PARALLEL SIMULATION OF LARGE-SCALE WIRELESS SENSOR NETWORKS

***RAMAN CHADHA, @DR. SATYA DEV GARG,# SACHIN AHUJA**

**Research Scholar, Monad University*

@TCS Consultant, Noida

#A.P. VCE, Meerut

ABSTRACT

In the scenario of a usual catastrophe or a terrorist attack, a large number of self organizing, low-cost sensor devices can be deployed over the affected area. Each device equipped with its own power source, sensor, processing unit and low-power radio, can be imbued with the intelligence to seek out its neighbors and join in a wireless network spanning the geographic domain. The sensed quantities can then be forwarded to collections points, where the information is aggregated and presented to emergency response teams. We are developing a high-performance framework for the large-scale simulation of wireless ad-hoc networks (SWAN). Our framework is comprised of inter-operating sub-models for terrain, dispersion of hazardous substance, radio propagation, and the actual source code of ad-hoc networking protocols. In this paper, we describe the architecture of this framework and present experiments that confirm its usefulness in the study of routing algorithms.

1. INTRODUCTION

Current accelerated developments in signal processing and computer technology will soon allow large scale sensor networks to become viable and valuable in a wide variety of applications. Advances in microelectronics have lowered the cost of building blocks that can be put together to construct a new generation of sensors. These sensors can contain components for measurement, data acquisition and processing, and radio communication. Their small size and low per-unit cost will allow large collections of these sensors to be deployed over a geographic area, where they cooperate in gathering detailed information about variables of interest. The term “smart dust" has been coined to describe the smallest of these kinds of sensors built with micro electrical-mechanical systems (MEMS) [3,12,13,16]. The intelligence imbued in these small devices comes from their ability to self-organize. Sensors can interact with each other and construct, at deployment time, a wireless ad-hoc communication network which has the capacity to determine, on its own, how to route sensed data to randomly placed, and perhaps even mobile, points of collection. While much of the research in this area is currently focused on miniaturization, manufacturing and deployment of sensors, a sizeable portion of effort is being applied to software development for these tiny, embedded computers. The requirements of the code that executes in this kind of platform pose great restrictions on the programmers, who are faced with multiple limitations in terms of memory space, power consumption and scalability of communication algorithms. These restrictions make code development for sensors a very complex task. Testing and evaluating the software constructed for these platforms is further complicated for two main factors: first, experiment conditions are neither repeatable nor controllable, and second, the number of nodes in the network is potentially very large. For these reasons, experiments with real sensor networks must be made with a number of nodes that allows them to be manageable, which is typically ten or less [6]. The obvious engineering aid to use in these circumstances is computer simulation. Accurate, comprehensive simulation models for wireless networks, however, can be extremely computation intensive and most efforts in this area have focused on relatively small networks of a few tens of nodes [11,14]. The question that naturally arises is whether the performance of the designs evaluated by these modest simulations will scale up with the size of the networks. Considering that sensor networks aim at reaching tens of thousands of nodes, it is a problem of vast proportions to simulate realistic scenarios in which the network model inter-operates with intensive field simulation models. In this light, a high-performance computing approach becomes essential to the viability of these simulations. In this paper, we report the development of our Simulator for Wireless Ad-hoc Networks, or simply, SWAN. This project represents the coming together of Dartmouth's expertise in constructing a high-performance, scalable simulator, and BBN's experience with routing software for wireless ad-hoc networks. SWAN is more than the sum of its parts. Dartmouth's DaSSF [15] has first been released in the Fall of 1998 and has confirmed its promise time and again in the simulation of communication networks [7,8]. DaSSF's lean interface (see [9]) is, perhaps, as noteworthy as the performance it delivers because it allows for extreme ease of inter-operability. Simulation models for DaSSF can be constructed in a structured way, reused and extended. This feature was key in the execution of our project. BBN's portable WiroKit router for ad-hoc networks is another good example of inter-operability. By virtue of its design, WiroKit has few and well-defined points of contact with the environment on which it executes. It was created to be portable not only across deferent wireless platforms, but also easy to transport into simulation test beds. By enabling direct execution at source code level, WiroKit's reliability as a final product is increased, since what is verified and validated by simulation is ready to execute on target platforms without modification. The motivation for the development of SWAN came from the context of research carried out at the Institute for Security Technology Studies (ISTS) at Dartmouth College [1].

