DRAFT: Shay Kutten- Research Statement

October 2007

I study distributed algorithms of networks (especially asynchronous networks). This is part of the area of distributed computing that includes many subareas such as applications of combinatorics in graphs and networks, systems, shared memory systems, impossibility and lower bounds, among others. While many researchers concentrate on pure theory, or systems, I pursue the goal of solving practical problems using theory. Section 1 below describes various theoretical concepts and problems I introduced and studied derived from practical issues. Some of those had an impact on networking systems too.

Section 2 elaborates on one such subject I was an initiator of, especially in the context of fault tolerance. Note, that a major characteristic of distributed systems is the fact that each component has only a partial view of the system. The paradigm of *locality* is devoted to designing algorithms whose decisions in each component, or the cost of reaching the decision, depend only (or mostly) on the component's neighborhood, rather than on the whole (*global*) network. Many of the locality concepts and results I presented were in the context of fault tolerance, also discussed in Section 2.

Section 3 elaborates on an even more specific set of contributions. Trees are a very important subject in computer science in general, and distributed trees play even a more critical role in distributed network algorithms. Section 3 lists optimal and first results in a comprehensive set of distributed trees algorithmic issues.

Before listing past contributions, let me mention two related research emphasises in my current work. Still pursuing the border of theory with practice, I am addressing new and emerging communication paradigms, while, at the same time, striving to bring to maturity paradigms I have studied before (especially, that of locality) .

In spite of the large body of work on locality, especially in the last several years, there is still much to do to make the theoretical solutions accessible to the practical communities. Initial solutions often use very simplified models. I am now considering additional practical issues and problems. For example, I address the generalization of solutions from a synchronous model to an asynchronous one, which is more realistic. This can be helped by a specific practical context- that of sensor networks. This context is still emerging, and there is a greater chance to influence its architectures. (This study is performed with the aid of a grant from the Israeli ministry of Science and Technology).

In terms of emerging paradigms, I intend to study emerging application layer paradigms (in the past, I studied mostly the network layer) such as *P2P* and *overlay networks*. Those are based partially on networking ideas and partly on parallel computing ideas. I am very interested in applying distributed network algorithms approaches for those paradigms. My goal here, is to increase the parallelism (and thus, the throughput) by avoiding congestion. Similarly, some older semi parallel models are now becoming more practical, for example, *shared memory*. I am trying a similar approach for these models too- developing methods for avoiding congestion, or contention. (This project is performed with the aid of grants from ISF- the Israel Science Foundation and from the SUN corporation).

1. Capturing Networking Issues by Formalizing Models and Research Problems

When I begun working in this field, the theory of computer networking was mostly a part of EE. A large part of my research has been devoted to formalizing models and research problems, mostly derived from real life. My intent was to apply theoretical computer science to networking. At the time I was working for IBM Research as a researcher and as a manager for Network Architecture and Algorithms and for Network Security, this work materialized as products. In the process I won several IBM's awards.

journals later) I presented a graph theoretic model for optimizing scheduling of radio transmissions, especially for the task of radio broadcast, i.e., the task of delivering a message to all the nodes. See J[1,2] (items [1] and [2] in the attached list of journal papers) and C[4] (see attached list of conference papers; in this research statement, I refer only to those that did not appear in jorunals). This has since become a thriving and active area of research, and much of the literature in this area today follows the model developed in J[1,2].

In J[5], I demonstrated that distributed algorithms can be simplified by describing them in terms of processes that can move from one node to another, rather than in terms of messages between static processes. The use of such "mobile agents" became very common in practice in the nineties, but was not known at the time of that study. (Hence, the terminology I used is different than the one that caught eventually). Another simplifying contribution was a demonstration that a modular composition of algorithms is possible and natural even when algorithms are distributed.

