*Invited paper*

# Nomadicity: Anytime, anywhere in a disconnected world*

Leonard Kleinrock

*Computer Science Department, UCLA, Los Angeles, CA 90095-1596, USA*

**Abstract.** Nomadic computing and communications is upon us. We are all nomads, but we lack the systems support to assist us in our various forms of mobility. In this paper, we discuss the vision of nomadicity, its technical challenges, and approaches to the resolution of these challenges. One of the key characteristics of this paradigm shift in the way we deal with the information is that we face dramatic and sudden changes in connectivity and latency. Our systems must be "nomadically-enabled" in that mechanisms must be developed that deal with such changes in a natural and transparent fashion. Currently, this is not the case in that our systems typically treat such changes as exceptions or failures; this is unacceptable. Moreover, the industry is producing "piece parts" that are populating our desktops, briefcases and belt-hooks, but that do not interoperate with each other, in general. We require innovative and system wide solutions to overcome these problems. Such are the issues we address in this paper.

## 1. The vision

The usual assumption that most of us make about our computing and communication environment is that we are "always" connected. The Client/Server paradigm makes such an assumption. The local area network with PC's connected to each other and to servers makes such an assumption. The computer on the Internet makes such an assumption. Most of us think this is the natural state of affairs. But this is fast becoming an antiquated view! For indeed, most of us are "nomads" when it comes to computing and communications. We live in a disconnected world much of the time as we travel between our office, home, airport, hotel, automobile, branch office, bedroom, etc. Thus we must take the disconnected state as a "usual" one, instead of an "exceptional" one. To be disconnected is not a failure mode, as is the current view. Rather, it is a common mode.

As nomads, we own computers and communication devices that we carry about with us in our travels. Moreover, even without carrying portable computers or communications, there are many of us who travel to numerous locations in our business and personal lives, and who require access to computers and communications when we arrive at our destinations. Indeed, a move from my desk to a conference table in my office constitutes a fundamentally nomadic move since the computing platforms and communications capability may be considerably different at the two locations (even though they are separated by no more than 5 feet). The variety of portable *computers* is impressive, ranging from laptop computers, to notebook computers, to personal digital assistants (or personal information managers), to smart credit card devices, to wrist watch computers, etc. In addition, the *communication* capability of these portable computers is advancing at a dramatic pace from high speed modems, to PCMCIA modems, to email receivers on a card, to spread-spectrum hand-held radios, to CDPD transceivers, to portable GPS receivers, to gigabit satellite access, etc.

The combination of portable computing with portable communications is changing the way we think about information processing [1]. We now recognize that access to computing and communications is necessary not only from one's "home base", but also while one is in transit and/or when one reaches one's destination. Indeed, anytime, anywhere access.

The conclusion is clear: we must architect our systems to be nomadic-aware from the start, and not as ad-hoc patches to systems that are rigid in their view of connectedness. The essence of a nomadic environment is to automatically adjust all aspects of the user's computing, communications, and storage functionality in a transparent and integrated fashion.

The transparency we talk about is with respect to the following:

- *Location* you are at.
- *Communication device* you are using (modem, Ethernet card, etc.).
- *Communication bandwidth* you have available.
- *Computing platform* you are using.
- Whether or not you are in *Motion*.

The notion of transparency here does not refer to the quality of service one sees, but rather to the perception of a computing environment that automatically *adjusts* to the processing, communications and access available at the moment. For example, the bandwidth for moving

* This work was supported by the Defense Advanced Research Projects Agency DARPA/ITO, under Contract DABT-63-C-0080 "Transparent Virtual Mobile Environment".

data between a user and a remote server could easily vary from a few bits per second (in a noisy wireless environment) to hundreds of megabits per second (in a hardwired ATM environment); or the computing platform available to the user could vary from a low-powered Personal Digital Assistant while in travel to a powerful supercomputer in a science laboratory. Moreover, the ability to accept partial or incomplete results is an option that must be made available due to the uncertainties of the informatics infrastructure.

