A PORTABLE SOFTWARE IMPLEMENTATION OF A HYBRID MANET ROUTING PROTOCOL

Peter Sholander, Paul Coccoli, Tracey Oakes and Sean Swank
Scientific Research Corporation, Atlanta, GA, U.S.A.
{psholander, pcoccoli, toakes, sswank}@scires.com

ABSTRACT

This paper describes a portable software-implementation of a hybrid-routing protocol that uses a flexible mix of proactive and reactive routing techniques within Mobile Ad hoc Networks (MANETs) such as Military tactical networks. After a review of the benefits/applications of proactive, reactive (on-demand) and hybrid routing, the architectural details and protocol operation of SRC's MobileRoute™ software are presented. Experimental results for that software's processor utilization on a 650 Mhz Pentium III processor are given. The MobileRoute software was originally developed on Red Hat Linux. This paper outlines the procedures that were used to port that routing software to Windows CE and the QualNet network-simulation tool.

1. INTRODUCTION

Current Internet routing protocols such as Open Shortest Path First (OSPF) were developed for the fixed Internet backbone where routers are bolted to the ground. It is well-known that OPSF fails in mobile wireless environments because it attempts to continuously track every change in the network topology. In wireless environments, OSPF's "proactive" approach to routing causes wasted overhead that often saturates the wireless medium with control traffic for routes that are never used. As such, there has been extensive research on new routing protocols for the "Mobile Ad hoc Networks (MANETs)" projected for next-generation military networks.

MANETs differ from wired Internet Protocol (IP) networks in several respects. Ad hoc networks lack the centralized infrastructure found in both cellular and fixed networks. Nodes and infrastructure may be highly mobile. Second, there is a blurring of IP's typical distinction between routers and hosts. Third, most military MANETs have low bandwidth (kbps) wireless links and battery-operated nodes that require power-efficient operation.

Existing IP routing protocols can be classified as either "proactive" or "reactive". Proactive protocols attempt to continuously evaluate all of the routes within a network – so that when a packet needs to be forwarded, a route is already known and can be used immediately. OSPF is an example of a Proactive Routing Protocol (PRP) for wired IP backbone networks. MANET-specific examples include Optimized Link State Routing (OLSR) [1] and Topology Broadcast based on Reverse Path Forwarding (TBRPF) [2]. In contrast, Reactive Routing Protocols (RRPs) invoke a route determination procedure "on-demand" only. Thus, if a route is needed then some sort of a global-search procedure is employed. The classical flood-search algorithms are simple reactive-protocols. MANET-optimized examples include Ad hoc On-demand Distance Vector (AODV) routing [3] and Dynamic Source Routing (DSR) [4].

It is well-known that proactive-protocols are not optimal for either MANETs that have rapidly changing topologies or sensor networks that require emission control (EMCON) modes-of-operation. However, purely reactive protocols are often inappropriate for several common MANET topologies such as cluster-based networks and relatively static networks. In addition, reactive protocols introduce additional latency (and possibly source-routing overhead) for real-time traffic. As such, "hybrid" or "zone" routing protocols that use a mix of both proactive and reactive routing techniques at each network node have been proposed. Two examples are Cornell's Zone Routing Protocol (ZRP) [5] and SRC's "Wireless Ad hoc Routing Protocol (WARP)". WARP is based on Cornell's ZRP, with additional enhancements for Quality of Service (QoS) support. (Note: WARP resulted from a collaboration between SRC, Cornell University and Air Force Research Lab during 1999.)

There have been theoretical comparisons of proactive, reactive and hybrid routing protocols [6]. BBN's comparison of HSLS, DSR and ZRP then showed that (for fairly uniform node densities and traffic patterns) the asymptotic overhead for proactive, reactive and hybrid routing scaled as $O(N^{1.5})$, $O(N^2)$ and $O(N^{1.66})$, respectively, in large N-node networks. However, that work also noted that ZRP might have superior scaling

performance with respect to traffic load, non-uniform traffic patterns, and mobility-rates.

In general, hybrid routing's flexibility allows the network operator to adjust the protocol operation to match the network's current mission and state. For example, purely proactive operation might be used in relatively static networks such as inter-ship links. In contrast, purely reactive routing might be used in: a) dynamic networks such as clouds of tactical Unmanned Aerial Vehicles (UAVs); or b) networks of ground-based sensors that have strict Low Probability of Detection (LPD) requirements. These protocol adjustments could occur without changing the network software or "rebooting" any of the underlying MANET routers [5]. This flexibility is a tradeoff against the added complexity of hybrid protocols. As such, the next section describes the complexity of SRC's current software implementation of a hybrid routing protocol for MANETs.

