SCALABLE URBAN NETWORK SIMULATION (SUNS)

David Rhodes, Benjamin Epstein
OpCoast
Brick, NJ
{ dave, ben }@opcoast.com

and

Barry Perlman
US Army CERDEC
Fort Monmouth, NJ
Barry.Perlman@us.army.mil

ABSTRACT

A clear and accurate understanding of communication network performance is essential, especially where all layers of the protocol stack – from physical propagation to application transport protocols – must be accounted for. While some solutions exist for RF communications over 'open-terrain' areas, the urban environment is particularly challenging. To meet this challenge, a server-based, real-time solution for assessing complex communication and network effects in urban environments has been developed. By combining advanced RF ray-tracing propagation modeling and a full network simulator, the Scalable Urban Network Simulation (SUNS) software is able to provide accurate communications effects services. SUNS supports tactical modeling and simulation by predicting electromagnetic signal coverage and path loss through application of a full 3D physical knowledge of the urban environment and terrain features. The commercial OPNET network simulator is used as the core of the system, with SUNS operations supported through a custom OPNET model that functions as a modeling and simulation server. The server, in turn, communicates with other node models within the OPNET simulation. In the approach, urban propagation data are pre-computed using high-performance computing (HPC) resources utilizing ray traced models of the urban scene.

INTRODUCTION AND BACKGROUND

Assessing the performance of wireless communication networks, including impact at higher tactical levels (e.g., military / force level), represents an ongoing challenge in communications systems modeling. We can view the communication system in accordance with the protocol layers inherent in layered protocol design. At the lower layers, physical effects such as antenna performance and RF propagation are of paramount importance along with media access control (MAC). Internet protocol (IP) and routing generally form the next layers, while transport protocols such as UDP and TCP occupy the next. Above these layers reside applications which can operate autonomously or act as directed by users. In tactical environments, these applications operate as driven by the demands of users or of tactical applications.

We can see that the multilayer communications stack forms an environment wherein each 'layer' not only utilizes the layers above and below, but also wherein each layer must dynamically react to those around it. For example, if a communication fails at the transport or tactical operational level, it might be retried; packet errors or drops may cause other layers such as routing or transport to react in various ways. While 'cross-layer' design [1,2] attempts to optimize performance by easing the strict separation of protocol layers, networked communications, and its simulation, still primarily operate as a discretely layered environment. Unlike prior communication effects server efforts [3-5], the SUNS system is focused on the urban environment. This provides a particularly challenging problem.

A comprehensive approach to military operational assessments therefore requires communications modeling that takes on a holistic approach to account for the full interaction of all communications layers – including high level tactical layers as well. However, the computational aspects of these layers are much different. Antenna modeling and propagation rely on lengthy physical domain electromagnetic (EM) calculations; the network and other layers generally make use of discrete event simulation. Yet while both the network and upper layers make use of discrete event simulation, they are each large independent programs. In our case, the primary client that operates at the upper layer is the OneSAF system [6]. This mix of modeling environments naturally leads to use of a client/server software architecture as attempting to combine these tools into a single executable process would be difficult and the result unmanageable.

ARCHITECTURE

The basic architecture of the SUNS server solution is shown in Figure 1. A custom model is built into the network simulator to provide the server interface. This modeling component communicates with each node within the network simulation to enable/disable, relocate and initiate simulated communications as directed by the client. As communications occurs, either as explicitly requested by the client or as the result of routing or media access layer operations, propagation data requests take place to determine point to point losses. These loss requests are handled

