RESEARCH DIRECTIONS IN DISTRIBUTED COMPUTING AND COMMUNICATIONS

Yechiam Yemini

CUCS-87-83

RESEARCH DIRECTIONS IN DISTRIBUTED COMPUTING & COMMUNICATIONS*

Yechiam Yemini

Computer Science Dept. Columbia Univ. NY, NY 10027

ABSTRACT

This paper presents a brief discussion of some research directions and open problems in the areas of distributed computing and communications Three areas of theoretical problems and two technical problem domains are considered The paper aims at providing a brief tutorial/survey of the respective problem domains The presentation is naturally biased to reflect the interests of the author

1. Theoretical Foundations

1.1 Coordination

For the purpose of this section, a *network* is a labeled graph A *network problem* is one involving a computation of a function of the network. Examples of network problems include computing a minimal spanning tree, shortest path tree and a maximal matching [Papadimitriou, A and Steiglitz, K. 82] Problems of *network coordination* involve the **cooperative solution of a network problem by processors placed at network nodes and to which only the data local to the respective nodes (i.e., the local topology and labels) is available.** Examples of coordination problems include the computations of optimum routing [Kleinrock, L. 76, Schwartz, M. 77], computation of a minimal spanning tree [Dalal, Y 77, Spira, P.M. 77, Chang, E 80, Gallager, R.G et Al 79, Dechter, R et Al 81], computation of maximal flow [Segall, A 83] and range measurements [Yemini, Y 83a]. The complexity of a coordination algorithm is typically measured in terms of the (worst case or average) number of messages required to accomplish the computation of the respective problem.

Coordination problems typically arise in the design of protocols for the distributed cooperative solution of certain network management problems. As computer networks evolve from communication mechanisms into distributed computing systems, the significance of network coordination problems may be expected to increase.

Theoretically speaking, it is always possible to provide a centralized solution to a coordination problem Simply collect all data available to nodal processors to a single central processor and solve the problem there disseminating the solution, if necessary, to the required nodes. The delivery of information to the central node requires at most $o(r)$ message transmissions, where r is the radius of the network. If n is the number of nodes in the network there are $o(nr)$ transmissions involved. The radius of a random network with n nodes is on the average $o(lgn)$, thus the centralized solution requires $o(nlgn)$ message transmissions for a random network with n nodes.

The centralized solution is basically undesirable Firstly, some coordination problems possess a distributed solution involving fewer communications Secondly, the centralized solution requires that a single node be charged with the computational task; this is both an unreliable solution and one where the computing power available in other nodes is not utilized to reduce the computing time Thirdly, the communication complexity measure used above does not capture the bottleneck effect created by $o(n)$ message transmissions at the central node

Very little experience and understanding of network coordination problems exist Is the description of network problems, as specified above, plausible? Obviously not Real networks do not involve arbitrary graphs Algorithms that perform well in the worst case of a general graph may be much worse than other algorithms when the graph is a drop-tree network How do we design coordination algorithms to perform well (optimally) for graphs representing real networks? Is the complexity measure above (i.e., counting message transmissions) adequate? Obviously not Counting messages does not capture communication bottlenecks nor does it capture the delays involved in sequential transmissions (i.e if n nodes transmit simultaneously

* This research was supported in part by the Defense Advances Research Projects Agency, Proj. N00039-82-C-0427, and The National Science Foundation, Proj. MCS-81-10319

or if they transmit sequentially, the times involved are substantially different) Finally, one of the important rationales for a decentralized solution is that typical network problems involve time-varying labels and/or graphs. The above definition of coordination problems does not capture the dynamics of these changes.