2. MOBILEROUTE SOFTWARE

SRC's Wireless Ad hoc Routing Protocol (WARP) is based on Cornell's Zone Routing Protocol (ZRP) with additional enhancements for Quality of Service (QoS) support. The initial implementation of WARP provides best-effort routing and QoS routing based on link-stability and node-energy status.

MobileRoute™ is the commercial name for the current SRC multi-platform implementation of the WARP/ZRP concepts. The MobileRoute software development took place on x86 PCs and Red Hat Linux (6.2, 7.1 and 7.2 kernels) with a subsequent port to Compaq iPAQ Personal Digital Assistants (PDAs) running Debian Linux on 206 MHz StrongARM processors. The WARP software requires no kernel modifications. During 2002, it is being ported to other operating systems/platforms such as Windows CE and the Joint Tactical Radio System (JTRS).
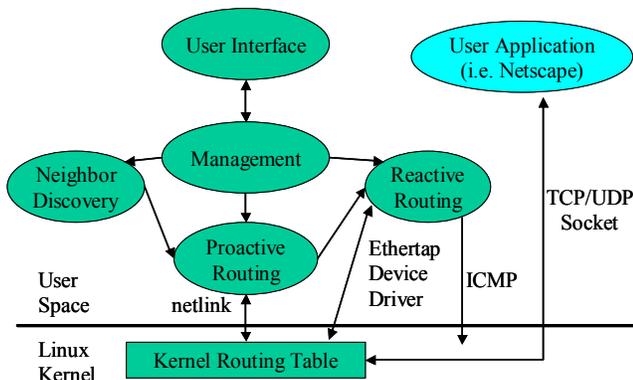

Figure 1. High-Level Software Architecture for MobileRoute Software

WARP abides by standard IP layering practices. It commits no layering violations, and resides above a COTS TCP/IP stack. WARP performs its on-demand route discovery and maintenance using UDP datagrams. The interface to the Linux kernel's routing tables then uses a clean, open interface (e.g., the netlink sockets library).

2.1 Protocol Operation

The WARP software's functionality is broken up into several processes. WARP's Neighbor Discovery Protocol (NDP) locates one-hop neighbors. Each node periodically multicasts a "Hello" message, while simultaneously listening for other nodes' "Hello" messages. The reception of another node's "Hello" message indicates a one-hop neighbor relationship between those two nodes. (Note: SRC chose to implement a Layer 3 neighbor-discovery process in order to improve portability across radio systems and increase compatibility with Military security-practices. Link-layer information about "up/down" links could also be leveraged within the MobileRoute architecture.)

WARP's Proactive Routing Protocol (PRP) is a timer-based link-state routing protocol. This approach allows better control over the routing-overhead. In contrast, "triggered" link-state protocols such as OSPF, which send every topology change throughout the entire network, can rapidly saturate mobile wireless networks with their routing overhead.

WARP's PRP builds and maintains an internal link-state table based on the neighbor information passed from NDP and the Link State Advertisements (LSA) received from other nodes in the source's local "zone". (Note: the simplest definition of a "zone" is all nodes within X hops of that source node.) LSAs are periodically multicast to the nodes in the source node's local zone – in order to propagate that node's view of its zone's network topology. The information in a node's internal link-state table is then used to modify the kernel's "best-effort" and "Quality of Service (QoS)" routing tables. The MobileRoute software (in conjunction with the Linux kernel) uses each IP packet's Differentiated Services CodePoint (DSCP) marking to determine which of these two routing tables is used to determine each packet's next-hop.

When an application attempts to send traffic to a node not contained in its node's routing zone, the application data is sent to the RRP process via the Ethertap interface. RRP sends a "Route Request" to the nodes on the edge of that node's routing zone. Each node that receives that request either answers it with a "Route Reply", if the node has a route to the destination, or forwards that request. When a Route Reply is received by the source node, the

route is stored in the Route Cache, and the application's data packets are then sent to the destination, embedded in "Source Routed Data Packets". Nodes may report invalid routes (due to topology changes) via a Route Error message. The recipient of a Route Error prunes the associated route from its Route Cache, and then sends an ICMP Host Unreachable message to the application whose data packet was using that broken route. This preserves a user's expectations for the behavior of existing applications such at Netscape. The appropriate hold-down timers are used for all RRP messages to avoid generating excessive routing overhead.

