13, 14]). As explained above, this task is very different from the design of the graph for overlay networks. More directly related are studies on connection establishment in fast (e.g., optical) networks, see e.g. [18, 16, 15]. A similar task exists in other types of networks that have means for fast switching, see e.g. [17, 21, 19, 20].

The topology update problem is inherent in distributed networks. As mentioned, the topology may change from time to time as a result of external events (e.g., failures), and for a network to continue functioning it is required to learn the new topology (see, e.g., [4, 10]).

Clearly, algorithms for this task too must be different in overlay networks from their counterparts in underlying networks. In addition, for overlay networks it often makes sense to combine the two tasks of topology control (update and design), and change the topology simultaneously with learning it. This approach is demonstrated in, e.g., [5, 7, 8, 9]. A formalization of the problem for a rather static case was suggested by Harchol-Balter, Leighton and Lewin in [5]. The specific P2P application they had in mind was the logical networks of servers placed by Akamai Technologies in various Internet sites. When given another peer's identifier (e.g., IP address) $ID(v)$, a peer u can send messages to v "directly" (e.g., by a UDP message). Hence, a logical (overlay) directed edge $\ell = \langle u, v \rangle$ represents the fact that the peer u at ℓ 's tail knows the identifier $ID(v)$ of v . The initial overlay network (capturing the nodes knowledge) is represented by a directed graph $G_0(V, E_0)$ (i.e., each u knows only $ID(u)$ and $\{ID(v) \mid \langle u, v \rangle \in E_0\}$). The graph G grows as nodes learn the addresses of non-neighboring peers. A node u can learn $ID(v)$ by receiving a message from v itself or from some other node w who knows $ID(v)$. Conceptually, this adds the directed arc $\langle u, v \rangle$ to the edge set. The task of the learning algorithms of [5] was to gradually transform the initial overlay network into a complete graph.

Note that in the described model, knowledge implies connectivity, hence the topology update and topology design tasks are combined in [5]; in a more general setting, an overlay link may represent additional requirements, e.g., the presence of a permanent TCP connection. Since maintaining a TCP connection has a cost, the designer may choose not to have an overlay link between some nodes u and v even after the topology update process has learned the route that connects them. Hence, the topology update and topology design processes may behave differently.

A number of (randomized and deterministic) algorithms are presented in [5] for transforming a graph into a complete graph in synchronous networks. A stronger model (with improved complexity) is treated in [6]. A deterministic distributed algorithm (with improved complexity) for the weakly connected case is presented in [7]. The asynchronous case is treated in [9, 8].

A more general model is presented in [2]. For the *undirected case*, the message complexity of the algorithm of [2] is optimal $O(n)$, but the time complexity is very high - $O(n)$. (This leaves open the question of a combination of a better time complexity together with a better message complexity for the *undirected case*).

In fact, [2] presented a whole family of models that may be applicable to the research proposed here, despite having been introduced in a different context. Those models represent the fact that in modern networks the explicit cost of handling of a message by an upper layer process in a general purpose computer increased, while the cost of performing the lower layers of the communication decreased. A similar phenomena can be seen in the context of the current research. The underlying network can

perform many task more efficiently than the overlay network, but some tasks must be performed by the overlay nodes (since the application designer has little control over the underlying network). Similar models to those of [2] were also suggested in the context of parallel computing, cf. [12, 11].

2 Research Objectives and Expected Significance

The introduction of the concept of layers in a network simplified the task of designing network architectures tremendously, reducing complexity and helping to set standards. In a similar manner, overlay networks are now becoming the inherent enabling architecture of parallel and distributed computing. The technology of large systems dictates the layering of the networks themselves (as opposed to the previous approach of layering inside one network) in order to provide the necessary functionality, flexibility, service quality and reliability to a dynamic population of end users. Tools and concepts for the architecture of overlay networks are thus very much in need. In fact, in overlay networks this need may be even higher than in traditional networks, because of the higher complexity. An additional benefit is that a good architecture may enable a useful economical model, that already starts to develop. Recall that traditional underlying networks may have been composed of layers that belonged to the same network, according to technologies that matched each other. Of course, this was also the case with a traditional single, or even parallel computer. As opposed to that, economical models encourage the development of overlay networks from mix and match components, and in very complex ways.

To be able to handle such networks, it is required to change the way network architecture design and topology control are conducted. First, in order to be efficient, the overlay network connectivity patterns need to be highly dynamic, capable of changing on demand in reaction to the on-going changes in consumption patterns, underlying network behaviors and vendor pricing changes.

In addition, the components of the overlay network itself may include dynamic elements. A simple example is the need for proactive movement and replication of tasks and media objects from node to node in order to cope with changing demands. A yet more complex issue is the need to add and release on demand overlay network components (links or nodes) that are either shared with other overlay networks or are leased from a third party (say, adding storage nodes from an SSP or leasing computational resources from another service). In this model, the capacity and abilities of nodes can also change on demand.