1 1 1 Example: Data Collection

Suppose each node in a given network possesses some data. Any given node might wish to collect all the data in the network in order to execute some local algorithm. E. Chang [Chang, E. 80] suggested the bug/echo coordination algorithm to solve the problem. The given node serves as the originator of a "bug" message that it broadcasts to all its neighboring nodes Nodes continue broadcasting the bug to any of their neighbors that were not visited by the bug. Each node identifies its "father" as the node from which it received the bug, and its "sons" as all the nodes that accepted the bug from it. The bug thus spans a tree rooted at the origin node through the network graph. A node is a leaf in the tree if it has no sons (i e., all its neighbors have been visited by the bug) A leaf node turns the bug into an "echo" message and sends it back to its father Each node waits for echoes from all its sons and then sends an echo to its father. Echoes might carry the data back from the nodes, up the the tree, to its root Clearly, the number of transmissions involved in this algorithm is O(n)[1]

Now suppose that a set of m nodes tries (asynchronously) to collect the same data from the network. If each of the m nodes executes its own bug/echo algorithm, then m data-collection trees will be spanned in the graph, each rooted at a different originator The number of transmissions involved is O(mn) Is it possible to improve this? There are a few positive answers possible. Let us briefly outline one coordination algorithm that achieves coordinated multiple origins data collection in O(n) transmissions. Each of the origins might start a bug/echo algorithm. The different bugs create a set of trees through the network. However, this time a node becomes a leaf if all its neighbors have been visited by **any** bug. In each tree, the originating node collects all the data using the echoes. At the end of the first phase of the algorithm, the network nodes are partitioned into disjoint trees whose roots possess the data for all the nodes in their respective trees. Two neighboring leaves of two trees are said to form a bridge In the second phase, each of the trees formed in the first phase becomes a "node" in a graph whose edges are bridges The echo/bug algorithm is executed in this new graph to collect all the data from each of the origins and is then used to disseminate all of this information to each origin.

We can now add the following questions to the open problems stated above How do we adequately describe coordination algorithms? How do we prove the correctness of the algorithms? How do we provide a more realistic and less crude performance analysis? How do we deal with temporal dynamics of the network/data (e g., what if, as the bug travels through the network described in the above example, the topology changes so that the echoes cannot return along the same path)?

1 2 Artificial Society

How should one design network algorithms? Most practical algorithms running in networks fall into the category of resource sharing algorithms. Accordingly, let us restrict our interests to the problem of designing effective network resource sharing algorithms Problems of this nature have been addressed in numerous works over the past two decades [Kleinrock, L. 75, Kleinrock, L. 76, Schwartz, M. 77, Tanenbaum, A.S. 81] Typically, a global optimization paradigm is assumed in addressing a network resource sharing problem, namely, a global performance objective is assumed and an algorithm (protocol) to (approximately) optimize the goal is proposed. The problem with the global approach is that it leads to results that are not easily decentralizable (sometimes provably)

Perhaps the best examples of decentralized resource sharing mechanisms with which we are familiar, are those used by society Interestingly enough, despite the rich body of formal methods and heuristic descriptions of social organizations for effective resource sharing, very little has been done to apply this knowledge to optimum network algorithms.

The social-approach to the design of network algorithms may be loosely described as follows: Every computing agent in the network is endowed with a personal selfish utility function representing its objectives Rather than optimizing a single global objective the agents aim at obtaining a Pareto-optimal [Shubik, M 83] operating point (i e , an operating point at which it is impossible for any agent to improve its own utility without affecting any others

[1] Note that the discrepancy between this and the O(nlgn) centralized solution complexity originates here since we allow nodes to combine messages

Such optimization is carried out in a completely decentralized manner. Global priorities may be achieved by centralized prioritization of agents. In short, one views a network as an artificial society of computing agents, each pursuing its own goal while interacting with other agents. This approach was applied to network and distributed problem solving in [Yemini, Y. and Kleinrock, L. 79, Davis, R 79, Yemini, Y 81, Brooks, R. 83, Kurose, J. 84, Kurose, J. et al. 83]