WARP's NDP and Reactive Routing Protocol (RRP) are separate from its PRP to allow flexibility in porting to handheld devices and microsensors. WARP's RRP uses explicit source-routing that provides end-to-end QoS support in conjunction with PRP. Its major difference with DSR is that WARP's RRP is a user-space application that should be more portable across operating systems. Its disadvantage is the added overhead associated with its UDP encapsulations for both data and control packets. (Note: the WARP software architecture is designed so that other group's PRPs and RRPs can be substituted for SRC's protocols.)

WARP User Interface (UI) is currently a command-line interface (CLI) to the underlying processes. Future plans include an SNMP interface and a Graphical User Interface. Finally, WARP's Management process reads the configuration files and configures (but does not modify) the underlying Linux kernel. It also provides the interface between the UI and the routing processes.

2.2 Quality of Service (QoS) Routing

A more general term for QoS-based routing is "constraint-based" routing wherein the route-selection is based on metrics other than simple hop-count. This is a known hard problem. Let $d(j,k)$ be a routing metric for a link between node j and k. Then, for any path $P = (j,k,l,...m,n)$, that path's routing metrics (d) may be classified as follows [7].

- Additive if: $d(P) = d(j,k) + \ldots + d(m,n)$
- Multiplicative if: $d(P) = d(j,k)* \ldots *d(m,n)$
- Concave if: $d(P) = min.(d(j,k), \ldots d(m,n))$
- Convex if: $d(P) = max.(d(j,k), \ldots d(m,n))$

The standard routing metrics of delay, jitter, cost and hop count are additive. Reliability is multiplicative, while bandwidth is concave. It is well-known [7] that finding optimal routes based on a combination of two, or more, additive and/or multiplicative constraints is NP-complete if those metrics are independent. Algorithms for routing on hop-count and bandwidth are much simpler [7].

Bellman Ford's algorithm or Dijkstra's algorithm suffices for those two constraints. For example, the route computation algorithm could first prune all paths that fail the minimum bandwidth requirement, and then run Bellman-Ford on the remaining candidate paths. As such, the MobileRoute software currently uses a combination of hop count and one, or more, concave/convex metrics that function like a maximum/minimum bandwidth constraint.

WARP supports both best-effort routing based on hop-count and QoS routing based on wireless-specific routing metrics such as link-stability and "node energy status". The "node energy status" metric allows preferential avoidance of routes through battery-operated handheld radios − in favor of ones through vehicle-mounted radios and/or AC-powered radios. Many mobile nodes have limited battery power, so maximum utilization of high-available battery power or nodes that are using AC power is often helpful in a mobile ad hoc network. Sensor networks are one example of this metric's intended application. This metric is implemented as a convex routing metric, since a path's "node-energy status" is calculated as the *inverse* of the fractional battery-power at the node (in that path) with the least remaining battery-life. Hence, it can be easily integrated with a hop-count routing metric. (Note: SRC used the inverse of the remaining battery-power because existing route-calculation algorithms (e.g., the Dijkstra's algorithm used by PRP) typically assume that higher link-metrics correspond to worse paths.) On Linux, the fractional battery-power can be read from the /proc/apm file. This Linux feature allowed SRC to emulate this metric using AC-powered laptops and desktops in our test-bed.

The link-stability routing metric gives preference to more "stable" routing-table entries. As such, it may be well suited to flows that require QoS support. TCP flows also prefer stable paths since packet loss forces the TCP flow-control mechanism into its "slow start" mode, which reduces a flow's end-to-end throughput. This metric can also be implemented as a convex routing metric, since a path's "stability" is equal to that of the least stable link. Hence, it can also be easily integrated with a hop-count routing metric.

3. SYSTEM ABSTRACTION LAYER

The System Abstraction Layer (SAL) shown in Figure 2 adds a software layer to the WARP architecture that abstracts common operating system features into a higher level API. Additionally, it offers client processes a unified event-driven framework for asynchronous inter-process and inter-client communication. The abstraction layer localizes all system-dependent code, making porting to other platforms much easier.