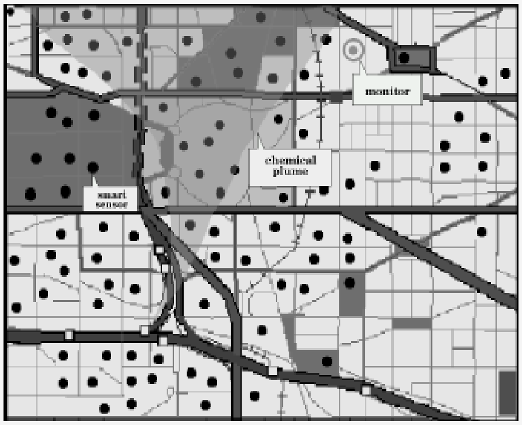

Fig. 1. Chemical Emergency Scenario

In this first stage of SWAN's development, we have built a simulation model to study the viability of wireless sensor networks as tools to aid the emergency response to chemical or biological threats. Next, we describe the concrete scenario that gave shape to our simulation framework. Suppose a deadly chemical agent is released in a metropolitan area and its plume is carried over the city propelled by wind currents as depicted in Figure 1. The first-responders are faced with the critically important problem of determining how the level of chemical contamination across the affected area evolves in time. This knowledge is crucial to operations such as the evacuation of the populace and the coordination of a response force. A large number of self-organizing sensors can be carried by helicopter, for instance, and scattered over the affect area in the first phase of emergency response. When the network comes alive, each Smart Sensor (or node) periodically uses its measuring device to assess the level of contamination and beams the relevant data to its nearest neighbours using a low-power, short range radio. The network then propagates this information to one or more Monitor nodes using paths determined by an autonomous routing protocol. The collected data can then be aggregated, processed and used to display the evolution of the chemical plume using real-time measurements. The remainder of this paper is structured as follows. In Section 2, we present the Scalable Simulation Framework (SSF), the structural glue that allowed us to describe and construct loosely coupled models that interoperate to achieve a full-edged system simulation. In the same section, we also brief present the Dartmouth Scalable Simulation Framework (DaSSF), a high-performance, multi-purpose and multi-platform simulator that complies with SSF specifications. Next, in Section 3, we present BBN's portable WiroKit router, which implements the sensor network routing protocol in our simulations. Section 4 presents the architecture of our simulator showing how its basic components were put together and how they inter-operate. In Section 5, we go on to brief discuss a novel RF channel model that allowed us a good measure of computational simplicity while maintaining good level of detail in our wireless simulations. Only the basic principles of this RF channel model are presented here, since a thorough exposition is outside the scope of this paper. In Section 6, we describe our simulation model and show the results of the first experiments performed with our framework. The empirical data obtained shows our results for networks of up to 10,000 nodes and indicates that our simulator supports models scaled up by another order of magnitude. Finally, Section 7 offers our concluding remarks and also outlines ongoing and future research directions in our project.

