Ethersim: a simulator for application-level performance modeling of wireless and mobile ATM networks

Mani Srivastava a,*, Partho Mishra b, Prathima Agrawal c, Giao Nguyen d

a EE Department, UCLA, Los Angeles, USA
b AT&T Laboratories, Florham Park, USA
c AT&T Laboratories, Whippany, USA
d CS Department, UC, Berkeley, USA

Abstract

The paper describes Ethersim, a simulation tool to model and study the performance of multimedia-oriented integrated service ATM networks with mobile hosts and wireless links. The key motivation behind Ethersim is to study the application-level impact of host mobility and wireless channels. Ethersim has a discrete event based simulator core and incorporates models of user applications and transport, network and MAC layer protocols. It provides the capability to specify a cellular wireless ATM network topology and host mobility patterns. The software architecture of Ethersim employs five special entities: an air module, a map, a mover, mobile hosts, and base stations. We also present case-studies of using Ethersim to model and study the interaction of transport layer, connection rerouting protocol, and radio characteristics in the SWAN [P. Agrawal, A. Asthana, M. Cravatts, E. Hyden, P. Krzyzanowski, P. Mishra, B. Narendran, M. Srivastava, J. Trotter, SWAN: A Mobile Multimedia Wireless Network, in: IEEE Personal Commun. Mag., April 1996] mobile and wireless ATM based multimedia network. © 1998 Elsevier Science B.V.

Keywords: Mobile ATM simulation; Wireless ATM simulation

1. Introduction

The increasing deployment of wireless access technology along with the emergence of high speed integrated service networks promises to provide mobile users with ubiquitous access to multimedia information in the near future. However, transforming this vision into reality poses several problems, of which one of the most challenging is how to redesign the existing network protocols currently used in wired networks so as to allow seamless end-to-end communication over networks with both wireless and wired links. This design process requires an understanding of how network and multimedia application performance is affected by the choice of algorithms and protocols at various layers of the network hierarchy, the characteristics of the wireless links and their differences from wired links, the presence of mobile hosts, and the mobility patterns of these hosts.

Traditionally, wireless links characteristics and higher layer protocols for multimedia networks have been studied in isolation. For example, the performance of wireless links has been studied with the primary focus being on understanding and improving the performance of the physical and medium-
access-control (MAC) layer protocols. Similarly, the performance of higher layer protocol suites such as ATM and TCP/IP have been studied in context of the high speed integrated service wired networks. Only recently has there been interest in studying the impact of wireless link characteristics and host mobility patterns on higher layer protocols, such as studies by various researchers on the interaction between wireless communication and the TCP protocol [11.5,26.6]. These studies have demonstrated that temporary disruptions in the physical connectivity due to noise and fading in wireless links, and handoffs of a mobile user from one base station cell to another in a cellular network, can lead to long lived disruption in communication at the user level because of specific assumptions made by higher layer protocols, such as TCP. Unfortunately, since most of these studies have been conducted using actual implementations, they have not been able to extensively modify protocol policies, host mobility patterns, or link characteristics to generalize their conclusions. These complex systems are also analytically intractable, leaving simulation as often the only reasonable alternative methodology to extend analysis and obtain generalized conclusions.

In this paper we describe Ethersim, a tool designed to model application-level performance of integrated service networks with mobile hosts and wireless links. The tool grew out of a need to model and study the performance of various alternative protocols and algorithms for a wireless and mobile ATM based multimedia network called SWAN [2,3] that our research group previously developed at AT&T Bell Laboratories. In particular, we wanted to study the interaction of adaptive applications, transport layer protocols, connection rerouting schemes, and radio characteristics in a SWAN-like wireless and mobile ATM system.

Ethersim has been built using a discrete event based simulator core and incorporates models of user applications and transport, network and MAC layer protocols. It provides the capability to specify network topology and host mobility patterns. It provides the capability to specify network topology and host mobility patterns. There are five special entities in Ethersim relevant to modeling mobility and wireless communication: an air module, a map, a mover, mobile hosts, and base stations. The air module models the physical air interface effects (e.g., RF power decay, frequency collisions, etc.). The mover is a central entity that moves the mobile hosts on the map. Ethersim allows for both random and goal-directed movements of mobile hosts, and also allows synchronized goal-directed movements to model conference room type mobility patterns. Ethersim is structured in a modular fashion to permit functionality at different levels of the protocol stack to be modified independently, thereby allowing network protocol designers to study the interaction between policies embedded in the protocols at different layers. Ethersim also includes various performance measurement and graphical user interface routines to interpret the simulation results.

1.1. Related work

By and large the available simulators that are relevant to our work fall into two broad categories: wireless link simulators and network simulators. The former focus on detailed modeling of the wireless link characteristics such as RF propagation, noise, and fading, but provide no support for modeling higher network layers. The latter focus on networking algorithms and protocols, using fixed network topologies with static hosts, and simple link models that may be adequate for wired links but are inadequate for wireless links.

Clearly, neither of these two classes of simulators is adequate for modeling wireless and mobile networks. Realizing this, various industry and academic groups developing such networks have resorted to one of three approaches to their modeling and simulation. The first approach is the use of custom developed C function libraries or C++ class libraries to create stand-alone ad hoc programs that model and simulate specific networks, such as for GSM [25]. Such custom libraries, however, cannot serve the purpose of a general simulation tool.

The second approach is to combine wireless link simulation and network simulation using software environments that either provide a mixed-domain simulator or allow multiple simulators to be interconnected using a simulator backbone. As an example, the Ptolemy [10] software environment from
Berkeley provides multiple simulation domains, each corresponding to a different model of computation. Available domains include synchronous dataflow, dynamic dataflow discrete event, process networks, VHDL, etc. The Ptolemy simulator kernel orchestrates the schedulers corresponding to each of the different domains. Different parts of a system can be modeled using the domain that is well suited to it. For example, the wireless link may be modeled using one of the dataflow domains while the higher network layers may be modeled using the discrete event domain. Berkeley’s Infopad wireless and mobile system project [7] used Ptolemy for some of its wireless link simulation and modeling requirements. Overall, with their roots in block diagram simulators in DSP and communications, Ptolemy and other such simulators have weaker support for network models. Also, while flexibility is a strong suite of the approach of combining multiple simulators or multiple simulation domains, it does come at a cost of efficiency.

The third approach is that of a single simulator consisting of a discrete event kernel core with entities to model various elements of a mobile and wireless network, such as the mobile hosts, the wireless link, etc. Ethersim falls into this category.