SRC's R&D addressed two general problems. The first is that most research-grade networking code is specific to one kernel version of one Operating System (OS). In contrast, DoD and commercial systems are heterogeneous. The second problem is that DoD acquisition programs now require the validation of networking protocols in large networks with 100's to 1000's of nodes. As such, the simulation of those protocols must use "real code" for the network layers and applications – so as to validate the correctness of production-software in large networks without running large-scale field exercises. This is a critical capability since differences between "kluged-up" simulation code and the production "gold-code" can produce significant performance-differences.

As such, SRC developed a "System Abstraction Layer" (SAL) for the MobileRoute software that allows portability between Operating Systems and platforms. The SAL decouples the MobileRoute software from the underlying OS's socket calls, memory management techniques and inter-process communications as shown in Figure 2. SRC's commercial goal was porting the MobileRoute software to Windows CE. In addition, SRC is porting MobileRoute to a JTRS Step 2C Radio during 2002. Section 3.2 describes how SRC also ported the MobileRoute software to the QualNet discrete-event simulation tool, which is one of the DARPA recommended simulation tools for the Future Combat System program.

3.1. WindowsCE

Windows currently has a much larger market-share than open-source software, and a wider range of supported hardware platforms and wireless-network adapter cards. With this in mind, SRC has ported WARP to a Windows CE-powered PDA running PocketPC2002, such as a Compaq iPAQ.

SRC had developed software to implement IP Differentiated Services (RFC 2474) on Windows NT hosts. SRC used a third-party's IP Security (IPsec) Client as a framework for implementing Differentiated Services (DiffServ) within a working Windows NT NDIS (Network Driver Interface Specification) driver. SRC enhanced the driver's source code to implement a subset of the DiffServ architecture, including packet marking and class-based queuing. SRC also developed a control application to allow the user to define per-application priorities according to well-known TCP and UDP ports. This information was stored and accessed by the NDIS driver that marked the packets. That NT NDIS driver ran on any PC running Windows NT or Windows 2000. The predominant goal of implementing DiffServ at the host was to extend IP QoS to the desktop, with an emphasis on improving IP-based services to cellular Personal Communications Services (PCS) users.

The aforementioned software (which is a requirement for MobileRoute on Windows), along with WARP, was ported to Windows CE during 2002. That port required all of the kernel-specific functions (sockets, semaphores, route table access, etc.) to be written to the appropriate Windows CE interfaces. Since the SAL contained the kernel-specific function calls, the majority of the software that was ported to WinCE was contained in the SAL.
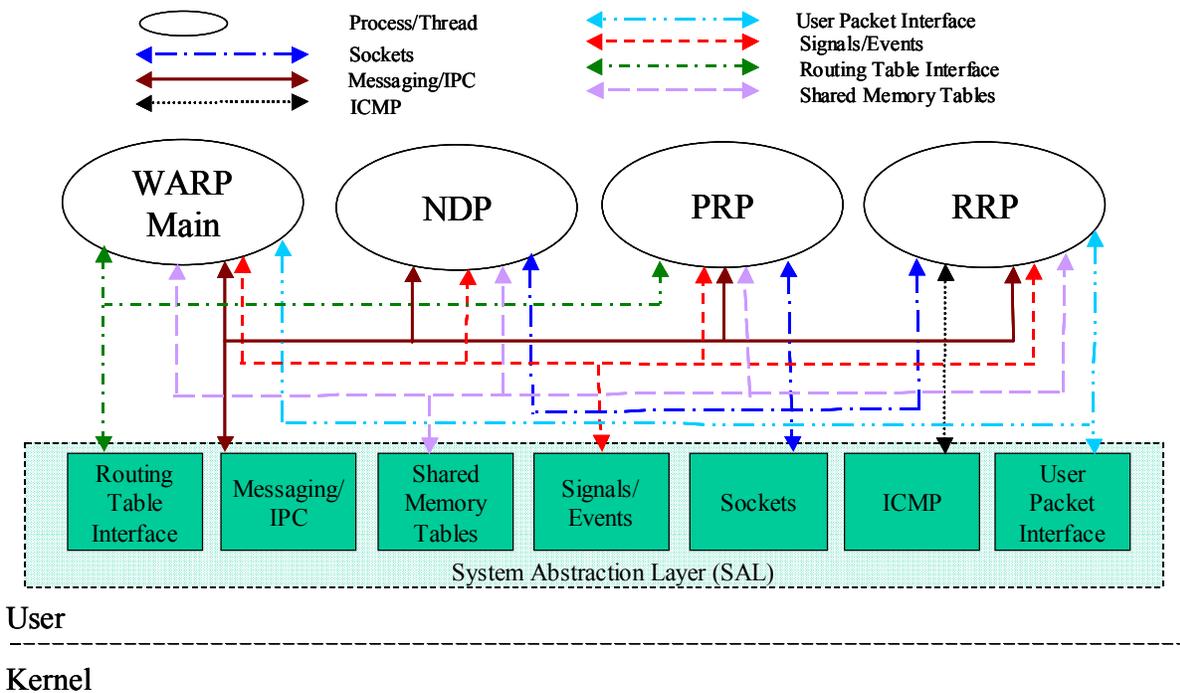
Figure 2. System Abstraction Layer (SAL)