2. DASSF: DARTMOUTH SCALABLE SIMULATION FRAMEWORK

The S3 consortium has developed the Scalable Simulation Framework (SSF), a lean and simple interface for the construction of simulation models. SSF provides the modeler with the power to express the inter-relationships of model components in a systematic and structured fashion (we refer the reader to [2,9] for details). The simulation worldview imposed by the SSF API is process-oriented, isolating the modeler from the intricacies of managing event-lists and of explicitly dealing with the advancement of time. From a programming paradigm respective, SSF is object oriented and its API defines five base classes: entity, process, out Channel, in Channel, and event. An entity object is a container for state variables and a process, which describes how the state changes in response to interactions with other entities and/or to the passage of time. Each entity has a temporal \alignment", which in synchronization speak, situates it in a logical timeline. Entities that are co aligned are able to inspect each other's state variables. Temporal alignment serves to give the framework clues for concurrent scheduling in such a way as to maintain causal consistency, making sure that the future state of an entity doesn't affect the past of another. The exchange of data between entities is achieved through a channel, which denotes a unidirectional of events between two entities. In reality, channel is a concept that is implemented by the definition and mapping of two classes of objects: in Chan nel and out Channel. For communication to occur between two entities, the out Channel of one must be mapped to the in Channel of another. When an out Channel is constructed, it is associated with a minimum delay value and subsequent write operations may specify further delays individually. An SSF model is able to express how each sub-model communicates with others, clearly stipulating how data is exchanged, but more importantly, exposing the temporal coupling of its subcomponents. Since the model is described as a graph, where nodes are entities (possible containing processes) and edges are channels with well defined minimum delays, one can easily execute it using conservative parallel simulation techniques. This characteristic is key to the scalability of the simulation, since parallel execution increases the offer of memory space and computational power. The definition of this powerful, although simple API, has made a contribution to the simulation community in two different ways. First, it has lead to the creation of a family of compliant simulators for different programming languages and different computing platforms. Since SFF was designed so that the API could easily be translated to different programming languages, bindings have been produced for C++ and Java. With little or no modification, an SSF model written in one specific language can be ported to any SSF compliant simulator for that same language, independently of the nature of the computing platform (serial or parallel). Second, and perhaps most importantly, the structured approach imposed by the SSF API has allowed simulation programmers to make extensive used of design patterns, what stimulated the creation of databases of models. Experience has shown that this was an important factor to the development of communication network models of a scale previously unseen [7,8]. Furthermore, access to comprehensive databases of verified and validated model components reduces development time for new experiments, at the same time as increasing reliability of the finished product. Once a model component has gone through the extensive testing that warrants its placement in a database, it can be safely used as the cornerstone for a construction of larger proportions.

3. BBN'S WIROKIT ROUTER

WiroKit, developed by BBN Technologies, is a highly portable router for wireless ad-hoc networks. It is explicitly designed to run without modification in simulators or in real hardware platforms. That is, precisely the same interface definitions are used for the code that runs on a simulator and the code that runs inside a mobile radio unit. The design followed an object-oriented approach and, in fact, WiroKit is completely contained in a single object. This feature is essential for simulation environments, for it allows multiple copies of WiroKit to execute in a single address space. The platform requirements to run WiroKit are minimal. It implements the full software code base for routing protocols, forwarding engine, thread scheduling, and queue and memory management. There is virtually no need for an operating system. The only demands placed on the computing platform are that WiroKit be given a portion of memory at start-up time, access to a real-time clock and a minimum amount of the total CPU cycles for the execution of its main thread. The WiroKit router object receives packets from higher protocol layers, which it uses to build frames that are passed down to the radio modem. Conversely, it receives frames from the radio modem, which are stripped down into packets which are passed up to higher protocol layers. Within WiroKit, any routing protocol can be specified, as long as the same application programming interface (API) is maintained. This edibility allowed us to equip our router objects with algorithms specific to our application, that is, wireless sensor networks. Sensor networks are an area of active current research; see, for instance, [5,6,10,11,12,13,14]. In our research, we intend to tackle two key issues pertaining to this topic: routing protocols specific to this type of application, and also efficient datagram forwarding mechanisms. Since these are both complex in their own right, and tangential to this paper, we will only briefly discuss them here. As our research progresses, however, these will be main points of interest to us. Routing Protocols distribute information about \what is where" throughout the sensor network, and hence enable the sensor nodes to forward messages (datagram's) from one hop to another towards their intended destinations. Most routing protocols scale poorly with the number of network nodes. It is not

uncommon for a protocol's expense to grow with the square (or worse) of the number of network nodes. This expense can be in terms of over-the-air control traffic, node memory, or node CPU requirements. This obviously poses enormous problems for simulations, which, in turn, emulate the actions of N nodes. The resulting simulation, thus, often scales as N3 at best, and often as N4.
For our first experiments with sensor network routing, we have designed and implemented a simple tier routing protocol that discuses information about the distance to the data sinks (monitor points) in the network. Such protocols were employed in very early packet radio experiments, as well as in contemporary research, and provide a very simple and relatively effective means of disseminating information about the locations of the monitor points. Note that they do not, however, disseminate information about how to reach any of the other nodes in the network, that is, the sensor nodes, and thus cannot provide two-way connectivity between monitors and sensors.