The advantages offered by the social approach to distributed problem solving can be summarized as follows: First, it is possible to utilize the rich body of methods constructed by social scientists in the solution of distributed problems. Second, the algorithms generated by this approach are decentralized to start with (unlike the global approach that typically yields centralized solutions requiring ad-hoc decentralization). Third, it allows unified systematic attack on classes of problems. Finally, the solution is robust to ill-behaved individual agents (i.e., a failure in the operation of any single agent does not cause a domino-effect by the decoupling of objectives among agents)

Recent work by J Kurose [Kurose, J. 84, Kurose, J. et al. 83] demonstrates how network resource sharing problems may be addressed by building an exchange economy in the network. Resources are initially allocated to the agents and prices set to certain initial values. Resources are then traded by the agents and prices are reset according to supply and demand. This iterative process leads to an equilibrium allocation of the resource that is Pareto-optimum. Such algorithms have been applied to a few resource sharing problems (e.g., channel access schemes) leading, in all cases, to optimum solutions.

1.2.1 Example Social Aloha

The Slotted-Aloha access scheme problem has been extensively addressed in the literature [Abramson, N 70, Kleinrock, L. 76] A time-slotted broadcast channel is shared among multiple bursty users. User i, having a packet, decides whether or not to transmit by tossing a coin with probability of transmission p(i). If two or more users transmit at a given slot, the transmissions collide. The problem is for users to set the value of p(i) to achieve optimum channel utilization.

Consider a multihop packet broadcast network using Slotted Aloha. If the resources are defined as channel usage at the nodes, then the transmission probabilities p(i) define an allocation of the resources to nodes. The utility of each node is definable in terms of the probability of its successful transmission. Nodes can prices their transmission probabilities and trade channel usage with neighbors adjusting prices according to supply and demand. Such a decentralized tatonment process can be shown [Kurose, J 84, Kurose, J et al. 83] to lead to Pareto-optimum equilibrium allocation of the channel. Simulation studies show that this process converges to classicaly optimal choice of transmission probabilities. Furthermore, the intial distribution of transmission probabilities provides for a continuum of priorities among nodes.

1.3 Analysis

The problem of developing adequate tools for the understanding and design of large-scale and complex computing systems is undoubtedly one of the prime challenges of computer science. As large-scale computing systems are formed and the interaction between their components follow patterns of increasing complexity, the problems of analysis of the system behavior become exceedingly difficult. Examples of domains where such problems arise include large-scale networks, multiprocessor switching mechanisms and distributed databases.

The problem of analysis is a problem of **description**. Basically, one is interested in a global, macroscopic description of the system behavior (e.g., average thruput and delay) Moreover, one is interested in a steady-state description of the behavior. There usually is ample information available on the microscopic rules governing the dynamic evolution of interactions among system components. **How do we pass from a microscopic dynamic description to a macroscopic description?** The dual problem is the problem of synthesis. Given certain global macroscopic performance objectives, how do we design the set of microscopic rules to optimize global performance objectives?

The classical Markovian approach to analysis requires that one describe the evolution of the system as a Markov process by using a detailed state description. The steady-state distribution of the Markov process is computed (if possible) and, from that, the global macroscopic behavior. While this approach is powerful in analyzing small systems (one whose state description is sufficiently simple) or systems decomposable to small systems, it cannot adequately address the problems of complex systems

Again, the resemblance to problems faced by other fields, such as physics, is striking. Yet despite the rich body of results developed by physicists to address the problems of coarse-graining descriptions, little use has been made of the analogies. It took physicists a few hundred years to find out that an abstract and unobservable quantity of "energy" could provide a more adequate description of the laws of mechanics than the use of an observable such as "force" It also took several centuries for physicists to build bridges between microscopic descriptions (many-body problems) and macroscopic (thermodynamic) descriptions. In the process, powerful methods were developed that may be found useful in addressing the problems of distributed systems analysis.

Can it be that a better road to analysis is to replace classical observables (arrival & service processes) with other quantities that may not be directly observable? Can we use approaches to the analysis of large-scale distributed systems similar to those used by physicists? Initial positive results in this direction have been reported in [Benes, V E 65, Ferdinand, A.E. 70, Yemini, Y 83b] and the closely related [Boorstyn, R.R., and Kershenbaum, A. 80].