Our major goals in this research project are the following. We would like to address the more specific problem of topology control for distributed networked multicomputers. The approach for addressing this problem should rely on introducing novel algorithmic concepts by which the overlay network computer interacts with the underlying networks and its peers, and peers interact with each other. This problem is one of the most visible differentiators in the ability to design overlay network based multicomputers that are more flexible, reliable and cost effective than dedicated, static, specially designed solutions. Hence efficient solutions for this problem are of considerable practical significance.

To achieve this goal, we shall identify different features for the topology design that should exist for different applications. We shall also identify differences that should result from different forms of design. For example, a more centralized design may be suitable for overlay networks that are more stable and that belong to some single authority, whereas a more distributed design may be more appropriate for “grass roots” and highly dynamic systems.

Finally, we also plan to address the basic architectural concepts of overlay and layered networks and the subtle interrelations between them. Once we understand the type of interactions and information flow that is required between the various layered networks, we can develop the array of information and control exchange protocols required for the standardization of such complex systems buildup.

3 Detailed description of the proposed research

Dynamic learning of changes and algorithms for link addition and deletion We have already addressed some of the simpler scenarios of this problem in the past. For example, in [7] we developed a distributed algorithm for adding virtual edges distributively so that a weakly connected overlay is transformed into a star graph. This could be useful for the task of routing under some very favorable conditions, as each peer can ask the star center for the route to any other peer.

We intend to address additional building blocks of the multicomputer, beside simple routing. Moreover, we intend to address more realistic scenarios, overcoming the weaknesses of the algorithm of [7]. In particular, that algorithm does not adapt to online changes introduced by the environment. For example, it does not handle peers who leave or join. Second, the communication cost model assumed therein is that a peer can communicate with all its neighbors at the same time in parallel. This is not realistic for an overlay network, as in reality, concurrent messages cause *congestion*. That is, a peer has only a limited (normally, even constant) number of communication interfaces (network adaptors), and sending messages to multiple virtual neighbors takes longer than sending a message to just one. Congestion may also occur on the receiver side, as processing messages from multiple virtual neighbors takes longer than processing just one. A related problem was addressed in [28]. It complements [7] in the sense that it reaches some desirable virtual topology (in [28] a ring), however, it also deals with congestion, by taking congestion into account in calculating the time complexity.

Congestion is one reason why a complete graph may not be the most desirable virtual topology for the overlay network. Another reason is the dynamic nature of the network. Since the topology may often change by external events (e.g. failures), the network might never be brought into a desirable optimal topology. Thus, it may make sense to look for an intermediate topology, offering a balanced trade-off between quality and ease of change. Such a topology should on the one hand be relatively easy to reach after failures (compared to reaching an optimal topology according to the desired measures). On the other hand, this intermediate topology should be “reasonably good” (though maybe not optimal) according to those measures. In this research we shall try to propose intermediate topologies attaining such a trade-off.

To handle a dynamic network, one needs to define how do peers notice, for example, that a peer failed. In a tightly coupled cluster, a failure is detected already on the hardware level. In a traditional network, a node may monitor its neighbor by periodically sending test messages. With a large number of potential virtual neighbors, monitoring all the neighbors is costly (due to the high cost of sending or receiving messages in parallel). Thus, as part of the virtual topology design one should decide on the topology of the “monitoring graph” (where an edge $\langle u, v \rangle$ represents the fact that u monitors v).

Let us next consider other functions, beside routing, that should be supported by overlay networks for a distributed multicomputer. A computational process performed on a multicomputer needs to take care also of load balancing, scheduling, privacy, efficiency of the data flow, and fault tolerance. (Here we refer to failures in the stored information or in the performed tasks, as opposed to failures in the topology of the multicomputer itself, addressed above.) All of those functions can be supported by making the correct decisions concerning the construction and maintenance of the right virtual topology. For example, a suitable virtual topology can help in distributing the work load. However, this should be balanced against the extra work needed to maintain the virtual topology. We intend to define optimization criteria for the maintenance of the virtual topology required to support the various functions of the multicomputer, and their inherent trade-offs. We shall then devise efficient algorithms and protocols to optimize them.

Let us list some additional extensions for the models of [5, 7, 6] needed in order to yield more realistic topology design and update algorithms.

- Recall that in [5, 7] a logical edge directed from node u to node v , representing in v “knowing $ID(v)$ ”, intuitively meant knowing v ’s IP address. More generally, such knowledge may include routing and access-related information (such as passwords). It is easy to transmit the knowledge of the IP address

of v from a node u to some other node w . On the other hand, some other kinds of knowledge are not that easy to transfer. For example, in the case of policy based routing (e.g. [3]), not every route that is valid from u to v is also valid from w to v . Another example for extra difficulties is the case that cryptographic protocols are used in the routing. In a still more general case, the setting of a new link between w and v involves operations not only in w and v , but also on the way.