Another simulator in this category from the literature is the recent mobile wireless network simulation environment Maisie [23] used for the WAMIS mobile wireless multimedia system project at UCLA [15]. Maisie is actually a general purpose discrete event simulator capable of parallel execution on multiprocessors. It uses a specialized message-passing parallel simulation language, also called Maisie. Since Maisie has been used by various research groups for modeling wireless and mobile systems under DARPA’s GloMo research program [18], a variety of models for mobile nodes, wireless channels and radios, operating system, application-specific traffic source, and network algorithm have been created by various researchers. However, reflecting their diverse origins and the inherent general-purpose nature of Maisie, these available models are not designed to all work together as a consistent library.

On a more ad hoc basis, various researchers have also used patches and tricks to coax popular network simulators, such as ns from UC Berkeley and Lawrence Berkeley National Labs (see http://www-nrg.ee.lbl.gov/ns/ and http://www-mash.cs.berkeley.edu/ns/), to model wireless channels and mobile hosts. For example, a simple wireless channel can be modeled in ns as a noisy shared channel and a CSMA based medium access protocol via an extension written by one of the co-authors (Giao Nguyen). Host mobility can be modeled in ns by using its frontend scripting language Tcl to change the network topology during the course of the simulation. In principle, more elaborate Ethersim-like notions of host location, maps and mobility traces may be programmed into ns via Tcl. However, ns and similar simulators have no direct support for mobility or shared wireless radio channels.

Finally, commercial simulators, such as OPNET, have also begun to provide modules for wireless and mobility added onto a conventional network simulator.

Ethersim, which is built on a generic discrete event kernel, is distinguished by a much general and richer variety of network components such as hosts, links, switches, and ATM and TCP/IP protocol modules that allow the modeling of a variety of mixed wired and wireless network scenarios. Instead of leaving the modeling of wireless link and the mobility entirely to the various components of the network defined by the user, Ethersim supports wireless and mobility aspects in an integral fashion as first class citizens. The notion of an air module, a map module, and a mover module are built into Ethersim, and provide efficient modeling of wireless and mobility. For example, the map and the mover together provide modeling of the geographical topology of the network and a variety of mobility patterns without burdening the user defined host models with them. Instead, an interface in terms of pre-defined mobility events is defined to the hosts.

1.2. Paper organization

The rest of this paper is structured as follows. In Section 2 we describe the mobile and wireless network model that we assumed in designing Ethersim. In Section 3 we describe the software architecture of Ethersim, and the implementation and capabilities of a few key modules. Section 4 presents case studies meant to illustrate the capabilities of Ethersim. In Section 5 we draw conclusions from our experience.
2. Mobile and wireless network

Ethersim uses the reference network model, shown in Fig. 1, comprised of a wired part and a wireless part. The wired part is composed of switches and wired static hosts, with point-to-point wired links connecting the hosts to switch ports, or one switch port to another switch port. This is similar to the structure of modern high-speed switched networks, such as ATM. It must be noted that the wired part of the Ethersim network model is intrinsically based on switches and point-to-point links, and does not support shared medium networks such as older non-switched Ethernet where multiple hosts may be on the same wired link. Examples of real-life systems that use such networks modes include Bell Lab’s SWAN [3], Berkeley’s Infopad [7], ORL’s Radio ATM [21], etc.

The switches and the hosts incorporate network and lower layer protocols. These layers are responsible for the signalling functions required to setup and tear down virtual circuits in addition to the transport functions of packet transmission, reception and forwarding. The hosts also have transport protocol modules associated with them which perform the functions of flow control, reliable data delivery, etc., depending on application requirements. Hosts are assumed to have applications running on them which act as traffic sources or sinks.

Some of the switches in the Ethersim model are special because they also have one or more ports equipped with radio interfaces. Such switches are called basestations. Basestations can be viewed as being located at the topological edge of the wired network, with their radio ports providing access to the wireless world. Physically, the basestations will be distributed in a geographical region with the antenna corresponding to each basestation radio port covering a geographical neighborhood whose size will depend on factors such as radio transmitter

Fig. 1. Mobile and wireless network model in Ethersim.
power, radio receiver sensitivity, free-space radio attenuation, multi-path fading, etc.

Basestations lead us to the final physical component of the network model – wireless mobile hosts. These hosts are derived from the standard wired host by replacing the wired network link interface with a radio. Just like wired hosts, the wireless mobile hosts also have the ability to participate in network signalling and data transfer protocols. Freed from its wired tether, a radio-equipped host can geographically move with the user who is carrying it. Many different types of motion patterns are possible in real-life, ranging from movements of an isolated user to ganged movements of a set of users who are going to or leaving a meeting. Further, a variety of terrain are possible in which a user can roam. This primarily affects the attenuation of radio signals as they travel from a transmitter to a receiver. The presence of walls and other obstructions can render the simple inverse-squared free space attenuation model useless.

At any moment, a mobile host is associated, or registered, with a proximate basestation (typically the nearest) with which it can communicate. The mobile sends or receives all its traffic to or from the wired network through the basestation at which it is registered. Multiple mobiles may be registered at a basestation. As a mobile host roams geographically, it may move from the vicinity of one basestation to another, and thus change the basestation with which it is registered. Such roaming requires the network protocols to be mobility aware. For example, in case of an ATM network, the virtual connections originating from or terminating at a mobile host need to be rerouted every time a mobile host moves [9,17,19].

In general, communication using radios (between the mobile and basestations) is a much more complex phenomenon than simple point-to-point communication using wired links. This is largely because all radio communication takes place in an inherently broadcast medium. Depending on factors such as compatibility of frequency, modulation scheme, coding, distance, transmitter power, receiver sensitivity, free-space attenuation, multi-path fading, etc., a radio transmitter on a mobile host or a basestation can communicate with a radio receiver on another mobile host or basestation, and can cause interference in the communication between another transmitter and a receiver.

In the Ethersim network model, a radio may have multiple channels and the radio may be configured to operate on one of the channels at any time. A channel may correspond to a CDMA code in a direct sequence spread-spectrum system, or to a frequency hopping pattern in a slow frequency-hopping system, or to simply a frequency slot. In a full-duplex radio transceiver, the channels of the receiver and the transmitter halves can be independently set. In a half-duplex radio transceiver, only one half is active.

---

![Fig. 2. General multi-channel organization of a basestation in Ethersim model.](image-url)
at any given time. For a radio transmitter and a radio receiver to communicate, the two must be on the same channel. Therefore, all the mobiles registered at a basestation radio port must be on the same channel. As shown in Fig. 2, a basestation may have multiple co-located radio transceivers and can support multiple channels. Basestation radio ports, whether on the same basestation or different ones, that are in the same vicinity will typically be on different channels to take advantage of spatial multiplexing. When a mobile moves, it may have to register at a radio port at a new basestation on a new channel in order to maintain connectivity.