4. THE ARCHITECTURE OF SWAN

SWAN was born from the integration of two major pieces of software DaSSF and BBN's WiroKit. Since both DaSSF and WiroKit were designed having ease of inter-operability as a primary goal from the start, these two pieces came together rather easily. While WiroKit provided the functionality for routing in wireless ad-hoc network models, DaSSF brought forward the structural cement that served to bind sub-models to one another. Figure 2 presents the main components of SWAN and shows the flow of data between them. From a broad perspective, our simulator is composed by four major kinds of sub-models: a Terrain Model, a Plume Dispersion Model, an RF Channel Model and a Node Model. Next, we describe in detail what they do and how they inter-operate. The Terrain Model is a static map that serves as the unifying point between the Plume Dispersion Model and the RF Channel Model. Since the evolution of both models can be subject to the geography of the terrain, for the sake of consistency, they must both be driven by the same description. This way, the same obstacle that stands in the path of radio waves will be present in the path of the chemical plume. In the current implementation of SWAN, we use at terrain, so both plume and radio signals can propagate freely over the simulated space. The Terrain Model remains, however, as a placeholder that will be important for the future development of the framework. The movement of air masses, which affect the evolution of the chemical plume with time, is described by the Plume Dispersion Model. Since this model is fairly isolated from the rest of the simulator, it can contain either the simplest or the most complicated descriptions of behavior. Due to the fact that it is only subject to interactions with a static terrain map, its states could actually be precompiled independently of the simulation of the sensor network. The Plume Dispersion Model provides the input that drives the wireless sensors. Considering that we have used at terrain, this model evolves independently of any other component in the framework, that is, it is driven by time alone. We have divided the terrain into square cells of side d and represented the state of each one by the level of chemical contaminant. For each cell, we compute the new contaminant level (after a fixed quantum of time elapses) by averaging the cell's own level and the level of its neighbours. We realize that this model alone can be extremely complex and computationally intensive. The point to keep in mind, however, is that, for our purposes, its presence in the framework is justified by a need to provide a realistic enough" stimulus to the sensor network. The precise evolution of the plume is not, in itself, one of our goals. As SWAN evolves, it may be interesting to instrument it so that network design can be automatically validated by comparing the evolution of the plume model against that of the sensed" plume. To create a Node Model to represent the smart sensors, we surrounded WiroKit with sub-models that represent entities that exist within the physical smart sensor devices. In this fashion, we have emulated the environment that WiroKit requires to run. As a router, WiroKit receives data packets from the Wireless Sensor Model, determines where they must be sent in order to eventually reach a Monitor node, and builds radio packets containing the routing information. The radio packets are, finally, passed to a radio modem which takes care of translating them into electromagnetic signals. In our model, we have done away with the modem, since, for our purposes, we don't need to reach down to that level of detail. This way, the output of WiroKit goes straight into our RF Channel Model. Conversely, WiroKit may receive input from the RF Channel Model: when packets traverse multiple hops to reach a Monitor, intermediate nodes will receive them, and then send them out in the appropriate direction. Also inside each node, it would be necessary to specify an Operating System Model. This component would emulate the functions of its counterpart found in smart sensors and, thus, deal with issues such as time keeping, thread management, and memory allocation. As it happens, the DaSSF runtime environment already provides these functions to the models that execute on it. Therefore, we didn't need to create an actual OS Model. When WiroKit needs to obtain the current time or allocate memory, DaSSF is called directly playing the role of the operating system. Also, when a WiroKit object is created in a DaSSF simulation, it passes the pointer to its main thread to the runtime environment. DaSSF will invoke the WiroKit thread as often as determined by the modeler. The RF channel model is a more complex and sensitive issue, since it bears so much direct innocence on the results of the simulation of the network. Since the quality and the complexity of this model are key to determining the viability of the entire simulation, we discuss it alone in the next section. We now proceed to describe the model for network nodes. GPS (Global Position System) provides current time and location information. As in the real GPS system, this information is globally synchronized across all nodes. Sensor provides the active element that measures the current local level of chemical activity. It also composes and transmits messages (datagram's) indicating the node identifier, GPS location and time, and current sensor reading. Monitor is the data sink", or collection point, for messages from the sensors. It creates logs with the messages received, associating with each one a GPS timestamp that indicates when it was received. Analyzing the evolution of the simulation, we can compare messages as sent by the sensor against messages as received at the monitor. In this fashion, we can determine how many messages were lost, compute statistics of transit delay, among other possibilities. IP simulates the Internet Protocol layer in a host computer. It provides IP datagram headers for each sensor message. The importance of this component lies in its relevance to the extensibility of the framework: in the future, the wireless network model can be made to inter-operate with a model for wired networks. Note that our present simulation does not employ TCP or UDP; instead it transmits messages over bare IP datagram's. Router is the ad-hoc wireless routing engine together with its associated forwarding engine. This is exactly the BBN-supplied WiroKit code, which is an actual router (see Section 3). As we've pointed out before, this contains code that is identical to what runs on real hardware, that is, on sensor nodes.

