HARDWARE-IN-THE-LOOP SIMULATION TECHNIQUES FOR VALIDATING SOFTWARE DEFINED RADIO (SDR) SOFTWARE

Andreas Yankopolus, Peter Sholander, and Glenn Frank
Scientific Research Corporation, Atlanta, GA, U.S.A.
{ayank, psholander, gfrank}@scires.com

ABSTRACT

This paper describes a Hardware-in-the-Loop (HIL) simulation capability for wired and wireless networks that allows Internet Protocol (IP) packets, application-layer information, and Distributed Interactive Simulation (DIS) Protocol Data Units (PDUs) to be passed between real nodes on an external network and their "mirrored" virtual mobile-nodes within a Mobile Ad hoc Network (MANET) simulation. This provides a "Communications-Effects Server" capability for those real-nodes. The current applications include modeling cooperative attack by Unmanned Aerial Vehicles (UAVs), instrumented test-ranges, and networks of Unattended Ground-based Sensors (UGS). This HIL environment also functions as a software validation test-bed for Linux-based sensor-processing software and Joint Tactical Radio System (JTRS) compliant routing software.

1. INTRODUCTION

Simulation-based acquisition programs are starting to require the validation of networking protocols in Hardware-in-the-Loop simulations that contain 10's of real actors and 1000's of virtual entities. The simulation entities should run the "real code" for the network layers and applications – since this can validate the correctness of production software in large networks without requiring large-scale field exercises. This is a critical capability because "kluged-up" simulation code and the production "gold code" can produce significant performance differences. In addition, simulation code often uses shortcuts that complicate a subsequent port to target platforms such as JTRS.

Another issue is realistic modeling of the "communications effects" introduced by wireless channels and mobility. As an example, consider Information Assurance (IA) software running on UGS nodes that use a MANET-routing protocol as shown in Figure 1. In SRC's existing HIL test-bed, the Warfighter nodes are real nodes (Compaq iPAQs), while the sensor nodes are simulated entities with QualNet. All of the nodes then use QualNet as a "Communications Effects Server". A real node has a corresponding "mirrored node" within QualNet that uses QualNet's Media Access Control (MAC) and Physical (PHY) layers, in conjunction with realistic propagation and terrain models, to model that real node's wireless connectivity to the virtual nodes and other real nodes. The simulated entities use the QualNet models for IP. However, those simulated entities can use a "System Abstraction Layer" [7] to run "real" (e.g., Linux-based and JTRS-compliant) routing software and "real" application layer sensor-processing software. In this case, the simulation tool (e.g., QualNet) also functions as a "software validator" that debugs network-layer and application-layer software in large networks of real/virtual nodes while maintaining reasonable fidelity for the communications-effects (e.g., latency and loss), mobility models and operational scenarios.

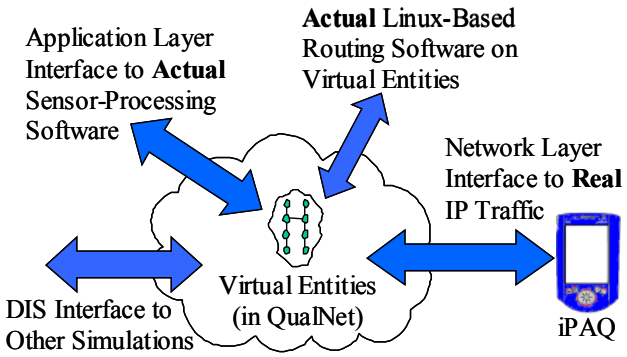

Figure 1. HIL Interfaces for Wireless Sensor Networks

Existing simulation tools have similar capabilities. For example, ns2 has both an "opaque mode" and a "protocol mode" [1]. In opaque mode, the ns2 simulator does not manipulate the "real-world" protocol fields. Live data packets may be dropped, delayed, re-ordered, or duplicated, but no protocol processing is performed. In protocol mode, the ns2 simulator is able to interpret and/or generate live network traffic containing arbitrary field assignments. References [2] and [3] describe "IP-Layer Bridges" that allow external applications to function as traffic generators for a simulated wired network. Ref. [4] describes techniques whereby a general-purpose computer can emulate a multi-port delay-error simulator. Conversely Ref. [5] describes how real systems can generate realistic network-traffic traces as an input to a hardware-based Delay-Error Simulator. Finally, the High-Level Architecture (HLA) and DIS standards allow an application-layer "HIL

Bridge" capability between the real and virtual domains. However, this past work did not fully support the application shown in Figure 1 -- with the provision of the Communications Effects Server for multiple real nodes being the key omission. This new HIL capability allows multiple real wired and wireless nodes to interact with an arbitrary number of simulated nodes. An SRC-developed System Abstraction Layer (SAL) also enables the simulated nodes to run the same routing and application-layer software-code as the real nodes.