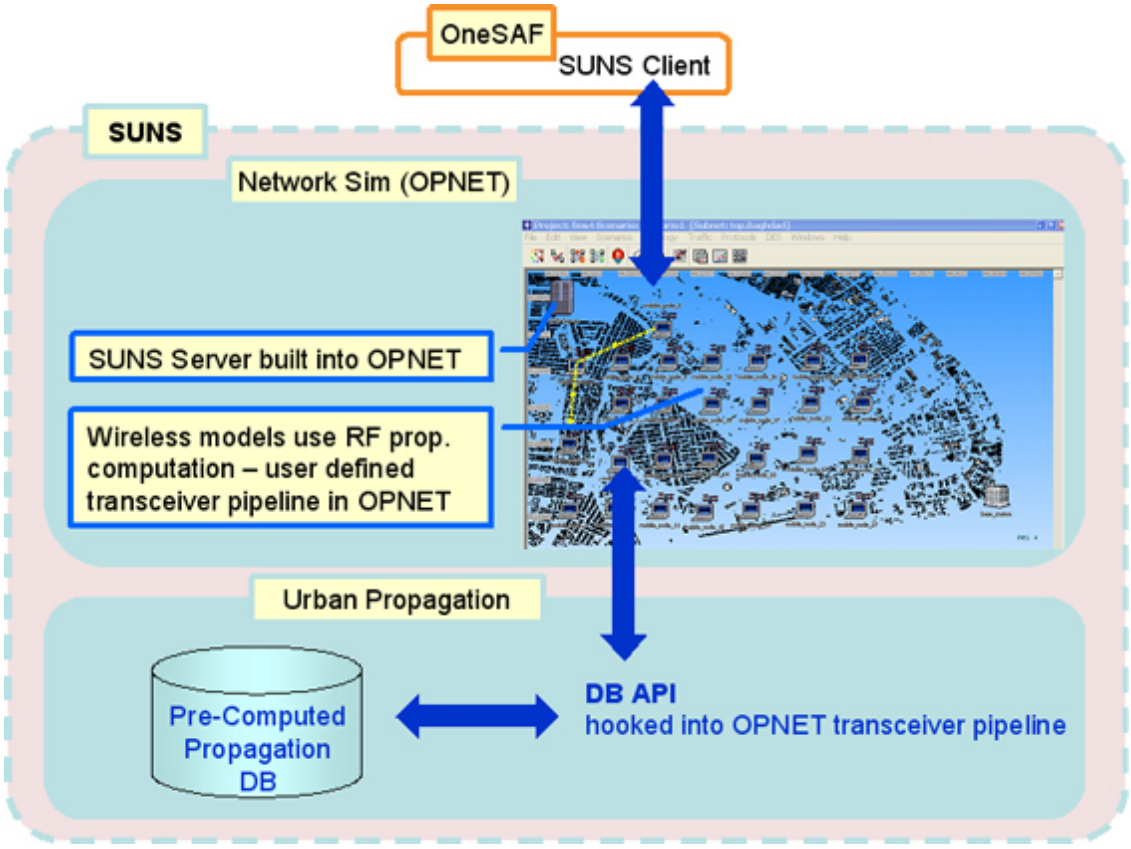

Fig. 1. SUNS Architecture.

by a modification of the simulator code that enables access to a database containing pre-computed propagation data for the urban scene under study.

The details of the OPNET [7] models and the propagation database and its computation are presented in later sections. SUNS and the clients communicate via TCP or UDP. Client/server message formats follow that as specified in the Communications Effects Server (CES) [8], with a few extensions. The messages of primary importance allow: creation and deletion of a radio node, status and location updates to a node, communication effects requests and responses.

OPNET MODELS

The primary models developed for SUNS in the OPNET framework include the SUNS server, which only exists as an interface and not as a communication node within the simulation itself, and the wireless radio node models that interact with the server and the general simulation framework. Figure 2 shows some of the node and process models used in SUNS. Finite-state based process models form the core of the modeling basis. For the states that are shown in red, the simulation stops within the state executing model C++ code before the stop and the simulation continues whenever an 'interrupt' is received wherein additional code is executed within the state. Code is executed within states shown in green as well, but the simulation does not stop within the state. In the server process, only 'self-interrupts' and 'remote-interrupts' are used in our modeling approach as will be explained later.

The server process is composed of an initialization state, followed by an idle state. In the initialization state, the server initializes an external socket for client communications and discovers and disables all mobile nodes within the scenario. The server then sends a self interrupt which will later trigger moving to the idle state. Prior to reaching the simulation stop point in the idle state, a check is made for any incoming messages from SUNS clients. If no incoming messages are received, a self interrupt is sent after 0.1 second. This will cause the simulation to follow the 'ALL_OTHER' transition back to the idle state after 0.1 second which then rechecks for incoming client messages. In this way, incoming client requests are checked ten times per second (the simulation runs in real time with clients, not at maximum simulation speed). The server also checks for previously sent messages that were never received (ex-