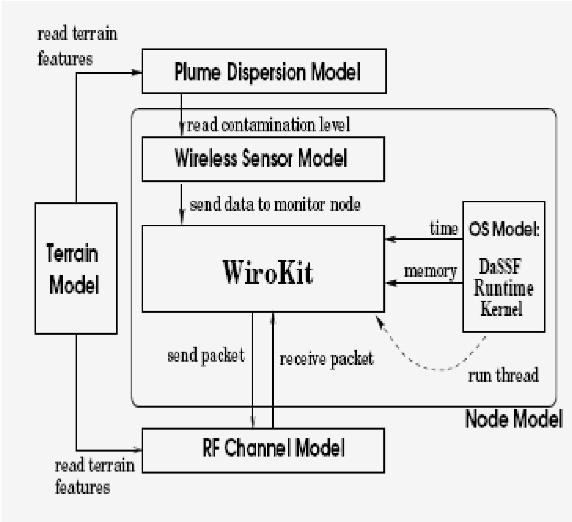

Fig. 2. Conceptual decomposition the SWAN framework

5. RF CHANNEL MODEL

Speaking in general terms, the RF Channel model describes the propagation of electromagnetic waves (radio signals) in geographical space. Accurate mathematical models for radio propagation can result in exaggeratedly heavy computations, which are, furthermore, difficult to partition for parallel processing. Rather than use classical approaches to RF modeling, such as those described in [17,18], we have devised a simplified model that is both novel and computationally efficient. Although more accurate, classical models for RF propagation are not scalable. The large number of nodes we desire to simulate in our sensor networks lead us to construct a channel model that, retaining only the characteristics most important to a packet radio network, scales up as needed. Our model substitutes the mathematical detail of time and distance dependent functions with stochastic equations that make it computationally manageable, and yet, expressive. Let us assume that our models work with a multi-access radio channel such as that defined by the IEEE 802.11 standard. From a networking viewpoint, the two most important characteristics of the channel are the packet delay and the probability of packet loss in transmission due to interference. As indicated in Figure 4, these two characteristics are quite different for unicast versus broadcast transmission. These differences arise because of the details of the 802.11 behavior. Allowing us an abuse of terminology, let us say that we can define channel busyness as a quantity that reflects how busy it has been within a recent interval of time. A unicast transmission is destined for a single receiver and is sent repeatedly until the intended receiver has acknowledged the successful received of the packet, or until some ceiling on transmission attempts has been exceeded. The probability of loss for unicast packets increases slowly and monotonically with the channel busy-ness in the vicinity of the receiver, since the sender will retry the transmission a number of times. Similarly, one can expect that channel delays will rise monotonically with channel busy-ness. The delay, however, increases quickly with the busy-ness because each such packet will need ever more retransmissions as the channel becomes busier. Conversely, a broadcast transmission is sent out to any radio receiver within range. Broadcast packets are, generally, sent just once, though in some implementations they may be sent multiple times to increase the reliability with which they are received. Due to factors beyond the scope of this paper, the probability of loss for broadcast is much higher from the start. This probability increases even further with channel busy-ness until it becomes quite small for a very busy channel. The delay, however, grows slowly since each packet is transmitted only once, or a few times, rather than repeatedly until an acknowledgement is received.