This paper focuses on the *practical* software-design issues needed to implement the HIL capabilities shown in Figure 1. As such, it outlines how the QualNet simulation tool can be extended to provide an HIL capability that supports multiple, mobile nodes combined with a realistic Communications-Effects Server capability. The extension of these techniques to OPNET and ns2 is feasible.

As a brief overview, QualNet is an event-driven network simulation tool (www.scalable-networks.com) that is a commercial version of the GloMoSim package developed under the Defense Advanced Research Projects Agency (DARPA) Global Mobility (GLOMO) program. QualNet is designed for modeling IP traffic across Mobile Ad hoc Networks. It follows the ISO model and uses a TCP/IP stack derived from the BSD stack. A key feature of QualNet is that large simulations often run faster than real-time. For example, simulating 30 minutes of sensor-network operations (with 100 sensor nodes that use Dynamic Source Routing (DSR) [6] as their routing protocol) takes about 1 minute of "wall-time" on a single-processor 1.8 GHz Pentium PC. This feature is the basis for this paper's Communications Effects Server capability. (Note: other tools such as OPNET may also run "faster than real-time" in some scenarios.)

2. APPLICATION-LAYER HIL INTERFACES

A general-purpose HIL environment should allow application-layer information, DIS PDUs, IP packets and non-IP data packets to be passed between real nodes on an external network and virtual nodes within a simulation, as shown in Figure 1. This section outlines both the Application-Layer HIL Interface and the DIS Interface. The next two sections focus on the more difficult problem of bridging IP packets from multiple real nodes into a MANET simulation tool such as QualNet.

2.1 Application-Layer HIL Interface

For this interface, all of the network (IP) and transport layer (Transport Control Protocol (TCP) and User Datagram Protocol (UDP)) information is removed before the application-layer traffic is passed between a real node and its mirrored node within QualNet. The transport and network layer code in QualNet then regenerates this header information as the packet travels down the IP stack in QualNet. This application-layer interface is straightforward since there are no communications or mobility effects involved.

2.2 Distributed Interactive Simulation (DIS) Interface

In support of Air Force Research Lab's (AFRL's) Cooperative Attack research, a DIS interface was required to receive Position, Location and Tracking (PLT) data from a real UAV. The standard QualNet distribution lacked a built-in DIS capability. However, SRC was able to easily construct one based on VR-Link, which was written by MÄK Technologies (www.mak.com). Other vendors' DIS software packages were also available. So, this interface was straightforward.

3. HIL INTERFACE FOR IP TRAFFIC

The HIL simulation environment shown in Figure 1 requires an HIL interface between simulated IP-speakers and real IP-speakers. That interface should support both UDP and TCP traffic.

This section first presents the overall architecture for an IP-Layer HIL Bridge. It then gives details on six important implementation issues – namely: a) IP header format translation; b) "Packet Teleportation" between the HIL Bridge code and the mirrored nodes in QualNet; c) porting Linux-based routing software to run within QualNet; d) ICMP issues; e) Address Resolution Protocol (ARP) issues; and f) clock synchronization between the real and virtual domains. For simplicity, this section gives a detailed design for the case of one real stationary-node interacting with N virtual nodes. The next section then outlines a design that supports multiple real mobile-nodes.

This paper focuses on developing an HIL capability for the QualNet simulation tool. However, these techniques are applicable to other tools such as OPNET and ns2.

3.1. HIL Architecture for One Real Node

As shown in Figure 2, the real node is "mirrored" with QualNet to model propagation and mobility effects. As such, QualNet is used as a "Communications-Effects Server for the real nodes. This mirrored node in QualNet does not run applications or routing protocols within the simulation. Instead, the applications and routing protocols run on the real machine with an unmodified IP stack.