In Linux, MobileRoute's NDP, PRP, RRP, and WARP Management pieces are all separate processes. For the port to Windows CE, SRC designed the MobileRoute software such that it was one process, with WARP Management, NDP, PRP, and RRP all being separate threads in the WARP PROCESS. WARP's NDP, PRP, and RRP "threads" then required minimal effort to port to Windows CE. The WARP Management process did require significant effort since it is responsible for: a) WARP-environment initialization; b) starting and monitoring the NDP, PRP, and RRP threads; c) acting as a dispatcher for messaging between the NDP, PRP, and RRP threads; and d) clean-up prior to termination. Much of the WARP Management functionality was specific to the platform that it was running on, and hence a larger effort was required to port that piece of the MobileRoute software.

On Windows CE, the IP Differentiated Services software described earlier was also needed by WARP to provide the equivalent functionality that the Ethertap driver provided in Linux. Since the IP Differentiated Services software was an NDIS driver, it had access to all incoming/outgoing IP packets and could provide the necessary functionality to the Windows CE version.

3.2. QualNet

QualNet (www.scalable-networks.com) is an event-driven network simulation tool that is a commercial version of the GloMoSim package developed under the DARPA Global Mobility (GLOMO) program. QualNet is optimized for modeling TCP/IP traffic across MANETs.

The main technical issue with porting routing-code to the QualNet simulation package was that "real" code ran on multiple nodes while QualNet emulated the same code on multiple virtual-entities. As such, the QualNet "Node Data Structure" (which indicated which virtual entity was currently being simulated) had to be passed to many of the WARP functions. The appropriate QualNet-related systems calls were also added to the SAL.

In order to call QualNet functions from within the SAL wrapper functions, the QualNet Node structure needed to be available. 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 (process or thread). By protecting this data within the SalTask structure, each task on each node was provided with its own "protected" memory region within the confines of the QualNet simulation environment.

QualNet is essentially callback-based; messages passed between layers are handled by layer-specific message handlers. SRC's SAL is also callback-based; applications provide callbacks to handle system events, such as message queues, timers expiring or data arriving on sockets. With the exception of startup code, the WARP application callback functions ran unmodified.

QualNet messages were simply translated to SAL events, and the WARP callbacks then executed as normal.

4. PERFORMANCE RESULTS

A previous paper [8] compared the performance of WARP and Optimized Link State Routing (OLSR) in a 14-node test-bed. That paper's performance metrics were the packet-loss fraction and the protocol overhead (e.g., number of routing packets generated per successfully delivered data-packet). This paper focuses on a "unit-test" of the MobileRoute software's CPU usage versus forwarded load. Experimental results for the MobileRoute software's processor utilization on a 650 MHz Pentium III processor are given for the Linux version of the SAL. This section also describes the experimental setup and the technique used to measure the CPU usage.

4.1. Experimental Setup

This paper used the three-node network shown in Figure 3 to perform three separate tests. In all three tests, Node A used nttcp to generate a traffic flow of IP packets from Node A to Node C.
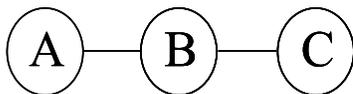

Figure 3. Test Network

The first test used static routes between all three nodes. This test measured the CPU usage (at Node B) of the Linux kernel's forwarding process and the overhead associated with sending/receiving IP packets. The second and third tests then used WARP as the routing protocol. In the second test, the zone radius of each node was set to two. In this case, each node used WARP's Neighbor Discovery Protocol (NDP) and Proactive Routing Protocol (PRP) to find a route from Node A to Node C. Packet forwarding still used the Linux kernel's forwarding process. In the third test, each node's zone radius was set equal to one. In this case, WARP's Reactive Routing Protocol (RRP) was needed to find a route from Node A to Node C. As such, this test also measured the overhead of RRP's user-space forwarding process at Node B.