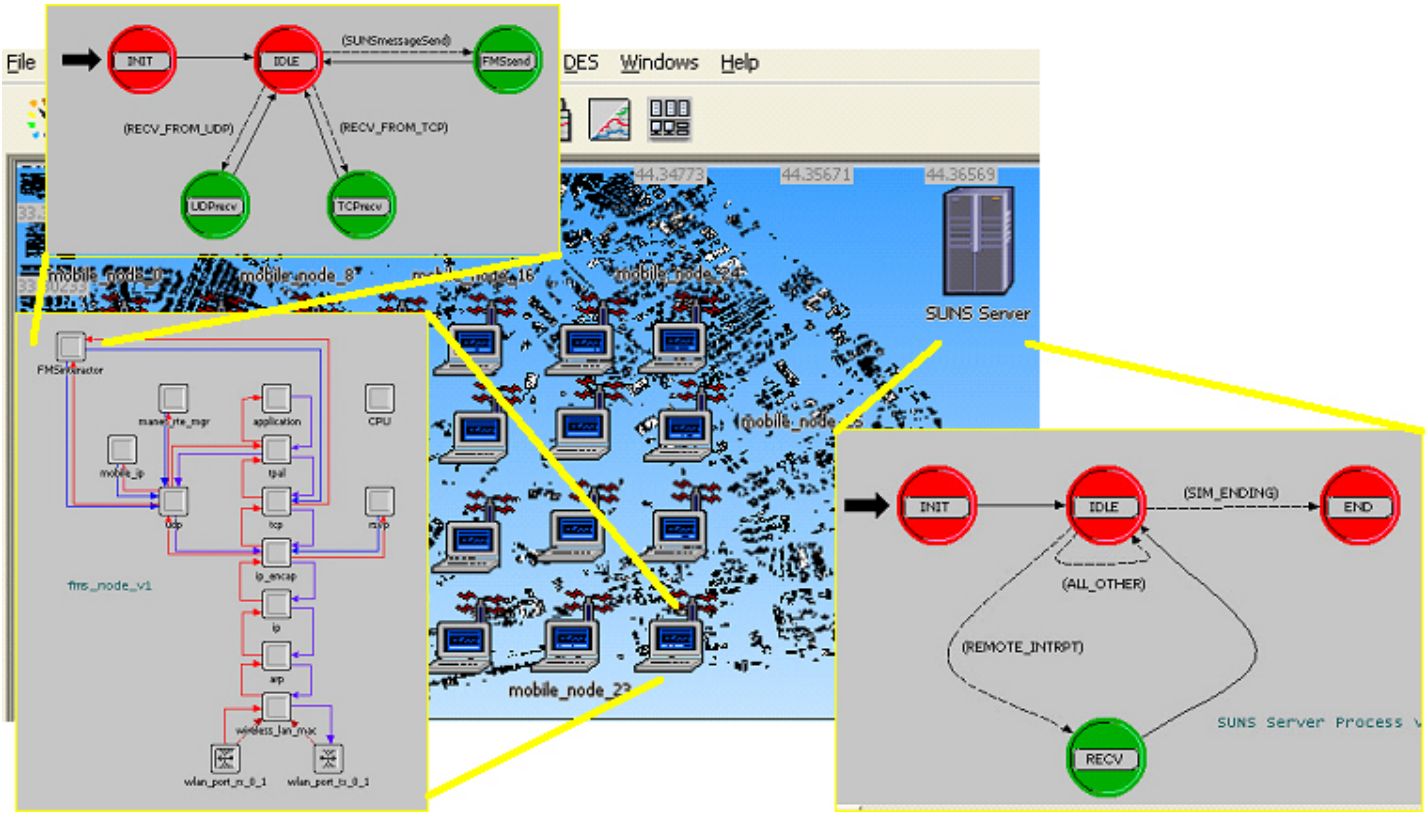

Fig. 2. OPNET models.

plained later). During this loop period, however, a remote-interrupt might be received which can only indicate that one of the mobile nodes in the simulation has completed a message reception. This causes a transition to the RECV state, which returns a communication-effects-response message to the SUNS client.

Among others, messages that the server may receive from a client include those to 'create' a platform, update/relocate a platform, and to initiate a communication-effects-request between two nodes. Upon receipt of a create platform type message, the SUNS server activates a mobile node in the simulation while keeping track of the mapping between the client and the simulator's internal nodal identification. During the message processing, nodes are positioned according to the location specified by the client using GCC coordinates. A communication effects request message from a client will cause the SUNS server to send a remote interrupt to the simulated node that is defined as the sender of the message.