Channels in the Ethersim model are intrinsically broadcast oriented because multiple transmitters can be active on the same channel. If two transmitters T1 and T2 are active at the same time, then T2 can potentially cause interference at T1’s intended receiver R1 if the signal strength of T2’s radio transmission received at R1 is high enough compared to the strength of signals received from T1. Various factors such as free space attenuation, multi-path fading, and obstructions such as walls affect the strength with which a transmitter signal is received at a receiver. To coordinate such interference intra-channel interference, medium access control (MAC) protocol module is associated with each radio to allow multiple transmitters to share a channel. Token passing, and collision detection based multiple-access protocols are examples of MAC protocols [8,12, 14,16].

While the MAC protocol helps eliminate or minimize interference from other transmitters on the same channel by providing a multi-access mechanism along the time dimension, a second mechanism of interference is between transmitters on different channels but same frequency. Ethersim channels are a way to divide the spectrum in the frequency and code dimensions. In frequency division systems, channels operate on different frequencies and are thus orthogonal with no possibility of inter-channel interference. In code division spread spectrum systems, channels operate on same frequency but with different orthogonal codes. Increasing transmit power in one channel raises the noise level in other channels operating with different codes. In slow frequency hopping system, different channels cyclically hop through different sequences of frequency slots. The hopping sequences are chosen from a weakly orthogonal set to reduce the chance of different channels colliding by hopping to the same frequency. But, clearly, frequency collisions cannot be eliminated when asynchronously hopping transmitters are in the same neighborhood.

The wireless part of the Ethersim network model is quite complex, with crucial roles being played by the division into multiple channels, medium access control, intra-channel and inter-channel interference, and radio signal attenuation. This is quite unlike the simple wired link.

2.1. Factors in application-level performance

From the discussion of the network model above, it is clear that a number of factors unique to mobile and wireless networks can potentially affect the application-level performance. First are the mobile network protocols, such as mobile-IP and mobile-ATM. Second are the characteristics of wireless links, including bandwidth, large-scale and small-scale propagation loss, receiver sensitivity, multiplexing technique (FDMA, TDMA, frequency hopping, direct-sequence spread spectrum, etc.), medium-access protocols, and hand-off protocols (hard vs. soft hand-off). The third factor is the flexible patterns of host mobility, which may include patterns such as random and directed movements of mobile users, and convergence and divergence of roaming mobile to/from a meeting location. The interaction of these mobility patterns with geographies with a large number of cells can have serious impact on application performance that would need to be modeled. Finally, crucial to modeling application performance are the application models themselves. Many applications, particularly multimedia applications, adopt sophisticated adaptive control mechanisms to combat time varying conditions such as in a radio channel. These control techniques include adaptation of play-out control mechanism at the receiver, and a proper modeling of such application behavior is important in an overall modeling of mobility and wireless.

3. Ethersim software architecture

The Ethersim software provides a modular simulator which supports all aspects of the wireless and
mobile network model described in the previous section, while allowing functionality at different levels of the protocol stack to be modified. At the core of the simulator is an efficient and generic discrete event kernel. The kernel manages the scheduling of events using an event queue. The core functionality is similar to any discrete event kernel – an entity can request an event to be sent to another entity at a future time (simulator time), and the kernel arranges for the delivery of the right event at the right time.

The interesting part of the simulator consists of the built-in software modules in Ethersim that use the discrete event kernel core to provide standard services for user defined and instantiated simulation entities. These standard services take the form of pre-defined events and messages to be generated or handled by the user code. The main software modules are: wired static host, switch, wired link, base station, wireless mobile host, air, map, mover, traffic sources (statistical and trace driven), protocol modules (transport, connection establishment, and packet scheduling), measurement modules, and graphical user interface modules. The wired static host, switch, basestation, and wireless mobile host, incorporate user customizable protocol sub-modules for connection setup/teardown, packet scheduling and data transport functions. The wired static host and wireless mobile host modules have associated transport protocol modules, including ones that accurately model the TCP transport protocol and an experimental video transport protocol. Both wireless and wired hosts also have associated statistical and trace-driven traffic sources. The basestation and the mobile host have attached to them user customizable MAC protocol modules. In addition to these primary modules, there are two supporting modules, measure and GUI, to provide performance statistic gathering and graphical I/O services to user code. User customization of the various protocol modules takes the form of writing C functions. The protocols in different layers can be modified independently, thereby allowing network protocol designers to study the interaction between policies embedded in the protocols at different layers.

Of the main software modules, only the basestation, wireless mobile host, air, map, mover, and selected protocol modules (connection rerouting, MAC, and location management) are relevant to wireless communication and mobility, and will be the focus of our discussion. The global relationship of all these software modules and sub-modules is shown in Fig. 3. The following subsections describe

![Diagram](image-url)

Fig. 3. Relationship between software modules in Ethersim.
the key mobility and wireless related software modules.

3.1. Map: models of geography

The map module is used to define a geographical region, and the placement of various wireless entities (basestations and mobile hosts) in it. The map is constructed using an undirected graph data structure whose nodes represent arbitrarily sized geographical regions referred to as rooms or cells, while edges represent inter-cell topological adjacency relationship. There is at most one edge between a pair of nodes. Nodes in the graph represent radio regions that correspond to geographical areas that are either dead-spots or have one main basestation, or are in the overlapping region within coverage of more than one basestation. The graph need not be planar, so that 3-dimensional adjacency relationships can easily be described. This representation provides a simple and efficient representation for modeling cellular radio coverage that is detailed enough for studying application-level performance impact without being so detailed as one would expect in a low-level simulation.

Associated with the map are several parameters. One parameter, associated with each edge in the map, is the RF loss coefficient \( L \), where \( L(a,b) \) represents the attenuation of an RF signal generated in cell \( a \) and received in cell \( b \), and vice versa (i.e. \( L(a,b) = L(b,a) \)). By using suitable values of the loss coefficient and cells representing suitably small geographical regions, the effect of walls and obstructions is modeled — one is not restricted to free space attenuation. Attenuation between two non-adjacent cells is calculated by finding the path between the two cells with minimum loss coefficient, where the loss coefficient of a path is the product of the loss coefficients associated with the edges constituting the path. Clearly, Ethersim’s approach of modeling signal loss is a simple abstraction of the complex physical world where signal loss is a result of a complex mix of static and dynamic factors such as free space attenuation, shadowing, multipaths, slow-fading, fast-fading, etc. However, such abstractions are needed when one is trying to study the interactions of various layers from the physical layer all the way up to the transport and applications layers. Otherwise the simulation time would be unreasonable.