These ideas form the essence of a major shift to "nomadicity" (nomadic computing and communications) that we choose to address in this paper. The focus is on the system support needed to provide a rich set of capabilities and services to the nomad as he moves from place to place in a transparent and convenient form.

## 2. The technical challenges[1]

Let us consider some of the obvious technical challenges we must face with regard to nomadicity. We begin by listing some of the key system parameters with which one must be concerned in designing nomadic systems support. These include: bandwidth; latency; reliability; error rate; delay; storage; processing power; interference; interoperability; user interface; cost, etc. Indeed, these are the *usual* concerns for any computer-communication environment, but what makes them of special interest for us is that the values of these parameters change dramatically as the nomad moves from location to location. In addition, some totally *new* and primary concerns arise for the nomad such as weight, size, battery life, processing power, mobility, interference, damage, loss and theft, of his portable devices.

One can easily identify the *physical parts* of a nomadic system as consisting of the following (among others):

People that move (or don't).
Things that move (or don't).
Things that communicate (or don't).
Things you connect to (or not).
Things that can process, store, etc.
Things that can sense.
Things that can actuate.

On the other hand, the *logical parts* of a nomadic system are more slippery to define. Among others, they consist of the following:

Context (what things surround and touch my current activity).
Individuated nexus (what is the set of currently working objects).
Shared objects (what things are shared with me and others).
Replicated objects (what things are copied in multiple locations).
Cached objects (what do I hold onto as I travel and use objects).

Many of our current systems problems are severe enough, but nomadicity exacerbates almost all of them, including the following:

Disconnectedness.
Variable connectivity: unpredictable and voluntary.
Variable bandwidth.
Variable latency.
Variable routes, virtual circuits, etc.
Variable requirements as the nomad moves.
Resource replication.
Awareness of the environment by the user (environment discovery).
Awareness of the user by the environment (user discovery).
Foreign languages encountered.
Adaptivity/compression to match bandwidth and platform capability.

In general, we are overwhelmed in this environment by the *management of distributed "stuff"*. Those entries at the top of this last list are capable of changing extremely quickly, making things even more problematical. Other legacy problems cause serious performance degradations as well; for example, application protocols that insist upon long chains of round trip exchanges of control messages in order to get initiated. Unduly large headers can lead to large efficiency losses in the slow, expensive communication environments often encountered by the nomad; the same can be said about latencies.

These technology challenges demand innovative and system-wide solutions. In the next section, we introduce some of the recent work that has been initiated in an attempt to address some of these issues.

## 3. Approaches to technology solutions

One can identify at least three things that need to be done if we are to tame the problem of understanding nomadicity. They are:

1. Develop a systems architecture and network protocols for nomadicity.
2. Develop a nomadicity reference model.

[1] Some of the ideas presented in this section were developed with two groups with which the author has collaborated in work on nomadic computing and communications. One of these is the Nomadic Working Team (NWT) of the Cross Industrial Working Team [2]; the author is the chairman of the NWT. The second group is a set of his colleagues at the UCLA Computer Science Department who are working on a DARPA supported effort known as TRAVLER, of which he is Principal Investigator.

3. Develop performance models of the nomadic environment.

We develop each of these in the rest of this section.

### 3.1. Develop a systems architecture and network protocols for nomadicity

The architecture should provide for interoperation between the wired and wireless infrastructures as well as handle the nomadicity concerns of unpredictable user behavior, unpredictable network, unpredictable computing, and graceful degradation.

The architecture should consist of the following components: integrated access to services; ad-hoc access to services[2]; bandwidth-processor/resolution adaptivity[3]; dynamic shift of functionality[4]; automatic sensing, searching, and/or tracking of users as they move around the network, sometimes appearing thousands of miles away from their last visit to the network[5]; cooperation among the key components of the system at all levels of the architecture which we can identify (for example, working our way up the stack, we might consider sensors, actuators, devices, network, operating system, services, apps, etc.); maximum independence between the network and the applications that are developed on the net.

An issue central to any discussion of a system architecture is *scaling*. A proper architecture should scale with respect to most of the key system design parameters. A short list is as follows:

heterogeneity,
addressing,
bandwidth,
distance,
number of users,
quality of service.