Figure 2 also shows the disposition of the mobile nodes within a simple scenario and expansions of their internal processes. The mobile node model is an extension of the OPNET built-in mobile node model that adds the FMS_interactor process which, in turn, interfaces to other node modules at the UDP or TCP level. The FMS_interactor process is shown in the upper left hand corner of the figure. During initialization, the FMS_interactor process issues 'UDP listen to port 5000' (arbitrary number) commands to the UDP layer[1] so that the model will send incoming packets on port 5000 up to the FMS_interactor process via a 'stream interrupt'. Following initialization in the init state, the model enters an idle state which just awaits simulation interrupts. If a remote interrupt is received, the model transitions to the FMS_send state. By design, the only remote interrupt that this model will receive is from the SUNS server model for initiation of a communication effects request. The model uses the mapping of client naming for the receiver to determine the model name and IP number for the receiver and sends a simulated UDP message of the size requested by the client to the model's UDP layer.

This simulated message may or may not eventually arrive at the receiver, but if it does it is sent to the FMS_interactor process as a stream interrupt. The FMS_interactor process, upon receipt of this stream interrupt, transitions to its UDPrecv state, which then sends the

[1] Only simulated UDP is used at present. TCP operation, which requires additional setup and teardown, is not explained here.

remote interrupt to the SUNS server for processing and eventual return of a successful communications effects response. However, what if the message transmission between communications nodes fails? In the idle state of the SUNS server, a check is made for previously sent messages that never where received; if such a message exists for over 2 seconds the server concludes that this message will never arrive and returns a failed communication effects response message to the client. Note that this 'time-out' period would have to be adjusted upward for very long messages. Also note that during the simulated message sending that all routing protocol actions, IP level operations (such as packet fragmentation), and media access layer actions are all occurring simultaneously. All RF propagation loss requests from OPNET utilize the pre-computed propagation values (described next).

URBAN PROPAGATION DATABASE

The primary scenario of interest includes a very dense urban area approximately 6 x 3 km. At the radio communication frequency of interest, ray-tracing [9] is warranted as the proper EM approach. Although any tool can be used to provide data for the propagation database, EMAG Technology's EMLounge [10] is the tool applied in urban ray tracing. This tool provides several features conducive to achieving necessary results such as the separation of user interface and solver and the solver's ability to use high performance parallel computers. To help understand the data, Figure 3 shows a graphical result of a propagation computation (various buildings are shown blue in color). A transmitter is located near the center of this small scene and propagation losses to receivers over the area are shown.

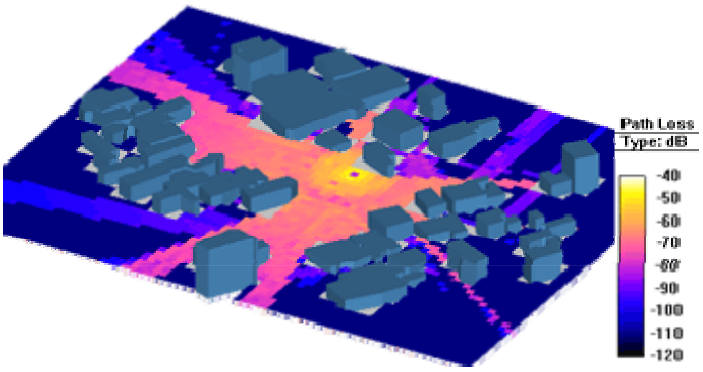

Fig. 3. Ray traced Results

As propagation computations in even subsets of this complex environment require hours of compute time, propagation losses must be pre-computed. The physical complexity of the area does not permit analysis of the entire scene. These facts give rise to resolving several interrelated issues which will now be described.