Other parameters associated with the map are related to the mobility aspect of the network model. Each edge connecting a node to another node is viewed as a door from one cell to another. A door, therefore, is a node-edge pair. Associated with each door \( d \) is a probability \( P_d(d) \) of a mobile host entering the door (i.e. using the door to enter the cell). Clearly, the sum of probabilities of entering all the doors in a cell, \( P(d_1) + P(d_2) + \cdots + P(d_n) \) should be less than 1, and the difference \( 1 - (P(d_1) + P(d_2) + \cdots + P(d_n)) \) corresponds to the probability of the mobile host staying in the cell. The probability of staying, in turn, can be used to determine the time a mobile spends in a cell. Also associated with each edge is a distance \( D(a,b) \), where \( D(a,b) \) represents the distance that needs to be travelled by a mobile going from cell \( a \) to cell \( b \).

The basestations are associated with a cell in the map, thus modeling the effect of placing the radio port of a basestation at a specific geographic location. The mobile hosts are also associated with a cell in the map, but this association changes with time as a mobile moves from one cell to another via doorways.

3.2. Mover

The map specifies where the mobiles can be, and the paths they can take. The mover is a behind-the-scene entity which moves the mobile hosts on the map. It does so taking into account both the map parameters, and the movement policy. Several movement policies are currently implemented. One is random movement, where the transition probabilities in the graph associated with the map determine where and when to move. A mobile host repeatedly draws a uniform random variable and based on the probabilities of entering the various doors decides whether to exit a cell through one of the doors or to stay in the cell. Another policy is goal-directed movement where a mobile host is given a goal cell, and it moves towards it via the shortest path (as defined by the distances associated with the map edges). The speed of movement is decided by the speed of the mobile host (a mobile host attribute) and the edge distances (map attributes). The third policy is group movement which is intended to
model meeting room scenarios. A subset of mobiles are designated to belong to a group, and they all either move towards a common goal such as a meeting room at the same time (converging mobiles) using a synchronized goal-directed movement, or all move away from a node in a mix of random and goal-directed movements (diverging mobiles).

The mover maintains all the necessary per-host state on behalf of each mobile host, and decides when and where to move each mobile host. The per host state includes speed, destination, and group. When the mover moves a mobile host, it generates a sequence of pre-defined events (HOST_ENTER, HOST_LEAVING, and HOST_LEAVE) to the mobile host as well as to the air module. The latter is sent the events so that mobility related wireless link recalculations can be triggered, and appropriate events generated for basestations and mobile hosts.

3.3. Basestation: a switch with radios

Basestation, as mentioned earlier, is a switch with radios on a subset of its ports. Like any switch, it maintains a routing table using which it routes packets coming in on an input port to an output port where they are queued in a buffer for eventual transmission. Some of the packets may be signalling messages, which are routed to a signalling sub-module instead of being sent to an output port. The signalling sub-module realizes various protocol state machines, such as for connection set-up. This description is true of an ordinary switch as well. A basestation, however, has two enhancements. First, the signalling sub-module has to be mobility aware and execute protocols that can deal with mobile hosts. Second, a basestation also has MAC (medium access control) protocol sub-modules associated with each radio port. Fig. 4 shows the structure of the basestation software module.

3.3.1. Connection manager sub-module

The Connection Manager sub-module, shown as CM in Fig. 3, implements several alternative protocols for connection management and rerouting in the presence of mobility. This allows, for example, the modeling of a variety of mobile ATM scenarios. The implemented protocols include those proposed by us in [19] and being used in the SWAN wireless and mobile ATM system [2,3], as well as various other approaches. While new protocol modules for connection management and rerouting can be written by the user, Ethersim defines the basic interface. When a mobile host registers at a new basestation via a REGISTER signalling message, a REROUTE signalling message is sent by the new basestation to the old basestation, along with a list of connection ids which need to be rerouted. The CM at the old basestation selects and takes appropriate actions to reroute the said connections to the mobile hosts new basestation. This is done in cooperation with the CM at other interested basestations, switches, and hosts. The cooperation takes the form of more signalling message exchange. Some of the rerouting schemes
which are available in Ethersim include: total connection rebuild, partial connection rebuild to a fixed anchor basestation or switch, partial rebuild to a suitably chosen ancestor basestation or switch, connection extension from old to new basestation, etc. In addition, there are options to (i) remove connection loops in the wired network in the connection extension scheme, (ii) either throw away packets in transit during rerouting, or to buffer the packets in transit at basestations and switches, and (iii) allow mobile host to trigger a partial rebuild based on connection-level performance measurements. The complexity of the implementation lies in the need to ensure that packet ordering is maintained throughout the process – this is mandated by connection oriented networks such as ATM. The detailed finite state machines for some of the rerouting schemes of Ethersim are described in [19].

3.3.2. MAC sub-module

When an incoming packet is routed by the basestation to an output port with a radio, the packet is put in a transmit queue associated with the output port. The MAC sub-module associated with the radio at that output port serves the transmit queue by attempting to transmit the packets in accordance to a medium access control (MAC) protocol which coordinates access to the wireless channel on which there may be other radio transmitters. On the incoming side, the MAC sub-module passes to the basestation all packets received from a radio transmitter, except for MAC signalling messages which are intercepted and acted on. The MAC protocol may be user defined, and involves exchange of signalling messages with other MAC sub-modules (at basestations and mobiles) on the same channel.

3.4. Mobile host

The mobile host is derived from the wired host, and subsumes all its functionality. In addition, however, it has three additional enhancements. First, the signalling sub-module is mobility aware and implements the host side of the rerouting protocols such as connection rebuild, connection extension, etc. Second, it has an associated MAC sub-module through which all the incoming and outgoing packets pass. The MAC sub-module implements the host-side of MAC protocols such as token passing. Third, the mobile host needs to handle mobility. This first two enhancements are similar to the corresponding enhancements in the basestation. We focus on the third: mobility. As mentioned earlier, the mobile hosts are moved on the map by the mover. The mobile hosts carry some state data and parameters, such as speed, to allow the mover to move the hosts meaningfully. In addition, associated with each mobile host is a current basestation, which is the strongest basestation heard by the mobile host. After moving a mobile host, the mover and the map together recalculate the strongest basestation and update the current basestation. In addition, HOSTLEAVE and HOSTENTER events are sent to the old current basestation and the new current basestation. These events are made available to allow simulation of rerouting protocols without necessarily having low level MAC handoff protocols available – HOSTLEAVE and HOSTENTER can be used in lieu of a handoff protocol. Actually, the mover and map also provide a HOSTLEAVING event which is sent to a basestation when the mobile host enter a geographical area where the radio ranges of two basestations overlap. In this overlap area the mobile host communicate with either basestation, although it is registered with only one of them. In real systems such overlap regions are used to implement soft handoffs. The HOSTLEAVING event allows simple emulation of soft handoffs for purposes of high level simulation. In short, the HOSTENTER, HOSTLEAVING, and HOSTLEAVE events which are generated by the map and the mover based solely on radio consideration (transmit power and attenuation) can be used as an abstraction of a low level handoff protocol. Such abstractions simplify and speed up the simulation when the focus is the performance and interaction of higher layer protocols.