1.3.1 Example: A Distributed Data Base System

Consider a distributed DBMS consisting of n lockable units (e g., records). A query that accesses the DBMS requires the exclusive use of a subset of units that it locks using one protocol or another If a query attempts to access a locked unit, it is queued and is serviced later on when the record becomes available. Assuming that queries are generated according to certain statistics, the problem of analysis is that of computing such macroscopic quantities as the average throughput of the system (e g., how many queries are processed per unit time) and the average blocking rate. A formal model can be presented of the system [Yemini, Y 83b] that is formally analogous to some models of statistical mechanics as follows: The inverse logarithm of the query generation rate defines the "temperature" of the system. An active query is analogous to a center of a "hard sphere" consisting of all queries that are blocked by it. This defines the "volume" of the system as that space occupied by the hard spheres. The "energy" of the system reflects the average number of queries concurrently processed. The "pressure" of the system reflects the average rate of blocked queries.

The formal analogy to physical models permits not only the derivation of macroscopic descriptions of the behavior of complex computing systems, but potentially a global approach to their design. For example, in the case of a distributed DBMS it allows for the identification of bottlenecks (records experiencing high-pressure) and permits their removal through redesign. Finally, some physical systems experience discontinuous phase transitions when the temprature changes through some critical value For example ferromagnetic metals become perfect magnets when they are cooled beyond their Curie temperature Such critical transitions typically involve formation of long-range order among the constituents of the system The existence of critical transitions in large scale computing/communications systems is of particular significance. In [Yemini, Y 83b], certain packet broadcast systems that exhibit critical phase transitions are described.

2. Technical Foundations

2.1 Protocol Development

According to recent estimates, by 1990 between a quarter to a third of all US households will have a computer Ten years from now, the computer will be a common household utility like the refrigerator and the television set. A major use of the home computer, as well as office and factory computers, will be access to information resources over computer networks. Universal accessibility of such resources is undoubtedly a key element in the unfolding of the information revolution. The development of protocols to facilitate universal accessibility to information resources is a prime technical problem that we face. Furthermore, as large-scale distributed computing systems assume an increasing role in supporting special computing functions, it is reasonable to expect that the variety and complexity of communication protocols will be increased What are some of the key open questions of the protocol design problem?

The problems of formal protocols descriptions (specifications) methods, implementation and validation/analysis [Sunshine, C 82, Rudin, H. and West, C H. 83] are still far from a conclusive solution Over the past decade, numerous formal tools have been proposed to support the above functions yet very little has been achieved in terms of full automation of distributed message-based systems design process. Like the VLSI chip designer, the distributed-system protocol designer faces the problem of managing design complexity The problem could be substantially eased through a software environment for protocol design A number of research efforts in this direction are currently being pursued [Rudin, H. and West, C.H. 83].

Of particular significance is the problem of unifying different tools for performance and functional verification of distributed systems. Distributed systems tend to exhibit "timing bugs" that generally are not captured by classical verification techniques. Timing bugs in communication protocols can account for some major problems, such as PABX's failing to establish connections and the space-shuttle failing to initialize its computing engines. In a recent work [Yemini, Y. and Kurose, J. 82], it has been shown that the alternating-bit protocol, perhaps the simplest and most widely verified protocol, does in fact possess a timing bug with potentially disastrous functional behavior. This bug cannot be adequately captured by current formal verification methods. This underlines the significance of building a stronger, more comprehensive foundation for protocol specifications and analysis techniques, such as how to formally specify protocols behavior accounting for both their functional and performance behavior, or how to unify verification/performance-analysis to capture the full behavior of the protocol. How can these formal methods be supported by a unified automated protocol-CAD workstation? These are some of the key technical challenges of this field.