IP datagrams (containing application and routing information) are passed between the real node's Ethernet network and the QualNet simulation as shown in Figure 2. This is accomplished by using a Packet Socket to read network frames from the Ethernet network, a "HIL Bridge

to move datagrams into and out of QualNet via the Interface Node, and various QualNet applications to support "teleporting" datagrams between the Interface Node and the real node mirrored in QualNet. Writing IP packets back to the Ethernet network uses a Raw Socket in the current implementation.

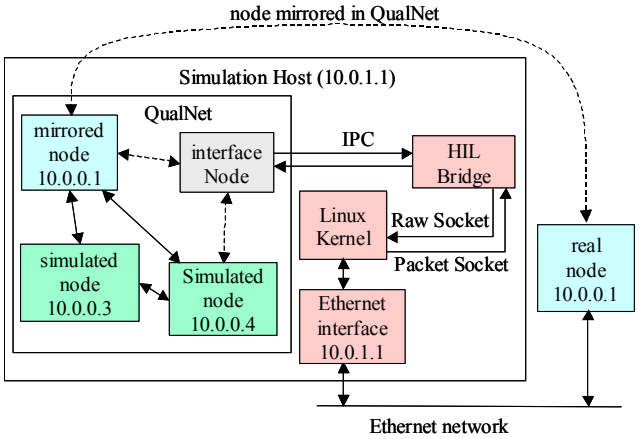

Figure 2. HIL Bridge Architecture

Packets from applications on the real node are forwarded to the simulation host where they are passed to the HIL Bridge application via Linux's Packet Socket interface. (Note: Packet Sockets preserve the Ethernet header information for use by the HIL Bridge code.)

The HIL Bridge adds a new "Message-Application header" to the incoming IP packet. The packet is then passed to the Interface Node in the QualNet simulation via an Operating System (OS) level Inter-Process Communication (IPC) mechanism. The Interface Node then "teleports" the IP packet to the real node's corresponding mirrored node. This teleport function uses a Message Application (MSG App). The mirrored node within QualNet checks the MSG App header and determines that the received packet contains an IP datagram from its corresponding real node. It reacts by pushing the complete packet down into its IP stack. The packet is then sent out the simulated MAC and PHY layers. The PHY layer of the simulated destination node eventually receives the packet (possibly after it has been forwarded by several intermediate nodes) and passes it up its IP stack.

The remainder of this paper focuses on the technical details on implementing this architecture.

3.2. IP Header Format and CRC Calculations

QualNet uses a TCP state-machine that was ported from FreeBSD. This TCP implementation provides fairly complete functionality, such as packet reordering and slow-start after the loss of an IP packet. However, IP fragmentation is currently disabled in QualNet. This is a typical simplification used within many network-simulation tools. It will only cause problems if fragmented packets are passed into the simulation.

Another difference between the IP, UDP, and TCP code in QualNet and the analogous code in a standalone computer is that the QualNet code does not perform checksum calculations on outgoing packets or check the checksums on received packets. This simplification speeds simulation execution and is perfectly legitimate because QualNet itself decides whether or not packets are received correctly at each hop. Since a standalone computer would drop a "QualNet IP packet", because it would fail the IP layer checksum test, the HIL Bridge code must translate between the header formats used within QualNet and IPv4 networks.

3.3 HIL Bridge

The HIL Bridge packet format shown in Figure 3 is used to transport application layer and IP layer traffic to and from QualNet. (Note: since this header is not used "over the wire", the fields are word-aligned for ease of coding rather than header-size efficiency.) The same header format is used for both the IP-Layer HIL Bridge and the various Application-Layer HIL Interfaces outlined previously.

Type (32-bit integer)	**Length** (32-bit integer)	**Source Node** (64-bit integer)	**Destination Node** (64-bit integer)	**Payload** (Variable length)

Figure 3. HIL Bridge Packet Format

This header, which contains *Type*, *Length*, *Source Node*, and *Destination Node* fields, is appended to the start of the actual data packet before it is sent across the IPC mechanism shown in Figure 2. The *Type* and *Length* fields are 32-bit integers while the *Source* and *Destination Node* addresses are 64-bit integers so they can handle Ethernet MAC addresses. The *Type* field indicates the type of data contained in the payload and the *Length* field the number of bytes in the payload and message-application header. Table 1 shows the currently-implemented packet types. The *Source Node* and *Destination Node* fields give the payload's source and destination addresses. For unicast IP packets, these fields will match the source and destination address of the IP datagram. For broadcast packets, they will indicate the addresses of the nodes that sent and received the packet. (Note: the source and destination node fields should always indicate a specific node address, never a broadcast address or range of addresses. Otherwise, an incoming data packet cannot be directed to the appropriate mirrored node within QualNet.)