The models we intend to develop will have to take such considerations into account. The construction of a multicomputer may need to take such considerations into account, since they will impact the efficiency of the routing in the multicomputer.

- In [5, 7, 8], the underlying communication network is modeled as a complete undirected graph over the set of nodes V . In the proposed research we intend to take the actual structure of the underlying network into consideration. For example, an edge between nodes that are far from each other in the actual networks, will be considered more costly than an edge between nodes that are close to each other in reality.

Dynamic learning of data flows and algorithms of relocating data Consider a pub-sub scenario, as an example to a parallel / distributed computation. In practical scenarios, the network architecture evolves as content is added and removed, users join and leave the service and infrastructure changes. Moreover, the demands of some user for various objects change over time, the underlying communication network itself may introduce failures and suffer a slow-down, or alternatively, there may be opportunities to upgrade services either on demand or permanently. Therefore, content needs to be relocated in a dynamic and adaptive manner, and bandwidth should be leased and released as demands change. Here, the overlay architecture can offer some mechanisms that may dramatically improve the cost and the responsiveness of the network in reaction to such changes. Content can be moved and removed dynamically as a reaction to the overlay climate.

This process of modifying the overlay can interplay with the design of the underlying network. Links between the overlay network computer nodes can add and release capacity or can be totally dropped. The total cost of ownership can be reduced while maintaining high quality of service and high availability of content.

Interactions between the overlay multicomputer and the underlying network The following example demonstrates another kind of topology improvement tasks we intend to handle, which is unique to overlay networks, as opposed to underlying networks. Consider a critical computational task performed on three machines of the multicomputer, located in Tel Aviv, Nice and Paris, and suppose that those machines are connected by a network with two overlay edges, (Tel-Aviv,Nice), and (Nice,Paris). For fault tolerance, the computer architect can order another overlay edge, connecting Paris and Tel Aviv, thus seemingly establishing a bi-connected network. However, the provider of the underlying network may construct the new overlay edge over an underlying route that passes in Nice. The result would be that one crash in Nice will disconnect the overlay network even though it is seemingly biconnected.

Now let us make the strong assumption that the multicomputer architect knows (1) the structure of the underlying network, (2) the routing policy of the underlying network and (3) the value of the parameters used in (2). For example, for (2) assume that the routing policy is shortest path. For (3) assume that the weight of edges (appearing in the calculation of the shortest path) does not depend on the load of the underlying network (load possibly coming from other clients, besides the multicomputer). In this case, the computer architect can calculate whether it can expand the overlay network by buying another node, along with communication edges for that node, so that its network is not disconnected even if the underlying network suffers one failure.

This opens a vast area of research problems we intend to investigate in this study. The simplest problem requiring the design of an algorithm may be the task described in the previous paragraph. We are going to define and solve also more realistic problems. Such problems take into account (1) more

involved routing policies, (2) underlying network providers who do not wish to disclose many details about their network, (3) changes in the overlay and in the underlying networks. Of course, this kind of interaction between the design of networks in different layers should be studied for many other graph design problems, not just the construction of a new path that is edge disjoint. The dynamic nature of overlay networks (as well as that of the underlying networks) adds a new dimension we intend to study. Another problem is that of a combination including more than two networks.

Note that one may pose many similar “graph improvement” problems, e.g. make the graph of the multicomputer 3 connected; make the graph have a low diameter, etc. A more general question is to develop a kind of an automatic compiler. Given any “graph improvement problem” can it be solved, given what we know of the underlying network?

In addition, examining the problem from the underlying network provider’s perspective, one may raise the question of the types of information and design primitives that should be supplied by the underlying networks for ensuring efficient design of the overlay network and multicomputer architecture. For example, should the underlying network supply a primitive of “add a disjoint path”? Supposedly, this helps the underlying provider conceal its network details. However, what should the price for such a primitive be? If the price is different for different underlying topologies, does the underlying provider expose the underlying topology just by giving its cost quote? Should there be a standard for overlay topologies provided by underlying networks? Is there a set of topology enhancement primitives for such a standard that can enable overlay designers design a network to their liking, and compare prices between different underlying providers?

Finally, similar questions, regarding the interaction between the overlay and the underlying networks, can be asked also for the more general problem that includes both the graph topology design and the server topology design.

About the researchers

The PIs and the co-investigators are all located in universities in France and in Israel, and enjoy standard research facilities common in such universities. Past collaborations of the French PI and the Israeli PI have yielded quite a few papers already, and there have also been joint studies and papers with the co-investigators. Some of the co-investigators are rather young scientifically (one is 2 years after PhD.) The research subject is close the research areas where the proposing researchers are known as leaders.