There is a growing body of nomadicity research and development. Some of the components that are being developed are listed below:

- An integrated software framework for a common virtual network layer [4].
- A flexible multilevel replication capability.
- File synchronization as disconnected systems reconnect at unpredictable times [5].
- Predictive caching (hoarding) of needed files and objects when one prepares to travel in disconnected or variable connectivity environments [6].
- Resource discovery of tools, data, content, devices, etc. [7].
- Automatic discovery of a user's profile when he enters a new environment (and the converse, namely, resource discovery).
- Adaptive database management as users, devices, files and other objects migrate around the network.

In many of these components, one finds that "adaptive agents" can be very useful. Adaptive agents are typically software agents that perform an intelligent service on behalf of the application or the network or some other object in the system. These adaptive agents are often at the middleware level and can serve the nomads, the applications, the network, the servers, the communication devices, the computing devices, etc.

For example, in Fig. 1(a), we make the classical assumption in a Client Server model that the network connecting the Client and the Server is a "Fat" network, i.e., one with considerable bandwidth. However, in the lower half of the figure, we see the nomadic assumption that the network is not always fat; sometimes it is a "thin" (low bandwidth) network and sometimes it is of zero bandwidth (namely the Client and Server are disconnected). Most of all, however, in the nomadic environment, the network capacity may change suddenly and drastically. Also in the figure, we see some adaptive agents. In this case, they might serve as intelligent programs that sense the bandwidth between the Client and the network; in fact, one can think of them as "impedance matchers". If the access bandwidth is sufficient, then large files can be shipped between the Client and the network in both directions. However, when the agent senses that the bandwidth is reduced, the agent might call for some form of compression on behalf of the

[2] By this we mean that access should be "easy". For example, the telephone is a wonderful device in that I can use it anywhere in the world as long as I have a coin plus an empty public telephone booth. I need not inform the telephone system who I am. I need not pay for the service ahead of time. I need not warn them ahead of time that I plan to use the phone. I need not setup an account with them, etc.

[3] That is, the system should adjust what and how much of a remote object it transmits to the user. If the access bandwidth and/or the user platform cannot support certain levels off complexity, then the system should match the resolution of what is sent to the user to the complexity he can accept.

[4] For example, in a Client/Server setup (which is really Client/Network/Server), it is usually assumed that the Network is powerful enough to support any needed transmissions between the Client and the Server. Hence, the Server often contains much of the intelligence and data, and whenever the Client asks for either, it is shipped over the (assumed powerful) network, to the Client. Now, however, we cannot assume that we always have a powerful network. As a result, we may need to put more functionality on the Client side when the network is "thin", and change this boundary of functionality as the network bandwidth between the Client and the Server varies.

[5] The question here is, which of those methods should one employ. One might choose to sense the presence of a user in an environment. Or one might search for the user only when traffic appears in the network destined for him. Yet another possibility is to keep constant track of the user as he moves around. Clearly, the optimum solution here depends upon the rate at which a user moves, and the rate at which traffic is generated for him. Many of these issues are addressed within the Mobile IP standards studies [3].

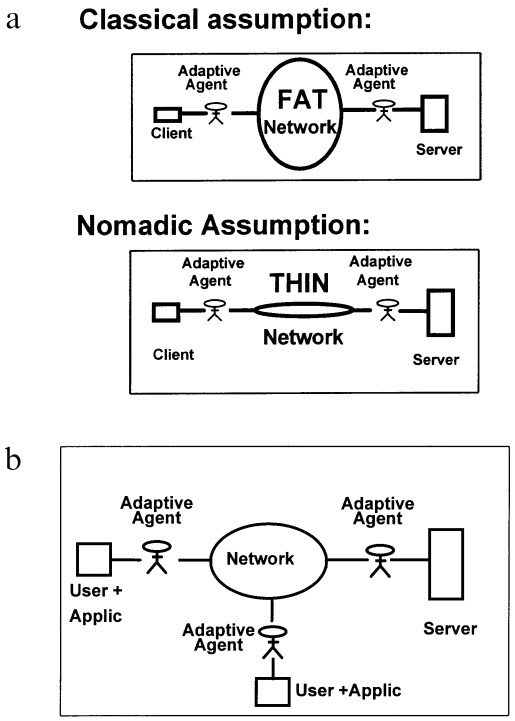


Fig. 1. (a) Adaptive agents in a Client/Server environment. (b) Adaptive agents in a Peer-to-Peer environment.