Table 1. HIL Bridge Data Types

Type	Data	Payload
1	IP traffic	IP datagram
2	MAIS endgame	Application data
3	DIS	Application data

When functioning as an application layer bridge, the HIL Bridge code can determine application packet types based on which socket received the packet (since each one listens on a unique port). When bridging at the IP level, raw IP packets are not classified but instead sent across the bridge all together without regard to application type. When receiving data-packets back from QualNet, the HIL Bridge gets the information from the message-application header added by the Message Application on the mirrored node. This information is used to generate real TCP/UDP/IP headers, with correct checksums and the IP address of the mirrored node's corresponding real node,) before placing the data-packet back onto the real network.

3.4 Packet Teleportation

The dedicated Interface Node shown in Figure 2 resides within the QualNet simulation and receives packets from the IPC mechanism (e.g., FIFO or message queue) before "teleporting" those packets to the correct mirrored-node within QualNet. The Interface Node does not participate in the simulation in any other way. The packet teleportation, which is accomplished using the Message Application, takes place in zero simulation time, and avoids the Interface Node's protocol stack altogether. This technique is only suitable in sequential simulations because it would violate lookahead assumptions required for parallel simulations.

The mirrored node removes the Message Application header before transmitting the packet within the simulated network. The packet traverses the simulated MAC and PHY layers before arriving at its destination node. If the destination is a mirrored node that is simulating a real host outside of QualNet, the destination adds the Message Application header to the packet and teleports it back to the Interface Node. The Interface Node in turn places it on the outgoing IPC mechanism for transmission to the HIL Bridge code. The HIL Bridge code uses the Message Application header to then send the IP packet to the correct real node. In essence, this allows QualNet to act as "Communications Effects Server" for a real node that is interacting with an operational scenario within a larger virtual environment. This description focused on bridging IP packets into QualNet. However, the HIL Bridge also provides an application-layer interface for bridging application layer data where the contents of the IP and transport layer headers are not important.

3.5 Mirrored vs. Simulated nodes

A QualNet node's IP stack must determine if it is running on a mirrored or simulated node. This is accomplished by running a passive "mirror" application on all mirrored nodes. The IP stack on a QualNet node checks for the presence of the mirror application in order to determine if its node is a "mirror" of a real node. If the QualNet IP stack finds the mirror application on the node, it teleports IP packets to the Interface Node. If the QualNet IP stack does not find the mirror application, it allows them to continue up the stack of its simulated node. This approach is advantageous because it requires no changes to the existing QualNet Node data-structure, which is used throughout the existing simulation code.

3.6 Porting Linux-Based Routing Protocols to QualNet

The main technical issue with porting routing-code to the QualNet simulation package is that "real" routing code runs as different copies on multiple nodes while QualNet emulates the same routing code on multiple virtual-entities. As such, the QualNet "Node Data Structure" (which indicates which virtual entity is currently being simulated) has to be passed to many of the routing protocol's functions. As described in [7], this can be accomplished via a "System Abstraction Layer" (SAL) that allows the routing software to run on Linux, Windows, the JTRS and QualNet.

In order to call QualNet functions from within the SAL wrapper functions, the QualNet Node structure needed to be accessible. This was accomplished by encapsulating it within a "SalTask structure", which was then passed as an argument to most SAL functions. The SalTask held all "global" data for a task. By protecting this data within the SalTask structure, each task on each simulated node was provided with its own "protected" memory region within the QualNet simulation environment.

QualNet is essentially callback-based and relies on layer-specific message handlers to process messages. SRC's current SAL is also callback-based; applications provide callbacks to handle system events, such as message queues, timers expiration or data arrival. With the exception of startup code, the routing protocol's native callback functions ran unmodified. QualNet messages were simply translated to SAL events, and the routing-protocol's callbacks then executed as normal. This technique allowed SRC to use QualNet as a "software-validator" that helped

debug our *unmodified* JTRS-compliant routing software in large networks. This same SAL technique was also used to port application-layer sensor processing software from Linux and Windows CE to run within QualNet simulations. (Note: a similar tool has been reported for OPNET [8].)