3.5. Air module: modeling of the wireless link

The Air module (Fig. 5) is the key module responsible for the wireless aspects of Ethersim. Functionally, the Air module can be viewed as a giant MAC level switch. It receives as input packets from the sending MAC sub-modules at various basestations and mobile hosts, and routes them to appropriate receiving MAC sub-modules. Every incoming
packet carries with it information about the frequency, spread-spectrum code (optional), and power with which it was transmitted. Every outgoing packet delivered to a receiver carries with it information about the signal strength at the receiver. Also, associated with the radios is the wireless link data rate, from which the transmit duration of a packet is calculated by the Air module. In addition to the switch-like functionality, the air module also degrades packet signal strength to model large scale propagation loss, introduces noise errors to model white Gaussian noise, and detects and marks colliding packets on same frequency at each receiver location. A packet is delivered by the air module to a receiver if it is on the same frequency band and spreading code, if signal-strength is greater than receiver sensitivity, if signal-to-interference ratio is greater than a threshold value, and if there is a MAC address match at the receiver or if the receiver is operating in a promiscuous mode.

The key concern in the Ethersim implementation is minimizing the number of events generated for tasks that need to be done on a per packet basis. Tasks at larger time scales have only second order effects, and are therefore less important. For example, map recalculations done after a mobile host move are at much larger time scales and need not be optimized much. However, since the Air module does per-packet processing and is meant to model a broadcast medium, it is crucially important to have a smart filtering policy in the Air module itself to cut down on the number of events generated by each packet transmission. This is implemented by pruning the number of receivers to whom a packet is delivered based on considerations such as frequency, received signal strength (RSS), interference, noise, etc.

For example, if a packet is sent in frequency slot $f$, then only receivers tuned to frequency slot $f$ will get that packet. Similarly, a packet corrupted by noise (the Air module includes channel noise models) is just dropped. Finally, if signals from more than one transmitters on the same frequency overlap in time with high enough strength at a receiver, they interfere and only garbage can be heard. This might be the case, for example, in frequency hopping radios where two transmitters may transiently hop on to the same frequency. Air module arranges for packets that suffer from such interference to be dropped as well. This filtering process is an implementation optimization which is a key to the efficiency of the Air module. To further avoid checking all the receivers for matching criterion during filtering, a dynamically updated centralized table is maintained to point to all receivers that are listening on a given frequency at a given time. This table is indexed by frequency, and is updated only when the frequency is updated due to, for example, frequency hopping. The first filtering criterion is therefore the transmit frequency of a packet, and the central table is used to get a list of eligible receivers. Each receiver in this set is then checked to see if the remaining criteria, such as the $\text{RSS} > \text{receiver sensitivity}$, are satisfied.

The receiver signal strength (RSS) is calculated by the Air module for each eligible receiver by multiplying the packet transmit power by the total path loss between the transmitter and the receiver. The path loss is obtained from the map (using the RF loss coefficients). To speed up the RSS calculation for each packet, Ethersim pre-computes a table of the total path loss between any two cells by using a shortest path algorithm with the minor modification that the cost of a path is a product instead of a sum. Since each radio has a cell id associated to it by Ethersim (the id is dynamically updated by the mover.
in the case of mobile hosts), the total path loss is quickly obtained by indexing into the path loss table with the cell ids.

After the frequency and signal strength criteria are satisfied, the Air module next checks for interference with an earlier transmission. To do this, each receiver has associated with it the latest end time of all the packet transmissions on each frequency slot. If the transmission time of the packet under consideration starts before all previous transmissions are finished on that frequency at the receiver, both the new packet and the previous packets with which it is colliding are declared corrupted for that particular receiver. Since RECEIVE events might be generated for packets that are later corrupted by colliding with a new packet, these receive events are recorded by the Air module and later deleted when necessary. Associated with every packet is a MAC address for the intended radio receiver, with a special MAC broadcast address also being available. As is obvious from the preceding discussion, the air medium is inherently broadcast. A packet with a non-broadcast (unicast) MAC address can be heard by other receivers as well and cause interference. Clearly the packet itself is of no use to receivers other than the intended one. Therefore, the Air module considers all packets for calculating interference and collisions, but generates actual RECEIVE events only for those receivers that match the MAC address criterion. This is another form of packet filtering by the Air module which further improves its efficiency.

The final filtering step in the process of packet delivery is for the Air module to decide whether noise has corrupted the packet. Using a noise model based on a single bit error rate parameter, the Air module emulates a random process that possibly corrupts a packet. If the packet is corrupted by noise at a particular receiver, then the packet is dropped and no RECEIVE event is generated. However, the packet continues to be taken into account for interference and collision.

3.6. Connection manager: parameterized modeling of virtual circuit rerouting schemes

An important protocol module in Ethersim is the Connection Manager, that runs at the mobile hosts, the basestation, and mobility-aware switch nodes to allow establishment and rerouting of ATM virtual circuits. Clearly, as would be clear from the literature [1,4,9,13,17,19,22,24,27], there are a variety of possible virtual circuit rerouting schemes. Some obvious and not-so obvious ones are listed below:

1. **Extension.** Extend a VC from the old to the new basestation.

2. **Extension with Loop Removal.** In this enhancement to extension, one removes VC loops that might get formed when a mobile revisits a basestation. This requires tearing down a VC segment, and short circuiting the VC at the switch where the loop is formed.

3. **Total Rebuild.** Tear down the entire VC and establish a new one in its place.

4. **Partial Rebuild to a Fixed Anchor Switch.** Define a fixed known anchor switch through which all VCs must pass, both before and after rerouting. To reroute a VC, tear down the VC segment from the anchor switch to the old basestation, and establish a new segment from the anchor to the new basestation. As a special case of this scheme, the anchor switch may be the basestation at which the mobile host was located at the time of VC establishment.

5. **Partial Rebuild to a Dynamically Selected Cross-over Switch.** Select a cross-over switch along the path of the current VC such that a path also exists from this switch to the new basestation. Tear down the VC segment from the cross-over to the old basestation, and establish a new segment from the cross-over to the new basestation.