Client and the network. In Fig. 1(b), we see a Peer-to-Peer environment which also takes advantage of adaptive agents.

There has recently been an explosion of different types of agents. Sometimes they are called "proxies" or "aliases" or "surrogates" or "knowbots", etc. But they all do similar things. The field cries out for some standard way of talking about adaptive agents, their structure, their capabilities, functions, etc.

This discussion is just the beginning of a proper discussion of a systems architecture and the related protocols. Much more needs to be done.

### *3.2. Develop a nomadicity reference model*

In order to talk about nomadicity, it would be helpful to develop a reference model that identified a structure or a set of components for nomadic systems in some orderly fashion. A number of reference models or layered architectures already exist, and it seems plausible to consider adapting (or adopting) one of those for nomadic systems.

For example, in a recent NRC report [8], the following "hour-glass" architecture was suggested. Among its interesting features is the fact that it unbundles the network technology substrate at the bottom of the architecture from the upper level applications and middleware services. This allows these various components to develop in a free market competitive fashion while at the same time guaranteeing that options are not precluded

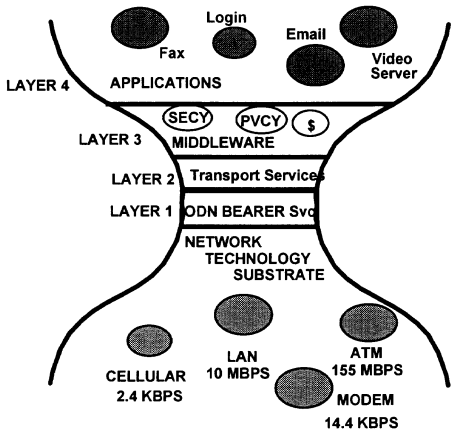


Fig. 2. The open data network 4-layer model.

by the construcation of vertically integrated systems (the so-called "stove-pipe" problem). In Fig. 2, we show the basic hour glass architecture which is known as the Open Data Network (ODN) model.

Perhaps a more appropriate layered reference model or architecture is one that recognizes some more detailed components of nomadic systems. For example one might suggest the architecture shown in Fig. 3.

Indeed, at UCLA, just such a model is currently being developed on the TRAVLER [9] project which is part of DARPA's Global Mobile Systems (Glomo) program. The UCLA Nomadic System Architecture is shown in Fig. 4. Here we see slightly different names for the various layers, and some examples of tools and applications that populate that model.

We have already identified some of the upper layer middleware and file systems components. At the bottom of the stack, we notice some details regarding how one can introduce "connectivity management" to deal with the automatic setup and connection to various communication infrastructures that the nomad might encounter in his travels [9].

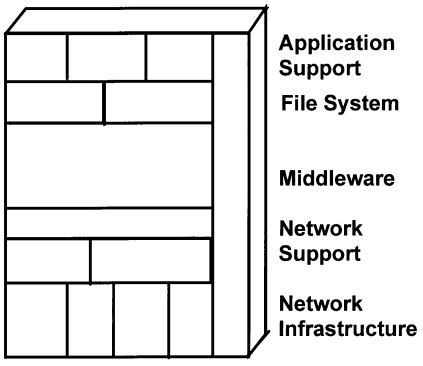


Fig. 3. A suggested nomadic system architecture.