3.7 Internet Control Message Protocol (ICMP)

QualNet does not simulate common ICMP messages, such as "host unreachable", "echo", and "host unreachable". Unknown ICMP messages received by a simulated node are silently discarded. Hence, ingress filtering of ICMP messages is not strictly necessary within the HIL Bridge but may improve performance in larger simulations.

3.8 Address Resolution Protocol (ARP) Issues

The QualNet simulation host computer must be configured to answer ARP requests (from real nodes) for all of the simulated and mirrored nodes. Since the real nodes (that are mirrored within QualNet) are likely to reside on the same Ethernet network as the source of the ARP requests, all real nodes other than the simulation host must be configured to only accept ARP replies from the simulation host.

This technique (along with static ARP tables at the real nodes) will support multiple real nodes with a fixed set of one-hop neighbors. However, the simulation host will answer all ARP requests made by the real node, whether or not the target of the ARP request is in radio range. That problem can be solved by writing an ARP implementation that runs on the Interface Node and determines whether or not to answer ARP requests based on the node locations in the QualNet simulation. With this improvement, the simulation tool can determine whether the ARP request's target would actually have heard the ARP request and whether the source would have successfully received the ARP reply.

Another issue is that if one real node sends an ARP request for another real node, it will get two replies: one from the real node and another from the sim host. One solution is to filter ARP requests at each node and reject those coming from other real nodes.

3.9 Clock Synchronization Between Real and Virtual Nodes

This HIL capability uses the Kansas University Real-Time (KURT) Linux (http://www.ittc.ku.edu/kurt/), which provides microsecond timing resolution and event-driven real-time scheduling. KURT-Linux decreases the timer resolution from 20-30 ms to 50-70 μs.

Within the QualNet simulation tool, the main-event loop contains a "Process Event" function. This HIL capability added a check to the Process Event function which determines whether the simulation is either ahead of or behind "wall time". If it is ahead then the HIL Bridge's IPC mechanism uses a blocking read. If it is behind then a non-blocking read is used. This simple technique maintains clock synchronization between the real and virtual domains – if the underlying simulation tool runs "faster than real time". It is essentially a low-cost implementation of the HLA RunTime Infrastructure (RTI).

4. ADDITIONAL DESIGN ISSUES FOR MULTIPLE REAL NODES

This paper has focused on the simple case of one real stationary-node and N virtual nodes. This section outlines the additional software required to support multiple real mobile-nodes within the HIL capability described by this paper. A future paper will give more technical details.

The major issue with supporting multiple real nodes is that forwarding datagrams through real nodes (in the case where a real node lies between the source and destination nodes) requires that the HIL Bridge send the packet back out (from a simulated node in QualNet) to the real node with modified Ethernet headers. Otherwise, the real node cannot determine which mirrored node sent that packet. Similarly, the HIL Bridge must also have access to the Ethernet source address of the incoming packets from real nodes. Otherwise, the HIL Bridge cannot forward that packet to the correct mirrored node in the network path.

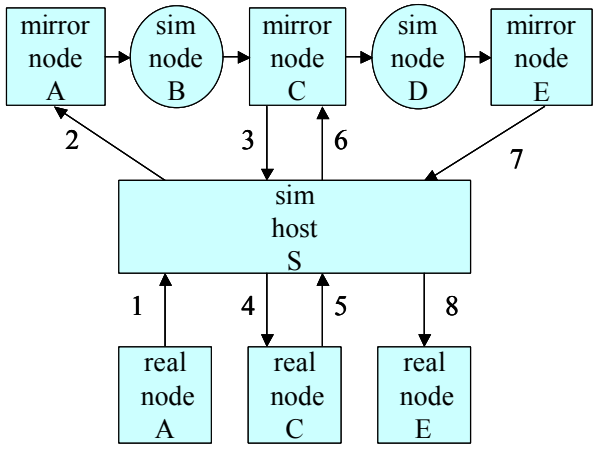

Figure 4. Multi-Node Packet Path When Multiple Real Nodes are Used in HIL Simulation

As an example, let real node A send a packet to real node E along a network path of A-B-C-D-E. Assume that packet must pass through simulated nodes B and D as well as real/mirrored node C. That packet therefore travels through the simulation host (S) four times. Figure 4 shows the path taken by packets between the real nodes on the network and the mirrored/simulated nodes within the

simulation. Real nodes appear as squares below the simulation host, mirrored nodes appear as squares above the simulation host, and simulated nodes appear as circles above the simulation host. Segments 1, 4, 5, and 8 occur "over the wire" and consist of actual Ethernet frames. Segments 2, 3, 6, and 7 occur over a Unix domain socket and consist of the IPC packets described in Section 3.3.