6. **Multicast to Neighboring Basestations.** Create all VCs as multicast VCs, and configure them such that all neighboring basestations to which a receiving mobile host may move to are part of the multicast group for that VC. When the mobile host does move, the data is already available. The multicast group is modified after the move.

These schemes need not be used alone. For example, [19] proposed a hybrid strategy whereby a fast but less optimal scheme (VC extension with optional loop removal) is used as the default, and a slow but more optimal scheme (Partial Rebuild to a Dynamically Selected Cross-over Switch) is triggered on a per VC basis by the mobile hosts based on connection-level performance measurements. Whatever be
the scheme, one must emphasize that any rerouting scheme must maintain in-sequence delivery of cells as required by ATM.

Fig. 6 shows some of these rerouting schemes.

In order to implement these rerouting protocols in Ethersim, we took a unique approach. Instead of implementing these schemes separately, we implemented a single parameterized Connection Manager module. This is based on an observation from our related research [20] that a common structure underlies all uni-cast rerouting schemes. Specifically, in [20] it was observed that all uni-cast virtual circuit rerouting schemes can be viewed as being composed of the following three steps in context of Fig. 7:

1. Selection of a suitable rebuild switch (or a base station) SW_{rebuild}, which may be selected based either on a static topological criterion (e.g. the ith switch from one end of the current VC, or a fixed known switch through which VC must pass), or on a dynamic criterion (to optimize some metric).

2. Establishment of a new VC segment from the rebuild switch SW_{rebuild} to the new basestation BS[0].

3. Tear-down of the old VC segment from the rebuild switch SW_{rebuild} to the old basestation BS[−1].

To illustrate their composition in terms of the three primitive operations, let us consider the various schemes when BS[−1] is the old basestation, and BS[0] is the new basestation. The extension scheme

---

Fig. 6. Some schemes for rerouting of virtual circuits.
fixed process, and allows the following aspects to be customized to create a unique instance of a rerouting protocol:
1. Policy for selecting SW_rebuild
   - SW_rebuild = BS[−1] (extension)
   - SW_rebuild = BS_peer (total rebuild)
   - SW_rebuild = intermediate node (partial rebuild)
   - SW_rebuild: fixed vs. ‘oracle’ vs. backtracking

2. Packets arriving at SW_rebuild from BS_peer
discard vs. buffer

3. Packets on SW_rebuild ↔ BS[−1] segment
discard vs. salvage and deliver in order

4. Removal of loops at basestations and switches

4. Case studies

In this section we will demonstrate the utility of Ethersim via two case studies. These case studies are designed to examine how wireless access and host mobility affect the achievable user and network performance, as measured by the throughput, per-packet delays and link utilization levels. In particular, we focus on the interplay between the congestion control and error recovery policies in the transport layer, the connection rerouting policies in the network layer, and the wireless radio link characteristics.

4.1. Effect of frequency hopping on performance

In the US, the 2.4 GHz industrial, scientific, and medical (ISM) band, which has 83 available 1 MHz wide slots, is commonly used for a wide variety of wireless applications. Legal requirements specify that the maximum period of transmission in any one slot is limited to 0.4 seconds every 30 seconds, for radios operating in this band. In addition radios need to cycle through at least 75 of the 83 slots in this band using a pseudo-random frequency hopping policy to pick the frequencies visited. This pseudo-random hop policy precludes any synchronization between radios operating in physically adjacent or overlapping areas thereby increasing the possibility of interference and data corruption between two or more radios that attempt to simultaneously transmit on the same or adjacent bands.
The likelihood of collision may be reduced by allocating a frequency hop sequence, i.e., the list of frequency slots visited, to each transmitter such that all transmitters in a physically adjacent area use weakly orthogonal hop sequences. Each sequence in a family of weakly orthogonal sequences is a list of N frequency slots (75 ≤ N ≤ 83) that a transmitter cycles through such that the maximum number of collisions with any other sequence in that family is bounded, for any phase of relationship between the two sequences. For example, if the minimum distance between contiguous hops is 6 slots, it is possible to generate 3 families, with 22 hop sequences in each family, such that Sequence i and Sequence j, within a family, are guaranteed to have at most one exact collision. Therefore when there are several transmitters active, the best case for a particular transmitter is when it collides with all of the other transmitters on the same frequency in its hop sequence. Similarly, the worst case occurs when it collides with each of the other transmitters on separate frequency slots.

It is possible to derive an average case collision probability via Monte Carlo simulations in each run of which every transmitter is assigned a random initial phase and all transmitters are assumed to have perfectly synchronized hop times. Table 1 shows the best, average and worst case collision probabilities as the number of transmitters is increased up to the maximum number possible in the family. When all 22 transmitters are active, there are collisions in 23.47 out of the 79 slots in the worst case. This collision probability results in a loss of almost 30% of the channel capacity due to the frequency hop collisions, for applications generating traffic at a constant rate that equals the channel capacity.

The reduction in application level throughput as a result of frequency hop collisions can be much greater for protocols that use transport protocols such as TCP to ensure reliable data delivery. Since these protocols typically use some form of window flow control to limit the maximum number of unacknowledged packets, the loss of packets due to collisions can cause the throughput to reduce until the lost packets are retransmitted. Moreover, the loss of acknowledgment packets flowing in the reverse direction can also adversely affect throughput even if the data packets are successfully transmitted over the air. Finally, the congestion control mechanism in TCP causes a reduction in the window size when there is packet loss — this also leads to a reduction in the average throughput.

<table>
<thead>
<tr>
<th># Transmitters</th>
<th>Time in collision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best</td>
</tr>
<tr>
<td>2</td>
<td>1.27</td>
</tr>
<tr>
<td>3</td>
<td>1.27</td>
</tr>
<tr>
<td>4</td>
<td>1.27</td>
</tr>
<tr>
<td>5</td>
<td>1.27</td>
</tr>
<tr>
<td>6</td>
<td>1.27</td>
</tr>
<tr>
<td>7</td>
<td>1.27</td>
</tr>
<tr>
<td>8</td>
<td>1.27</td>
</tr>
<tr>
<td>9</td>
<td>1.27</td>
</tr>
<tr>
<td>10</td>
<td>1.27</td>
</tr>
<tr>
<td>11</td>
<td>1.27</td>
</tr>
<tr>
<td>12</td>
<td>1.27</td>
</tr>
<tr>
<td>13</td>
<td>1.27</td>
</tr>
<tr>
<td>14</td>
<td>1.27</td>
</tr>
<tr>
<td>15</td>
<td>1.27</td>
</tr>
<tr>
<td>16</td>
<td>1.27</td>
</tr>
<tr>
<td>17</td>
<td>1.27</td>
</tr>
<tr>
<td>18</td>
<td>1.27</td>
</tr>
<tr>
<td>19</td>
<td>1.27</td>
</tr>
<tr>
<td>20</td>
<td>1.27</td>
</tr>
<tr>
<td>21</td>
<td>1.27</td>
</tr>
<tr>
<td>22</td>
<td>1.27</td>
</tr>
</tbody>
</table>