The most straightforward pre-computation would place potential transmitter (Tx) and receiver (Rx) sites in a closely spaced grid over the entire space and store the results in a database for later access. However, the scale of the scenario does not allow this approach. Our solution is to use potentially overlapping, variable-sized 'zones' within the scenario. The data for each zone is stored in a 'result table' while a single 'directory table' provides organization of multiple result tables. The results tables would provide propagation data, as computed by ray tracing, over short distances. Longer distance data relies on propagation loss computations through simple empirical models [11-13]. Use of empirical models for long urban distances is due to limitations in being able to analyze the very large numbers of buildings in the intervening and surrounding area Moreover, as the zones do not have to be regularly spaced nor are confined to a grid and are in separate tables, updating the data with more refined results over time is simpler. For example as computational power and memory capacity increases, analysis over larger zonal sizes will be possible; we can then remove the smaller zone tables and insert zones of larger size. No changes in the network level model are required to provide this enhancement.

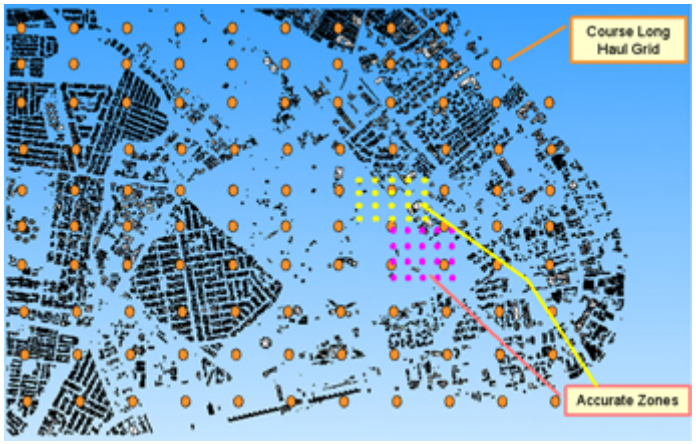

Fig. 4. Zone Example

Figure 4 illustrates the zone concept. The set of larger orange points that span the overall area is a zone, two other zones at closer spatial resolution are also shown. In this example, the directory table would have two entries referencing two result tables for each of these zones. In general, the zone is a set of Tx/Rx locations within a rectangular area (its actually a 3D box since elevation is also included). In the application of the RF analysis tool, the

Tx/Rx sites are regularly spaced, but there is no such requirement in the database design. A 'level of accuracy' (LOA) is also associated with each result table, and in this case the course zone would have a lower proclaimed LOA.

The directory table, called `prop_directory` in the database, has the following fields (ignoring elevation for the moment):

id tool version model loa resolution
freq polarization table_name
lat1 lat2 lon1 lon2

which provides the database id key, the tool used to create the data, the version of the tool, the model used within the tool, the LOA, the basic spatial resolution of the data, the frequency in MHz, the antenna polarization, the name of the table (this is the 'pointer' to the result table), the range of the zone in GCC lat/long units, respectively. Each result table, again ignoring elevation, has the following fields:

id rlat rlon tlat tlon loss

which gives the primary id key, the receiver location (lat/long), the transmitter location, and the propagation loss value, respectively.

During simulation, the propagation request attempts to retrieve the 'closest' match from the database. This match, however, is a function of the following parameters: frequency, Rx and Tx location, table resolution, and the level of accuracy. Our current search approach is to locate the propagation data table that: (i) includes the Rx and Tx location in the zone (this is basically not negotiable), (ii) that has a frequency match within 1 KHz, and (iii) among the possible tables, use level of accuracy as a primary sorting value and resolution as a second. The steps needed to extract the best propagation loss value now follow.

The first step is to consult the directory table (`prop_directory`) to find which tables include the zone of interest covering the Rx/Tx points. Both of these must be within the bounds indicated. This step may return more than one table as zones can overlap or be subsets of each other. Furthermore, each potential table cold reflect different levels of accuracy and resolutions. For Rx and Tx points of 22.1/33.2 and 22.0/33.1 (lat/long) at a frequency of 2.4 GHz, the following SQL statement works to select the result table with the best LOA and secondarily resolution for a zone within a frequency match of 1 KHz (0.001 MHz):

```
SELECT id,table_name FROM prop_directory
 WHERE (22.1 BETWEEN lat1 AND lat2)
   AND (33.2 BETWEEN lon1 AND lon2)
   AND (22.0 BETWEEN lat1 AND lat2)
   AND (33.1 BETWEEN lon1 AND lon2)
   AND (ABS(2400-freq) < 0.001)
 ORDER BY loa DESC, resolution LIMIT 1
```

This will choose the table that covers the Rx/Tx points with the highest LOA (from the `ORDER BY loa DESC,resolution LIMIT 1` phrase) with a secondary search criteria of lowest resolution; note reversing these clauses would result in finding the table with the highest resolution first and then secondarily using LOA. If there are applicable tables with equal LOA and resolution then an arbitrary one is chosen by this statement.