2.2 Integration of Media

The problem of providing communication technology to support integration of communication media (e.g., audio, data, graphics, facsimile, image) and communication and processing is widely regarded to be one of the major problems of computer communication technology in the coming decade. Integration is required to facilitate the automation of the office and to allow individuals at home to benefit from a large variety of information and communication services. Among the key questions to be answered by the technology are: What novel and useful services may be offered to users by integrated systems and how should communication architectures be designed to support optimum integrated communications?

In the area of user-oriented integrated services, early experimentation is aiming at the creation of integrated work-stations supporting text/graphics/voice processing and communications. Early systems (e.g., Forsdick, H.C., 83, Swinehart, D.C. et Al, 83) aim at multi-media, multi-window documents. The electronic medium does not have the limitations of the paper medium, and may be used in novel ways to present information. It is possible to create documents that present multiple streams of information through multiple windows on a powerful bit-mapped display. A stream may present text, graphics or voice. Streams may be used for coordinated presentation of related information. For example, a voice stream might be used to annotate a slide presentation through stream in a graphics window. By keeping a distributed document over a network, it is possible to exchange streams of information through the windows using any of the media. Thus, multi-media teleconferencing may be viewed as a distributed editing of a multi-media multi-window document. How should multi-media distributed documents be supported over a network? How should distributed editing be coordinated? How should multi-media documents be projected through limited capabilities terminals (e.g., ordinary CRT, telephone)? These are just some of the questions for which experimental research will need to provide the answers.

How should integrated multi-media communications be locally distributed? Local distribution is, at present, the scene of a clash between two philosophies & technologies, the PABX and LANs. The underlying assumption of classical telephony is that the devices served by the network possess very limited processing capabilities and thus all communication processing should be centralized in the network. The underlying assumption of LAN-technologies is that the devices served by the network possess very powerful processing resources and therefore could easily perform all communication functions, thus rendering the network a passive transmission medium. Distributing communication processing to the network periphery allows use of demand adaptive resource sharing as in packet switched communications. On the other hand, the complexity of coordinating distributed access to a resource limits the geographical range of the network and the ability of the network to support the regularity of traffic streams, centralization addresses these dimensions more fully. The centralized approach seems to serve voice communications more adequately while decentralized approach is more suitable to handle the requirements of bursty data communications. Is there an architecture between that of PABX and LANs that would support equally well both traffic types? A direction to search [Yemini, Y. 83c, Suda, T. and Yemini, Y. 83] is for an architecture that combines communication processing at the network periphery with minimal distributed processing (switching capabilities) inside the network.

References

[Abramson, N 70]
Abramson, N.
The ALOHA System - Another Alternative for Computer Communications.
In *AFIPS Conference Proceedings, FJCC*, pages 281-285. AFIPS, 1970.

[Benes, V.E. 65] Benes, V.E.
Mathematical Theory of Connecting Networks and Telephone Traffic.
Academic Press, 1965.

[Boorstyn, R.R., and Kershenbaum, A 80]
Boorstyn, R.R., and Kershenbaum, A.
Throughput Analysis of Multihop Packet Radio.
In *Proceedings ICC*. ICC, June, 1980.

[Brooks, R. 83] Brooks, R.
Experiments In Distributed Problem Solving With Iterative Refinement.
PhD thesis, Computer Science Department, University of Massachusetts at Amherst, February, 1983.

[Chang, E. 80] Chang, E.
Distributed Network Algorithms.
PhD thesis, Computer Science Department, Toronto University, 1980.

[Dalal, Y 77] Dalal, Y.
Algorithms For Broadcast Networks.
PhD thesis, Computer Science Department, Stanford University, 1977.

[Davis, R. 79] Davis, R.
The Contract-Net Approach to Distributed Problem Solving.
PhD thesis, Computer Science Department, Stanford University, 1979.

[Dechter, R. et Al. 81]
Dechter, R., Afek J., and Levy, H.
Distributed Algorithms for Spanning Tree Construction.
Technical Report, UCLA, 1981.
Preprint.