Bearing in mind these basic characteristics, we have implemented a novel channel model that emulates the behavior of 802.11, and that is, furthermore, highly parallelizable. Our RF channel model assumes that every time a message is received, we recomputed certain quantities to determine whether that message arrived successfully or not. To achieve this goal, whenever the k-th message arrives at a network node, we compute P_k^{loss} , an estimate for the probability of successful receipt. Figure 5 illustrates this channel model. Here we see that a receiver R is surrounded by a number of transmitters a, b, c, d, and e. Every time a message is sent toward R, either by unicast or broadcast, we first figure out whether transmitter is within distance of R.

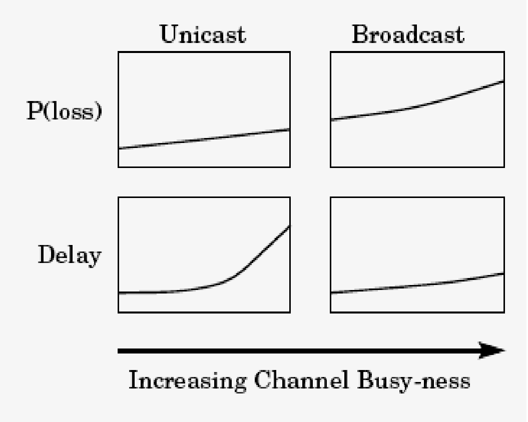

Fig. 4. Performance of a Radio Channel

For the k-th message that is sent to R, we define dk as the time between the arrival of this message and the arrival of the previous message, the (k 1)- th. When messages are sent from transmitter inside the circle of radius _ (such as a, c, d, and e, in Figure 5) with receiver R at the center, they are simply discarded. Otherwise, if the transmitter is within a distance from the receiver, messages are delivered or not according to a Bernoulli random variable with parameter P_k^{loss} k . Although the complete derivation of P_k^{loss} k lies outside the scope of this paper, we now give the reader a brief sketch of how it can be obtained.

Let us define the busy-ness of a channel in the vicinity of a receiver as a measure of how utilized the radio spectrum currently is and has been in the recent past. Formally, for a receiver R, we define busy-ness at the arrival of its k-th message as:

$$B_k(\delta_k) \;=\; 1 + e^{-\lambda\,\delta_k}\; B_{k-1}(\delta_{k-1}).$$

This measure of busy-ness basically indicates the number of "active" messages in the channel. It increases by one due to the new message just sent and retains a decaying memory of those previously sent.

6. CONCLUSIONS AND FUTURE WORK

We have presented the architecture of a scalable framework for the simulations of wireless ad-hoc networks. This project represents the coming together of two major pieces of software. DaSSF, the high performance, scalable simulator developed at Dartmouth College, served, mainly, as the structural glue that allowed sub-models to inter-operate. It provided, not only the infrastructure for data exchange, but more importantly, for the synchronization of all components. WiroKit, the portable router from BBN, was easily integrated with other sub models thanks to its few and well-defined points of contact. It was created to be portable not only across different wireless platforms, but also easily transportable into simulation test beds, allowing the direct execution of routing algorithms at source code level. The result of this project was more than the sum of its parts. Through experiments with our Simulator for Wireless Ad-hoc Networks (SWAN) we have demonstrated its functionality and scalability. Using the scenario of a natural catastrophe or terrorist attack, where a plume of hazardous material is carried over the landscape, we have shown that this framework can be of great help to study the performance of routing algorithms for networks of smart sensors. Our experiments have exposed network properties, namely throughput and packet delay as functions of traffic and network configuration. These results of these experiments have been paramount to validating the our model. We were able to observe congestion through packet losses and packet delays, characteristics that reflect the choices of routing algorithm and network configuration. Future directions for our work will follow two main paths, in parallel. In the first one, we will refine the sub-models in the framework and add components for which, at this stage, we included only placeholders. As next natural steps in this development, we can cite the development of a model that implements the IEEE 802.11 standard, and a benefited model for RF propagation and interference. The second main path in our future research will involve the development, refinement and evaluation of different sensor network designs. We will be looking into issues such as routing algorithm design, efficient strategies for data pre-processing, collection, and aggregation, as well as the real-time visualization of the state of the process being monitored.