The researchers have cooperated in the framework of a COST European program. This program helped the researchers meet in the past. It is sometimes a good base for applying to an ordinary European grant.

References

- [1] M.K. Aguilera, R.E. Strom, D.C. Sturman, M. Astley and T.D. Chandra. Matching events in a content-based subscription system. *PODC 1999*, 53-61.
- [2] I. Cidon, I. Gopal, and S. kuttan. New Models and Algorithms for Future Networks. *IEEE Trans. Inf. Th.* **41**, (1995), 769–780.
- [3] Cisco Corp. Policy-Based Routing. <http://www.cisco.com/warp/public/cc/techno/protocol/tech/policy-wp.htm>.
- [4] I. Cidon, I. Gopal, M. Kaplan, and S. kuttan. Distributed Control for Fast Networks. *IEEE Trans. on Commun.*, **43**, (1995), 1950–1960.
- [5] M. Harchol-Balter, T. Leighton, and D. Lewin. Resource Discovery in Distributed Networks. *15th ACM PODC 1999*, 229-237.
- [6] C. Law and K-Y. Siu. An $O(\log n)$ Randomized Resource Discovery Algorithm. *PODC 2000*, 5–8.

- [7] S. Kutten, D. Peleg, and U. Vishkin. Deterministic Resource Discovery in Distributed Networks. *SPAA*, 2001, 77–83.
- [8] S. Kutten and D. Peleg. Deterministic Resource Discovery in Asynchronous Overlay Networks. *Computer Networks*, **51**, 190–206, 2007.
- [9] I. Abraham and D. Dolev. Asynchronous resource discovery. *ACM PODC 2003*, 2003, 143-150.
- [10] J. McQuillan, I. Richer, and E. Rosen. The New Routing Algorithm for the Arpanet. *IEEE Trans. Commun.* **28**, (1980), 711–719.
- [11] D.E. Culler, R.M. Karp, D.A. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. *Principles Practice of Parallel Programming*, 1-12, 1993.
- [12] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the Postal model for message-passing systems. *4th ACM SPAA*, 11–22, 1992.
- [13] D. Naor, D. Gusfield and C.U. Martel. A Fast Algorithm for Optimally Increasing the Edge-Connectivity. *FOCS 1990*, 698-707.
- [14] A. Kershenbaum, P. Kermani and G. Grover. MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing. *IEEE Trans. Commun.*, COM-39, 503-513, 1991.
- [15] O. Gerstel, G.H. Sasaki, S. Kutten and R. Ramaswami. Worst-case analysis of dynamic wavelength allocation in optical networks. *IEEE/ACM Trans. on Networking* **7**, 833-846 (1999).
- [16] O. Gerstel and S. Kutten. Dynamic wavelength allocation in all-optical ring networks. *ICC 1997*, 432-436.
- [17] O. Gerstel and S. Zaks. The Virtual Path Layout Problem in Fast Networks. *PODC 1994*: 235-243.
- [18] S.C.M. Ram and G. Mohan. *WDM optical networks: concepts, design and algorithms*. Lavoisier, 2001.
- [19] J-C. Bermond, N. Marlin, D. Peleg, and S. Perennes. Directed virtual path layouts in ATM networks. *12th DISC*, 1998, 75-88.
- [20] D. Peleg and U. Pincas. Virtual path layouts optimizing total hop count on ATM tree networks. *J. Discrete Algorithms* **3**, (2005), 101-112.
- [21] I. Cidon, O. Gerstel, and S. Zaks. A scalable approach to routing in ATM networks. *WDAG 1994*, 209-222.
- [22] I. Cidon, S. Kutten, and R. soffer. Optimal allocation of electronic content. *INFOCOM 2001*.
- [23] L. Qiu, V. N. Padmanabham and G. M. Voelker. On the placement of web server replicas. *INFOCOM 2001*.
- [24] S. Choi and Y. Shavit. *Placing Servers for Session-Oriented Services*, Wucs-CS, 2001.
- [25] S. Jamin, C. Jin, A.R. Kurc, D. Raz and Y. Shavitt. Constrained Mirror Placement on the Internet. *INFOCOM 2001*.
- [26] P. Krishnan, D. Raz and Y. Shavitt. The Cache Location Problem. *IEEE/ACM Trans. on Networking*, **8**, 568–582, 2000.
- [27] J. Kangasharju, J. Roberts and K. Ross, Object replication strategies in content distribution networks. *WCW 2001*.
- [28] D. Angluin, J. Aspnes, J. Chen, Y. Wu and Y. Yin. Fast construction of overlay networks. *SPAA 2005*, 145-154.