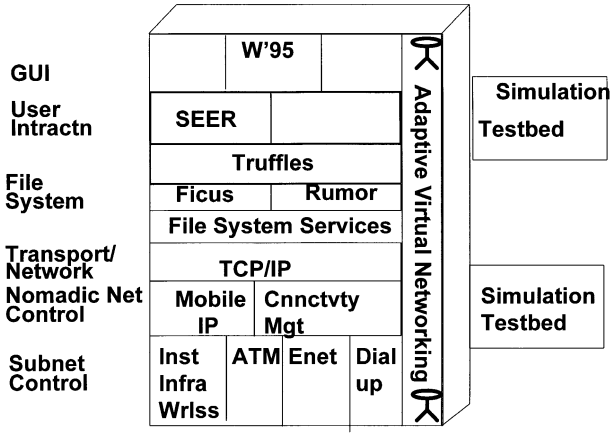


Fig. 4. The UCLA nomadic system architecture.

Clearly these are just some possibilities for a nomadic systems reference model. A number of groups are currently working on these and other models [10,11]

### 3.3. Develop performance models of the nomadic environment

The recognition of nomadicity is relatively new. As a result, there has been relatively little performance evaluation carried out for nomadic systems. Most of the work has focused on the lower levels of link behavior (e.g., wireless networking concerns [12–14]). The need to analyze and predict performance of these complex systems is very important.

Whenever one carries out performance evaluation, it is important to understand the process of modeling, analysis, and validation. The key point is that one is trying to predict the behavior of the real world. Any model that generates predictions must eventually have its results compared with those of the real world itself. This cycle is shown in Fig. 5.

One of the dangers in modeling is that the analyst may "fall in love" with his model and spend great effort in solving for the behavior of the model instead of focusing on predicting the behavior of the real world system.

Among the approaches to carrying out the modeling process, we identify the following choices:

*Mathematical analysis*:
The advantage of this method is its simplicity and ability to easily search a large parameter space. The problem often is the lack of realism one puts into the model and the possibility of analytically intractable models due to such things as non-stationarity, coupled queues, finite storage, etc. [15,16]

*Numerical evaluation*:
Here the problem of carrying out a difficult analysis is replaced with the possibility of exponential complexity in the numerical computation.

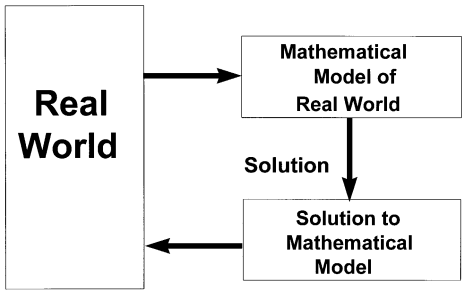


Fig. 5. The iterative nature of performance evaluation.

*Iterative solution*:
This is similar to the numerical evaluation case, but may cause additional problems regarding the rate of convergence of its solution.

*Simulation*:
This is a particularly effective tool which allows a great deal of realism to be built into the model. The use of parallel computers to help carry out the simulation is quite effective here [17]. The main problem with simulation is that it is often difficult to search a large solution space.

*Emulation*:
Here we get even more realism, but we suffer from possibly expensive and cumbersome implementations.

*Build the system and measure it*:
This provides the most realism, of course. However, it is almost impossible to search the parameter space and the delay in getting the results, as well as the possibly astronomical costs, represent some severe limitations of this method.

So what is one to do, since all have their pros and cons. The answer lies in the use of HYBRID models, wherein some portions of the system are evaluated with one modeling tool, while others may be evaluated in other ways. Often, it is possible to run all of these hybrid components simultaneously [17].

As an example of the use of the first method (mathematical analysis) let us consider a homework problem this author posed and solved 20 years ago, but never published until now. It is referred to as the problem of giant stepping in packet radio. Let us assume that we have a dense population of radios and we wish to transmit from point $A$ to point $B$ in some region. We assume that the distance between $A$ and $B$ is $D$ miles. We have a choice of trying to transmit at a very great power so as to reach the point $B$ in "one-hop", that is with no relays. The problem with this is that we will interfere with many other radios if we use such a large radius. An alternative is to transmit at a smaller power so that our transmission