In 1987, I joined the Network Architecture and Algorithms- then a small group in IBM that was working on designing the hardware of a fast network. I then started work on the design of the control protocols to go with that architecture. Our ideas initiated a large research and development effort that materialized later, first in the experimental fast networks PARIS and (the national NSF experimental network) AURORA and subsequntly in the commercial architecture and set of products NBBS, as well as in IBM's furst storage area network. IBM sold its networking division and intellectual property (to CISCO), but the architecture of NBBS had a noticeable impact on later fast network architectures, e.g. on MPLS. For parts of this work, I received IBM's awards for Outstanding Innovation and for Outstanding Technical Achievement. For these algorithms, I suggested changing the way the complexity of protocols was evaluated. Previously, mostly the number of messages was counted. Starting with PARIS, the newer networks were based on the ideas of [Turner], that is, they perform the message switching in a specialized hardware. Hence, it did not make sense to attribute a high cost to the number of hardware switched messages. Instead, I suggested to attribute cost to the cases where the general purpose computer had to be involved too (e.g., the reception of a message at its final destination, rather than at intermediate nodes). This allowed optimizing the protocols on one hand, and helped to decide which functions to put in the hardware switch on the other hand. This, and similar models, were studied later by prominent researchers. As a researcher, and later as the manager

of the group, I was responsible for various other networking projects and consultation to IBM units J[11,13,19,31], P[5,6,7] (see patents).

I founded the Network Security Project, which became the Network Security Group and has grown to be a large department in IBM Research today. We identified problems in symmetric protocols for authenticating communicating parties to each other. These protocols were used used in sensitive products used for vital banking communications, and have been discussed in classic textbooks. We demonstrated bugs in them, suggested new design principles and new protocols, and laid down a foundation for proving such protocols correct J[7,12,16]. IBM granted a royalty free license to patent P[1] so that its principles could be used in the Internet Payment Protocol and other Internet authentication. This area of research is still very active and our work is explained, for example, in Tanenbaum's classic networking textbook. Other patents were granted and used for security servers P[4,11], secure content P[2], message authentication P[9,13], rights protection P[12,10], and more. I received the IBM Outstanding Innovation Award for the authentication protocols, the IBM Research Award for the work on the security of IBM wireless networks, P[2,3,8,13,14], and several IBM patents awards.

Another issue I raised is the *competitiveness* of distributed algorithms. Previously studied *competitive* algorithms were sequential, and were efficient in spite of not knowing the future. We generalized the notion to the construction of efficient algorithms in spite of not knowing what already happened, but in remote places C[22]. A related issue I raised is the efficient estimation of the number of events that occurred far away from each other J[3,4]. (If such a report is delayed until another event occurs, it may be possible to save in the reporting, by bundling the two reports together; on the other hand, if no additional event occurs, waiting may cause a deadlock). In J[27], we introduced a time competitiveness measure for handling the delay in networks. Other papers that were among the first to suggest competitive algorithms in the context of networking are J[14,22,9]. A related paper is also C[8]. Competitive network algorithms have received meanwhile a rather wide treatment. New approaches (using methods from theoretical CS) to classical networking problems were presented in J[8,10,20,24], C[12,20].

2. Fault Tolerance and Locality in Distributed Algorithms

Dijkstra suggested the fault tolerance notion of *self stabilization* in the context of the classical problem of *token passing*. A token is supposed to circulate a ring network endlessly. A network (global) state where exactly one node has a token is correct, while a state where no node has a token is faulty, as is a state where there are multiple tokens circulating the ring. The main difficulty in detecting a fault is that a node can "see" only nearby nodes, and does not know whether a remote node contains a token. Self stabilizing systems recover from any incorrect state. In an $n-$node ring, Dijkstra's algorithm stabilized in $O(n^2)$ time, involving all the nodes, even if only one of them is faulty. As an example of locality, in C[38], the algorithm stabilizes in $O(f)$ time, where f is the unknown number of faults. Hence, the recovery is especially fast in the very common case of a small number of faults.

Previous studies each dealt with a specific task (except for one independent study). Moreover, previous solutions were global in nature, involving all the nodes in the detection and/or the recovery. In J[15], C[14], we suggested a self stabilizing distributed tree primitive that we explained could be used for performing a *reset*: a general method to transform algorithms to be self stabilizing. Such a primitive was suggested also by others independently, as well as by many others later. Still, the primitives suggested in C[29,25] are the most time efficient. Detailed reset protocols that use this primitive indeed were proposed later by various researchers.

We also suggested in J[15], C[14] an approach by which a fault can be detected by some node after consulting only its neighbors. This allows for fast detection, as opposed to global solutions in which there is a delay until reports are received from far away nodes. Such detection methods too were later generalized and studied by various researchers. In J[35], C[48], we gave lower and upper bounds for the amount of information that a node needs to convey to its immediate neighbors in the network for such a detection in various tasks. I have also been studying the localization of fault recovery, not just the detection. See J[21,23,25,38], C[44,46,56], in addition to C[38] mentioned above.