After this statement is used to select a propagation data table, say called `prop_v_900_2`, it is then queried to get the loss for the closest match to the Rx/Tx points as the sum of the squares of differences (`metricvalue` in the SQL statement):

```
SELECT id,loss,
  (ABS((rlon-33.2)*(rlon-33.2)  +
       (rlat-22.1)*(rlat-22.1)) +
   ABS((tlon-33.1)*(tlon-33.1)  +
       (tlat-22.0)*(tlat-22.0))) AS metricvalue
  FROM prop_v_900_2 ORDER BY metricvalue LIMIT 1
```

Note that this approach doesn't allow for finding an alternate table which might have better point matches (resolution) but not as good LOA. Other search tradeoffs such as softening the frequency match to find better resolution or LOA matches can be performed with alternate queries One way to try these tradeoffs is to eliminate the 'LIMIT 1' phrase in the initial search, or change the 1 to a higher value, and then search each of the propagation data tables returned. Also, right now we are only putting data with vertical polarization into the database, so this is not a search criteria; but of course the database design properly supports polarization parameters.

CONCLUDING REMARKS

The primary models developed for SUNS in the OPNET framework include the SUNS server and the wireless radio node models that interact with the server and the general simulation framework. These models, combined with our urban propagation database and propagation lookup approach, create a server-based resource for accurate communication effects in complex urban environments. A particular client for the SUNS system is the One Semi-Automated Forces (OneSAF) system. The interaction of OneSAF and the SUNS Server account for real-time communication effects that include all protocol, routing, IP, MAC and propagation effects. This enhanced functionality will provide for more advanced network centric training, analysis and experimentation for urban environments as

well as provide pertinent feedback to communication network developers.

The work is sponsored by the DoD High Performance Computing Modernization Office (HPCMO) under the Office of the Secretary of Defense.

REFERENCES

[1] V. Kawadia, P.R.Kumar, "A cautionary perspective on cross-layer design," *IEEE Wireless Communications*, Feb. 2005, Vol. 12, Issue 1, pp. 3-11

[2] S. Shakkottai, T.S. Rappaport, P.C. Karlsson, "Cross-layer design for wireless networks," *IEEE Communications Magazine*, Oct. 2003, Vol. 41, Issue 10, pp. 74-80

[3] J. Dulmage, H. Tuan, J. McConnell, M. Riehl, J. Wessel, J. Peace, "Communications effects server-realistic communications effects for distributed simulations," *MILCOM 2000*, Los Angeles, CA, pp 335-339

[4] Rajive Bagrodia, "Communication Effects Server for Mobile Ad Hoc Networks," *MILCOM 2004*, Monterey, CA.

[5] Rajive Bagrodia, Ken Tang, Steve Goldman, Dilip Kumar, "An accurate, scalable communication effects server for the FCS system of systems simulation environment," *37th Winter Simulation Conference*, 2006, Monterey, CA, pp 1226-1233

[6] http://www.onesaf.net/community/

[7] www.opnet.com

[8] *Communication Effects Server (CES) 2.0 User's Guide*, Version 1, September 2005, by Scalable Network Technologies, Inc.

[9] F. Aryanfar, K. Sarabandi, M.D. Casciato, K. Sabet, "Wave propagation characterization in complex urban areas using EMTerrano," *IEEE Antennas and Propagation Society International Symposium*, Monterey, CA, 20-25 June 2004, Vol. 2, pp. 1631-1634

[10] http://www.emagware.com/emterrano.html

[11] Y. Okumura, E. Ohmori, T. Kawano, and K. Fukuda, "Field strength and its variability in VHF and UHF land-mobile radio service," *Review of the Electrical Communication Laboratory*, vol. 16, no. 9-10, pp. 825-873, 1968.

[12] M. Hata, "Empirical formula for propagation loss in land mobile radio services," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 3, pp. 317-325, 1980.

[13] J Walfisch, HL Bertoni, "A theoretical model of UHF propagation in urban environments," *IEEE Trans on Antennas and Propagation*, 1988