REFERENCES

[1] Institute for Security Technology Studies (ISTS), Dartmouth College. http://www.ists.dartmouth.edu.

[2] Scalable Simulation Framework (SSF). http://www.ssfnet.org.

[3] Smart Dust. http://robotics.eecs.berkeley.edu/pister/SmartDust/.

[4] Lokesh Bajaj, Mineo Takai, Rajat Ahuja, and Rajive Bagrodia. Simulation of large-scale heterogeneous communication systems. In Proceedings of IEEE Military Communications Conference (MILCOM '99), November 1999.

[5] L. P. Clare, G. Pottie, and J. R. Agre. Selforganizing distributed microsensor networks. In Proceedings of SPIEs 13th Annual Interna- tional Symposium on Aerospace/Defense Sensing, Simulation, and Controls (AeroSense),Unattended Ground Sensor Technologies and Applications Conference, April 1999.

[6] Michael G. Corr and C. M. Okino. Networking recon_gurable smart sensors. In Proceedings of SPIE: Enabling Technologies for Law Enforcement and Security, November 2000.

[7] James Cowie, David M. Nicol, and Andy T.Ogielski. Modeling 100,000 nodes and beyond: Self-validating design. In DARPA/NIST Work-shop on Validation of Large Scale Network Sim-ulation Models, May 1999.

[8] James Cowie, David M. Nicol, and Andy T.Ogielski. Modeling the global internet. Comput-ing in Science & Engineering, 1(1):42{50, 1999.

[9] James H. Cowie. Scalable Simulation Framework API Reference Manual, 1999. http://www.ssfnet.org.

[10] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Scalable coordination in sensor networks. In ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom '99), pages 263{170, August 1999.

[11] Wendi R. Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-e_cient communication protocol for wireless microsensor networks. In Proceedings of the Hawaii International Conference on System Sciences, January 2000.

[12] V. Hsu, J. M. Kahn, and K. S. J. Pister. Wireless communications for Smart Dust. Technical Report Electronics Research Laboratory Technical Memorandum Number M98/2, Electronics Research Laboratory, University of California at Berkeley, February 1998.

[13] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Mobile networking for Smart Dust. In ACM/IEEE Intl. Conf. on Mobile Computing and Network ing (MobiCom '99), August 1999.

[14] J. Kulik, W. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. In ACM/IEEE Intl. Conf. on Mobile Computing and Networking (Mobicom '99), August 1999.

[15] Jason Liu and David M. Nicol. DaSSF 3.0 User's Manual, January 2001. http://www.cs.dartmouth.edu/research/DaSSF/papers/dassf-manual.ps.

[16] K. S. J. Pister, J. M. Kahn, and B. E. Boser. Smart Dust: Wireless networks of millimeterscale sensor. Highlight Article in 1999 Electronics Research Laboratory Research Summary,1999.

[17] Theodore S. Rappaport. Wireless Communications Principles & Practice. Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[18] Gordon L. St□uber. Principles of Mobile Communication. Kluwer Academic Publishers, Norwell, Massachussets, 1996.

AUTHOR BIOGRAPHIES

Raman Chadha having 13 years' experience in teaching and 1 year experience in an industry. His Area of Research includes Data Structures, Programming in 'C', C++ , Artificial Intelligence and Mobile Ad-hoc Networks. Presently working as an Astt. Professor(HOD, IT, MCA) in Vidya College of Engineering, Meerut. He has authored more than 8 books. He is pursuing PhD from

Monad University, Village & Post Kastla Kasmabad, Pilakhwa-Tehsil Hapur (U.P), under the guidance of Dr. Satyadev Garg (I.T. Consultant, TCS, Noida). His area of Research is ”**PARALLEL SIMULATION OF LARGE-SCALE WIRELESS NETWORKS”.**