[Ferdinand, A.E. 70]
Ferdinand, A.E.
A Statistical Mechanics Approach to Systems Analysis.
IBM J. Research and Development, 1970.

[Gallager, R.G. et Al. 79]
Gallager, R.G, Humblet, P.A., and Spira, P.M.
A Distributed Algorithm for Minimum Weight Spanning Trees.
Technical Report, MIT, 1979.

[Kleinrock, L. 75]
Kleinrock, L.
Queueing Systems Volume I: Theory.
J. Wiley and Sons, N.Y., N.Y., 1975.

[Kleinrock, L. 76]
Kleinrock, L.
Queueing Systems Volume II: Computer Applications.
J. Wiley and Sons, N.Y., N.Y., 1976.

[Kurose, J 84] Kurose, J.
Real-Time Communications in Computer Networks.
PhD thesis, Computer Science Department, Columbia University, 1984.

[Kurose, J. et al. 83]
Kurose, J., Schwartz, M., Yemini, Y.
A Microeconomic Approach to Distributed Resource Sharing in Computer Networks.
Technical Report, Computer Science Department, Columbia University, December, 1983.

[Papadimitriou, A. and Steiglitz, K. 82]
Papadimitriou, A. and Steiglitz, K.
Combinatorial Optimization: Algorithms and Complexity.
Prentice Hall, 1982.

[Rudin, H. and West, C.H. 83]
Rudin, H. and West, C.H. (ed.)
Protocol Specification, Testing, and Verification.
North Holland, 1983.
Third International Workshop, IFIPS WG 6.1.

[Schwartz, M. 77]
Schwartz, M.
Computer Communication Network Design and Analysis.
Prentice Hall, 1977

[Segall, A 83] Segall, A.
Distributed Network Protocols
IEEE Transactions on Information Theory Vol. IT-29, no 1, January, 1983

[Shubik, M. 83] Shubik, M.
Game Theory in The Social Sciences.
MIT Press, 1983.

[Spira, P.M. 77] Spira, P.M.
Communication Complexity of Distributed Minimum Spanning Tree Algorithms.
In *Proceedings of the Second Berkeley Workshop on Distributed Data Management and Computer Networks* 1977

[Suda, T and Yemini, Y 83]
Suda, T and Yemini, Y
Protocol Architecture of a Tree Network With Collision Avoidance Switches
Technical Report, Computer Science Department, Columbia University, 1983

[Sunshine, C 82]
Sunshine, C (ed.)
Protocol Specification, Testing, and Verification.
North Holland, 1982.
Second International Workshop, IFIPS WG 6 1.

[Tanenbaum, A.S. 81]
Tanenbaum, A.S
Computer Networks.
Prentice Hall, Englewood Cliffs, N J., 1981.

[Yemini, Y 81] Yemini, Y
Selfish Distributed Optimization in Computer Communication Networks.
In *20th IEEE Conference on Decision and Control* IEEE, 1981

[Yemini, Y 83a] Yemini, Y
Amalgamation of Distributed Information
In *Hawaii conference on System Science* 1983

[Yemini, Y 83b] Yemini, Y
A Statistical Mechanics of Distributed Resource Sharing Mechanisms
In *Proceedings, INFOCOM* IEEE, 1983

[Yemini, Y 83c] Yemini, Y
Tinkernet or, Is There Life Between LANs and PBXs?
In *Proceedings, ICC* IEEE 1983

[Yemini, Y and Kleinrock, L 79]
Yemini, Y and Kleinrock, L
On a General Rule for Access Control, or Silence is Golden
In J. Lagrange (editor), *Flow Control in Computer Networks.* North-Holland, 1979

[Yemini, Y and Kurose, J 82]
Yemini, Y and Kurose, J
Can Current Protocol Verification Techniques Guarantee Correctness?
Journal of Computer Networks .
December, 1982.