reaches only out to a radius $R$. Now, we cause less interference on each hop, but we must transmit over many hops. Furthermore, we assume that all users behave in the same fashion. We note that if we use a small radius, then the interference at that radius due to other traffic is small, so that we will be able to succeed in our transmission with small delay. However, we will have to travel over many hops, and this increases the delay. The big question is, what is the optimum radius, $R$, at which we should be transmitting. Let us assume that $R \ll D$ for simplicity. Let us further denote $T(R)$ as the average delay a radio suffers in transmitting over one hop (assuming all radios behave in a similar fashion). Now it is easy to see that the total delay, $T = [T(R)](D/R)$. If we solve this for the optimum $R^*$ (i.e., that $R$ that minimizes $T$), then we find the simple result that at the optimal $R^*$, it must be that

$$\frac{dT(R)}{dR} = \frac{T}{R} .$$

The meaning of this result can be seen in Fig. 6. Here we see an arbitrary $T(R)$ function versus $R$. We also see a straight line emanating from the origin; in fact this line is the line with minimum slope that can just touch the $T(R)$ curve. The optimality condition says that the optimal $R$ occurs at this point of tangency. The result is very general: it works for any $T(R)$ function.

The point of the previous example is that very simple analytical models can sometimes yield very powerful and sweeping results. The danger is that the models may be too simple, and must be tested against the real world measurements, as mentioned above.

### 3.4. What do you need?

It is clear from the discussion above that the set of talents to deal with the major issues of nomadicity are quite broad. It is not likely that one person or even one group can properly address all the issues. As an example list of talents needed, let us look at the following areas, each of which forms a building block in the study of nomadicity:

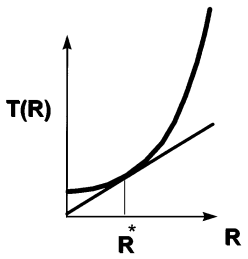


Fig. 6. The optimal transmission radius for packet radio.

Advance applications, such as multimedia systems.
Database systems.
File systems.
Operating systems.
Network systems.
Wireless communications.
Low power, low cost radios.
Micro ElectroMechanical Systems (MEMS) technology (sensors, actuators).

There is a clear need for strong cooperation across the disciplines described in this list if we are to make major contributions to nomadicity.

## 4. Conclusion

There are a number of compelling reasons why nomadicity is of interest. For example, nomadicity is clearly a *newly emerging technology* that users are already surrounded with. Indeed, this author judges it to be a *paradigm shift* in the way computing will be done in the future. Information technology trends *are moving in this direction.* Nomadic computing and communications is a *multidisciplinary* and *multi-institutional* effort. It has a huge *potential for improved capability* and convenience for the user. At the same time, it presents at least as huge a *problem in interoperability* at many levels. The contributions from any investigation of nomadicity will be mainly at the *middleware-level.* The products that are beginning to roll out have a *short term focus*; however, there is an enormous level of interest among vendors (from the computer manufacturers, the networking manufacturers, the carriers, etc.) for long range development and product planning, much of which is *now underway*. Whatever work is accomplished now will certainly be of *immediate practical use.*

Most of all, one must take an integrated viewpoint and bring to bear on the problem a variety of talents and capabilities (as listed previously). Taken in isolation, these problems are interesting; taken in their totality, these problems are challenging and fascinating.

As a final statement, it is fair to say that nomadicity is an emerging fact of life. The needs are real. The issues are fascinating. It makes all the problems harder. The payoffs can be huge. There exists a severe lack of an integrated approach. Confusion reigns. In a word, you can't (afford to) ignore the challenge of nomadicity!!!

## References

[1] M. Weiser, The computer for the 21st century, *Scientific American* (September 1991) 94–104.
[2] Nomadic Working Team of the Cross Industrial Working Team, Nomadicity: characteristics, issues, and applications (1995).