In J[18] we suggested a *local* distributed algorithm to decompose the network into a small number of clusters so that the radius of a cluster is at most a given k. The time complexity of the algorithm was only about k and the output at a node is affected only by nodes at distance about k from it. This enables the use of such an algorithm even if the network grows very large. Such a clustering is a useful primitive in networking, for routing, resource assignment, scheduling, etc. (Other clustering algorithms appear in J[25].) In J[17], we posed the question of the distributed computing of a minimum spanning tree in a time that is shorter than the depth of the tree. This requires many local communications, rather than long range communications on the tree, that may take a long time. On the other hand, we needed then to prevent a congestion of a large number of local communications from slowing down the algorithm.

3. Trees in Networks

From the practical point of view, trees play major roles in networking. From the theoretical point of view, they seem to embody most of the inherent problematic of distributed computing on networks. For example, to construct a tree, nodes have to break their symmetry to decide who will be the root. To detect that an edge closes a cycle (and should be omitted from the tree), far apart nodes need to compare their information. To be used for a broadcast, a tree must be made to span the network, even if the network changes. etc. For parts of my research on distributed trees, I received the Henry Taub Award for Excellence in Research.

In C[12], we present an algorithm to maintain a tree in a dynamically changing network. Previous tree algorithms either assumed that the root does not fail, or reconstructed the whole tree from scratch when the network changed. Here, the cost per change is optimal, and the routes over the tree are preserved whenever possible. Even if the root fails, the algorithm maintains the

property that the tree has a root. This solves the leader maintenance problem, allowing the network to avoid conflicting decisions (e.g., conflicting assignments of resources) since the root can arbitrate. Another application of C[12] is the most message efficient *reset* protocol (a protocol that translates other protocols so that they can work on dynamic networks).

In a broadcast, nodes that already heard the message forward it to their neighbors. Using a tree for forwarding prevents the message from looping in cycles in the network. This is especially useful in fast networks, since the actions needed for the fast switch to detect and avoids such looping, would have slowed the switch operation. The multicast algorithm for fast networks of J[31] used in IBM NBBS predated the multicast algorithms of the Internet, and the topology broadcast algorithm for fast networks of J[11] saves significantly on traffic, compared to the algorithm of the Internet.

In J[15], C[25,29], we presented tree maintenance algorithms that are also self stabilizing. In J[3,4,6], we suggested tree construction algorithms for networks in which some of the nodes may have crashed undetectably before the execution. (Otherwise, this is impossible). The algorithm of J[5] showed an efficient way to construct rooted trees in a unified way even though it could function in different kinds of networks. Previously, multiple algorithms were presented, each for a different kind of networks. J[5] also raised the question of how to traverse (and construct a tree in) directed networks and solved it for the Eulerian case. This was generalized in C[8] to general directed networks, where the graph *deficiency* (distance from being Eulerian) was identified as a parameter that dictates the traversal complexity. This question was later studied by other researchers in the non-distributed context of competitive learning.

In J[17,18], we improved the time complexity for the classical problem of constructing a minimum spanning tree distributively. In [13,30,32], we suggested tree construction algorithms for a wide family of *overlay networks* (where the algorithm in a node v can create a "cheap" virtual link to another node u whenever v knows the route to u). Such a model represents well various kinds of current networks, e.g. Peer to Peer systems. Problems of searching and delivering content using trees for overlay networks were addressed in J[29,34].

In C[2] mentioned above, we suggested a distributed combination between Depth First Search and Breadth First Search traversals to construct a tree for radio networks. In J[8], C[20], we demonstrated the use of trees (dynamic trees, in C[20]) for preventing deadlock in routing. In J[54], we improved previous algorithms (including mine) to report events to the root of a dynamic tree, and to issue permits for distributed requests.

In J[35,39], we introduced a model for the efficient verification of distributed properties. We demonstrated the model by providing tight lower and upper bound for verifying a minimum spanning tree. This can be viewed as a distributed version of a classical problem in data structures and graph theory.

References

- [Turner] Turner, J.S., "New directions in communications (or which way to the information age?)", IEEE Communication Magazine, Vol 24, No. 10, Oct 1986. Later appearing in a collection of landmark articles, May 2002.
- [Tanenbaum] Andrew Tanenbaum, Computer Networks, Fourth Edition, 2002, Prentice Hall.