Table 2 shows the proper headers for packets traveling both "over the wire" between the real nodes and the simulation host, and also via the IPC mechanism between the interface application and a mirrored node. The IPC header indicates which nodes sent and received the packet. It is used to: a) create the MAC layer headers on a packet sent to a real node from the simulation host; and b) to pass packets received from a real node to the correct mirror node.

Table 2. Network and IPC headers

Path	Type	IP		MAC		IPC	
		SRC	DST	SRC	DST	SRC	DST
1	Enet	A	E	A	B	N/A	N/A
2	IPC	A	E	N/A	N/A	A	B
3	IPC	A	E	N/A	N/A	B	C
4	Enet	A	E	B	C	N/A	N/A
5	Enet	A	E	C	D	N/A	N/A
6	IPC	A	E	N/A	N/A	C	D
7	IPC	A	E	N/A	N/A	D	E
8	Enet	A	E	D	E	N/A	N/A

Enforcing this sequence of source/destination MAC address within the Ethernet frames requires either:

- A custom MAC-layer driver to place IP packets back out on the wire via the Raw Socket interface. This driver would also drop ARP packets from other real nodes on the same Ethernet segment.
- Using an Ethernet Switch and/or multi-port Ethernet NIC to place each real host into its own Ethernet segment. This approach requires a separate Ethernet port for each real node, but it does not require "spoofing" Ethernet source addresses.

The latter approach is easier to implement, while the former approach allows greater scalability in the number of real nodes in the HIL test-bed. The packet socket interface currently used to receive packets discards the MAC layer information upon their reception. This information is critical when the simulation host is communicating with multiple real nodes as it provides a way to determine which node actually sent the packet. As Table 2 shows, the IP source address is not a reliable indicator of which node actually sent a packet. Finally, the MAC layer driver must also drop ARP packets received from other real nodes in order to maintain consistency between each node's routing table and ARP cache.

5. CONCLUSIONS AND FUTURE WORK

Hardware-in-the-Loop (HIL) simulations allow system designers and integrators to combine the reality of operational hardware and software with the inexpensive scalability of software simulations. This paper gave implementation details for a HIL test-bed (for wireless and wired networks) that supports one real node interacting with *N* virtual entities within a MANET application scenario. It also outlined the extension of that HIL architecture to support multiple real nodes. This HIL test-bed provides mechanisms for running the same routing and application layer software on both real and simulated nodes. It also allows the simulation tool to act as a "Communications Effects Server" for the real nodes. As such, this HIL capability can provide a software validation test-bed for networking software intended for SDR applications.

REFERENCES

[1] "Network Emulation with the NS Simulator", http://www.isi.edu/nsnam/ns/ns-emulation.html

[2] Luigi Rizzo, "Dummynet: a simple approach to the evaluation of network protocols", ACM Computer Communication Review, January 1997.

[3] Qiang Gu, Alan Marshall, "The Design of A Software Bridge between Real and Simulated Computer Networks", WMC01, Arizona, January 2001

[4] "NIST Net Home Page", http://dns.antd.nist.gov/itg/nistnet/

[5] B. Mah, P. Sholander, L. Martinez and L. Tolendino, "IPB: An Internet Protocol Benchmark Using Simulated Traffic", IEEE MASCOTS '98, July 1998.

[6] David Johnson, et al, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", draft-ietf-manet-dsr-08.txt, Feb. 24, 2003.

[7] P. Sholander, P. Coccoli, T. Oakes and S. Swank, "A Portable Software Implementation of a Hybrid MANET Routing Protocol", SDR Forum's 2002 Technical Conference, November 2002.

[8] Ram Ramanathan,, "Summary of DARPA/ATO FCS Communications Program on Utilizing Directional Antennas for Ad Hoc Networking (UDAAN), 2001, http://www.ir.bbn.com/projects/udaan/udaan-index.html