[3] IP Mobility Working Group, Routing support for IP mobile hosts, Internet Engineering Task Force, Internet Draft (1995).
[4] R. Bagrodia, W. Chu, L. Kleinrock and G. Popek, Vision, issues, and architecture for nomadic computing, *IEEE Personal Commun.* (December 1995) 14–27.
[5] P. Reiher, G. Popek, M. Gunter, J. Salomone and D. Ratner, Peer-to-peer reconciliation-based replication for mobile computers, submitted to *ECOOP '96 Second Workshop on Mobility and Replication.*
[6] G. Kuenning, G. Popek and P. Reiher, An analysis of trace data for predictive file caching in mobile computing, *Usenix Conference Proceedings* (June 1994) pp. 291–306.
[7] W. Chu et al., CoBase: A scalable and extensible cooperative information system, *J. of Intelligent Information Systems* 6(3) (1996).
[8] Nrenaissance Committee, L. Kleinrock, Chair, *Realizing the Information future: The Internet and Beyond* (National Academy Press, Washington, DC, 1994).
[9] *Transparent Virtual Mobile Environment: TRAVLER*, UCLA/DARPA Award # DABT63-94-C-0080, 10/94-9/97.
[10] M. Satyanarayanan, Mobile information access, *IEEE Personal Communications* 3(1) (1996).
[11] R.H. Katz, Adaptation and mobility in wireless information systems, *IEEE Personal Commun. Mag.* 1(1) (1994) 6–17.
[12] R. Jain, J. Short, L. Kleinrock and J. Villasenor, PC-notebook based mobile networking: Algorithms, architectures and implementations, *ICC 95*, Vol. 2 (June 1995).
[13] B. Woerner, T.S. Rappaport and J. Reed, *Wireless Personal Communications: Research Developments* (Kluwer, 1995).
[14] Ahmade, Krishna, Lamaire, Design issues in wireless LANs, *J. of High Speed Networks* 5(1) (1996).
[15] L. Kleinrock, *Queueing Systems*, Vol. I: *Theory* (Wiley-Interscience, New York, 1975).
[16] L. Kleinrock, *Queueing Systems*, Vol. II: *Computer Applications* (Wiley-Interscience, New York, 1976).
[17] R. Bagrodia and W-T. Liao, MAISIE: A language for the design of efficient discrete-event simulations, *IEEE Trans. Software Engineering* 20(4) (1994) 225–238. (Also, CSD Technical Report 920005.)

**Leonard Kleinrock** is a Professor of Computer Science at the University of California, Los Angeles, since 1963. He received his B.S. degree in electrical engineering from the City College of New York in 1957 and his M.S.E.E. and Ph.D.E.E. degrees from the Massachusetts Institute of Technology in 1959 and 1963, respectively. His research interests focus on performance evaluation of high speed networks and parallel and distributed systems. He has had over 200 papers published and is the author of six books. He is the principal investigator for the DARPA TRAVLER project as well as other DARPA contracts at UCLA. He is also founder and CEO of Technology Transfer Institute, a computer-communications seminar and consulting organization located in Santa Monica, CA, as well as founder and President of Nomadix, L.L.C., a Santa Monica firm that develops software and hardware products for nomadic computing and communications.

Dr. Kleinrock is a member of the National Academy of Engineering, is a Guggenheim Fellow, an IEEE Fellow, and was a founding member of the Computer Science and Telecommunications Board of the National Research Council. He has received numerous best paper and teaching awards, including the ICC 1978 Prize Winning Paper Award, the 1976 Lanchester Prize for outstanding work in operations research, and the Communications Society 1975 Leonard G. Abraham Prize Paper Award. In 1982 he received the Townsend Harris Medal. In 1982 he was co-winner of the L. M. Ericsson Prize, presented by His Majesty King Carl Gustaf of Sweden, for his outstanding contribution in packet switching technology. In July 1986, Dr. Kleinrock received the 12th Marconi International Fellowship Award, presented by His Royal Highness Prince Albert, brother of King Baudoin of Belgium, for his pioneering work in the field of computer networks. In the same year, he received the UCLA Outstanding Teacher Award. In 1990, he received the ACM SIGCOMM award recognizing his seminal role in developing methods for analyzing packet network technology and in 1996 the Harry M. Goode Award.
E-mail: lk@cs.ucla.edu