1 The term phase denotes the position of a transmitter in its hop sequence.
2 We use exact collision to describe the interference that happens when two sequences hop such that they are both in the same frequency slot at the time. In real life, radio often cause emissions into neighboring frequency slots because filter are never sharp. Therefore radios using different sequences may interfere even when they have hopped to adjacent or close by frequency slots. We shall refer to such interference as approximate collisions.
frequency slot is assumed to be 1 Mb/s, in the absence of collisions. An application is modeled as an infinite data source with the rate of transmission being controlled by the TCP protocol. Each TCP packet is of size 500 bytes and the TCP window size is 32 Kbytes (64 packets). The TCP timeout interval is 500 ms and the default frequency hop interval is 50 ms. Each TCP connection generates a duplex data stream with data and acknowledgments flowing in opposite directions and hence two separate hop sequences are assigned for each TCP connection: one for the basestation transmitter forwarding data packets and one for the mobile host transmitter forwarding ack packets.

Fig. 9 plots the throughput seen by the TCP connection labeled TCP Src #1, when it is the only active connection among the 22 connections shown in Fig. 8. For this scenario, the effect of the frequency hop collisions is to cause the data and ack packets for this connection to interfere with each other. During the hop interval of 50 ms, a maximum of 12 data packets can be sent out over the air by the BS. Since an ack packet is generated only when a data packet is successfully received, each alternate data packet and all the ack packets are lost – for a total of 6 data packets and 6 ack packets. Existing TCP implementations, which use cumulative acknowledgment policies, can only recover from multiple packet losses via a coarse-grained timeout, typi-

![Fig. 9. Oscillation in TCP throughput due to frequency collision (with only TCP Src #1 active).](image-url)
cally 500 ms. The timeout-based recovery causes the TCP transmitter to idle for almost the entire length of the timeout interval, with the throughput dropping to near-zero. The packet losses also trigger the congestion control algorithm to reset the TCP window size to 1 packet—500 Kbytes. Subsequently, the TCP connection takes a few seconds to build its window size up to a high enough value to fully utilize the 1 Mb/s channel. Since the hop interval is 50 ms, the collision interval is 4 seconds and hence by the time the TCP connection builds up its throughput, the next collision causes the throughput to drop again. Fig. 9 illustrates the resultant oscillation in the TCP throughput over time. This dynamic behavior results in the TCP connection being able to utilize only about 50% of the available channel capacity (515 Kbits/sec), even though the collision probability is only 1.27%.

In the next set of experiments, all 22 TCP connections (Src #1–Src #22) are active. If all 22 radio channels, used to transport the traffic for each of these connections, have the best-case phase relationships, then each connection loses data for 50 ms every 4 seconds, as in the previous set of experiments. However, in these experiments each active radio transmitter sends out both data and ack packets. As a result, the number of data packets lost during a collision interval is slightly higher than before because of the greater volume of traffic on each channel. Consequently the average throughput for Src #1 (490 Kbits/sec) is slightly lower than in the previous set of experiments. On the other hand, when all the transmitters have the worst-case phase relationships, each connection loses data for 21 50 ms intervals every 4 seconds. Due to the more frequent collisions, a connection is unable to ever increase its transmission rate to match the full channel capacity of 1 Mb/s and is able to utilize only about 12.9% of the average channel capacity (129 Kbits/sec). This is illustrated in Fig. 10.

One of the design parameters that controls the effect of the frequency collision on the TCP throughput is the duration of the frequency hop interval vis-à-vis the packet size. As the duration of the frequency hop interval becomes greater, more packets are dropped during each collision interval but the frequency of collisions decreases. The first effect causes transport protocols such as TCP which do not use selective acknowledgments to take longer to detect and retransmit packets resulting in a loss of throughput. On the other hand the lower frequency of collisions causes the TCP congestion control algorithm and the subsequent window size reduction to be triggered less often, leading to an increase in throughput. Further, since a smart MAC protocol will only transmit a packet if it is assured that it can transmit the entire packet before it is time to hop frequencies, a smaller fraction of the link capacity is
wasted when the frequency hop interval is large relative to the size of a packet. Fig. 11 which plots the TCP throughput for Configuration 2 for frequency hop intervals values of 10 ms and 50 ms, illustrates these different effects.

4.2. Effect of host mobility on performance

Networks that provide wireless access typically also support host mobility by providing support for host lookup and data rerouting as a host moves around. In networks using a cellular architecture, a key issue is the degree of degradation in user-level performance as a mobile host moves from one cell to another. Since mobile computing is still in its infancy, it is hard to precisely characterize the performance requirements of applications while a host is in motion – e.g., is it realistic to assume that a person can interact with his computer while walking down a hallway? However, it would appear that for some applications, such as voice telephony, there is a clear need to minimize the disruption time caused by a handoff. For other applications, such as file transfers, it appears to be desirable to minimize the reduction in throughput caused by a handoff.

The degradation in user-perceived performance depends on the duration of the handoff interval. This in turn depends on the time taken by the MAC layer protocols at a mobile host to discover and register at a new basestation, and the time taken by the network layer protocols to establish a new path through the network. For a virtual circuit oriented network, this involves modifying or rebuilding existing virtual circuits. Additionally, the (reliable) transport layer protocols operating end to end may create additional delays by attempting to discover and retransmit any packets lost during handoff. Ethersim allows us to study these effects, as the policies and parameters used in the MAC, network and transport layer protocols are varied.

The simulation configuration used to study the effect of host mobility is shown in Fig. 12 and models an office corridor type layout with basestations laid out at regular distances. The 8 basestations comprise two groups with 4 members, each of which is assumed to be connected directly by a switch. The two groups are in turn connected by another switch. As with the previous group of simulations, mobile hosts are assumed to be sending or receiving data to/from a host across a wide-area-network.

Two traffic sources are modeled. A file transfer type application is modeled as an infinite data source with the rate of transmission being controlled by the TCP protocol. A audio/video constant bit rate (CBR) type application is modeled as a source which injects data periodically with an average rate of 900 kb/s, thereby utilizing about 90% of the capacity of the last hop wireless link. In all of the experiments a
mobile hosts moves from Cell 1 to Cell 8 over the duration of the simulation, with an average stay of 2 seconds in each cell. In these simulations, the transmission frequencies are statically picked and the frequency hopping is disabled to avoid any data loss due to frequency collisions.

Our first set of experiments compares the performance of two rerouting policies — Extend and Rebuild. In the Extend policy, an existing virtual circuit is extended from the old basestation to the new basestation following a handoff. In the Rebuild policy, an existing virtual circuit is completely torn down and a new virtual circuit constructed from the source to the destination. The Extend policy is meant to minimize the handoff time since it only involves signaling and virtual circuit setup between adjacent basestations. On the other hand, the Rebuild policy is meant to construct a minimal cost end to end path every time a host moves. For both of the policies, we assume that the network attempts to forward all data in transmission on the old virtual circuits onto the new virtual circuits. These two policies are both very simple and are meant to illustrate the trade-offs inherent with one policy versus the other.

Fig. 13 illustrates the per-packet delay, and Fig. 14 illustrates the throughput, of a CBR source, with the Extend and Rebuild policy. With the Extend policy the time taken to complete an extension is typically less than 10 ms, except when the MH moves from Cell 4 to Cell 5. As a result the throughput is barely reduced due to the handoffs (see Fig. 14). However, because the extension causes longer
paths to form the delay increases gradually over time, with sharp jumps whenever domain boundaries are crossed. The spikes in the delay plot correspond to the delays suffered by packets which are in transmission at the time of the handoff and are buffered at switches/hosts until the virtual circuits are rebuilt. In contrast with the Rebuild policy the time taken to rebuild the virtual circuits is almost 100 ms. As a result, the packets which were in transmission at the time of the handoff incur delays of as much as 250 ms. Moreover, since we assume that in-transit packets are buffered rather than dropped, and since the source continues to inject packets at a rate of 900 kb/s during the handoff interval, a large queue builds up at the wireless link between the basestation and the host. This causes even packets transmitted after the handoff to incur large queueing delays, until the queue is flushed. Subsequently, the per-packet delays reduce to about 50 ms, which comprises mostly of the end to end propagation delay.

Fig. 15 shows the per-packet delay, and Fig. 16 shows the throughput, for the two rerouting schemes.
with TCP sources. In this case, the insights are quite different because of the interaction of the rerouting policy with the error control and congestion control mechanisms used in TCP. Fig. 16 shows that for the initial set of handoffs (S1 → S2, S2 → S3 and S3 → S4) the throughput reduces slightly immediately following the handoff, with both policies. This happens because of two reasons. First, TCP uses a window control mechanism to restrict the maximum amount of ‘unacknowledged’ data that can be outstanding at any time. At the time of the handoff, the window size is at the maximum possible value of 32 K and TCP has a full window worth of data in transit. Subsequently while the virtual circuits are being rewired, the flow of data and acknowledgments to/from the mobile host, and hence TCP is unable to send any additional data. Additionally, due to the handoffs there are packet losses – for the results shown here only a single packet is lost because packets in-transit are buffered and not dropped. The packet loss causes the congestion control mechanism of TCP to reduce the window size by 50% down to

---

**Fig. 15.** Effect of rerouting policy on per-packet delay with TCP source.

**Fig. 16.** Effect of rerouting policy on throughput with TCP source.
about 16 K. This value is not large enough to keep the wireless link fully saturated and hence there is a drop in throughput, until TCP increases its window size back to 32 kbytes.

The dynamic behavior is more complicated following the transition from S4 → S5. Since it takes somewhat longer for the REROUTE message to propagate from S4 to S5, more packets are lost over the air. The loss of multiple packets necessitates a timeout based recovery as pointed out earlier. Following the timeout, TCP reduces its window size down to a single packet. Due to this sharp reduction in the window size and the 500 ms idling the throughput reduces significantly. When the Extend policy is used the round trip delays are much longer than with the Rebuild policy (see Fig. 15). This causes the TCP connection to increases its window size more slowly over time. As a result, after the S4 → S5 transition, the throughput is lower with the Extend policy than with the Rebuild policy.

These results suggest that it may be possible to get the best of both worlds, i.e. low latency handoff and bounded end to end delays by performing extensions when handoffs are local and doing a rebuild otherwise. Moreover, the cost of doing a rebuild may be reduced by only rebuilding from a common ancestor rather than one end of a connection. Our results also suggest that for CBR traffic it may be more useful to drop packets rather than buffering them, specially if handoff intervals are large. These issues are examined in greater detail in a separate paper.

5. Conclusions

We have described Ethersim, a network simulation tool that integrates mobility and wireless links as first class citizens. Ethersim allows better modeling of mobile wireless networks than is allowed by conventional network simulators with only ad hoc support for mobility and wireless links. Ethersim addressed these problems by incorporating five special entities: an air module, a map, a mover, mobile hosts, and basestations. The air module models the physical air-interface effects such as radio signal attenuation and interference. The mover is a key central entity that moves the mobile hosts on the map, and allows for random movements, goal-directed movements, and synchronized goal-directed movements to model conference room type mobility patterns. We have also described the application of Ethersim to study the interplay of transport, rerouting due to mobility, and frequency-hopping based wireless link characteristics.

The current implementation attempts to strike a balance by abstracting the low level physical characteristics of the wireless link, which are computationally intensive to model in full detail, to a high enough level so as meaningfully study the interaction with logically complex network protocols. Therefore, instead of bit-by-bit simulation, Ethersim resorts to the higher level approach of associating a transmit frequency or power with an entire packet. Some important wireless link level phenomena, such as multipath interference, are however still not modeled in Ethersim. Also, while Ethersim includes a rich menu of protocol modules for higher network layers, the choices for lower level wireless protocols such as MAC are still limited. Properly addressing limitations such as these, without sacrificing the efficiency of high level modeling, is the subject of on-going work. Finally, we found the parametrized connection rerouting approach to the connection manager protocol module to be elegant and powerful, considerably simplifying the modeling of the protocols for ATM mobility.

References


Prathima Agrawal is head of the networked computing technology department at AT&T Labs in Whippany, N.J., USA. She previously worked in Bell Labs as head of the networked computing research department of Murray Hill, N.J. Dr. Agrawal received her B.E. and M.E. degrees in electrical communication engineering from the Indian Institute of Science, Bangalore, India, and a Ph.D. degree in electrical engineering from the University of Southern California. Her research interests are computer networks, mobile computing, parallel processing and VLSI CAD. Dr. Agrawal is a Fellow of the IEEE.

Gia Nguyen is a Ph.D. student in Electrical Engineering and Computer Science at the University of California at Berkeley. His research interests include wireless networking and mobile computing. He received a Bachelor of Science in Computer Science and Engineering from the University of California at Los Angeles in 1994, and a Master of Science in Computer Science at the University of California at Berkeley in 1996.