4.2. Measuring CPU Usage

Since WARP forks off multiple processes after startup, it is not trivial to measure computational overhead directly by seeing how much processor time the *warpd* process consumes. Instead, this test uses an alternative approach

that uses a "cycle soaker" [9]. A cycle soaker is a tool that measures the overhead of the system by using a subtractive approach: it tries to consume as much processor time as it can, and then measures how much progress it has achieved in periodic intervals. The amount of work that the cycle soaker does depends on how many cycles are taken by other processes on the same system.

The cycle soaker reports the current CPU load on the system (from 0 to 100%) at one-second intervals. Since the cycle soaker runs at a low priority and performs little work, the measurement is an accurate representation of the other activity on the system. For this paper, the average CPU-usage is computed by taking the overheads reported by the cycle soaker over a 100 second interval and then averaging them.

4.3. Results and Discussion

Figure 4 shows results for the three tests described above for forwarding loads between 68 kbps and 469 kbps.
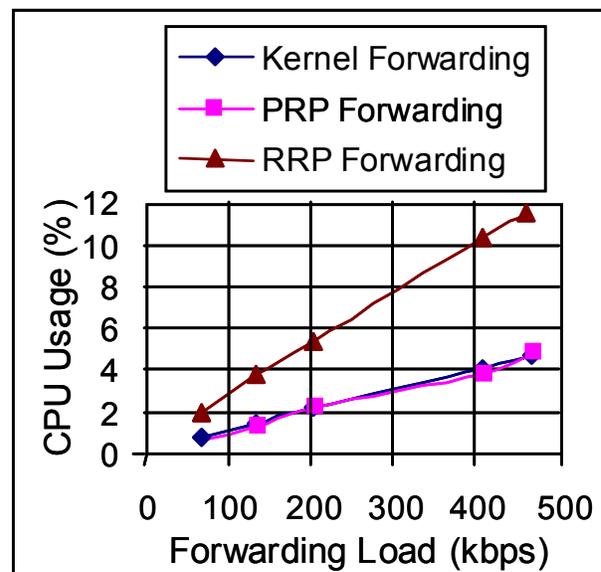

Figure 4. CPU Usage for MobileRoute Software

The CPU usage for the first two tests (labeled as "kernel forwarding" and "PRP forwarding") were virtually identical. This result was expected since the NDP and PRP timers were set to the defaults of 5 seconds and 10 seconds, respectively. As such, NDP and PRP provide negligible load to the system. The CPU usage for the third test was higher, since RRP's explicit source routing requires extra packet-copying between "user-space" and "kernel-space". The advantage though is that explicit source routing allows for better end-to-end QoS control in larger MANETs. It also allows load balancing and alternate path routing. If CPU usage is a paramount

concern then implementing a hybrid routing-protocol with a "stateless" RRP such as AODV is an option.

5. CONCLUSIONS

The main conclusion is that hybrid routing protocols can be implemented as portable application-layer software with reasonable complexity and performance. The MobileRoute software used in this paper is about 18,000 lines of code – of which about half is routing code and half is the overhead of the hybrid-routing framework and User Interface (UI). The CPU usage for RRP's user-space process showed the expected linear increase with forwarding load – with a utilization of 11.5% for a forwarding load of 461 kbps.

REFERENCES

[1] T. Clausen, et al, "Optimized Link State Routing Protocol", draft-ietf-manet-olsr-06.txt, Sept. 1, 2001. (See: www.ietf.org/html.charters/manet-charter.html for the latest versions of these Internet Drafts or subsequent RFCs.)

[2] R. Ogier, F. Templin, B. Bellur and M. Lewis, "Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)", draft-ietf-manet-tbrpf-05.txt, March 1, 2002.

[3] C. Perkins, E. Belding-Royer and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", draft-ietf-manet-aodv-11.txt, June 19, 2002.

[4] D. Johnson, D. Maltz, Y-C Hu and J. Jetcheva "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", draft-ietf-manet-dsr-07.txt, Feb., 2002.

[5] Z. Haas and M. Pearlman, "Determining the Optimal Configuration for the Zone Routing Protocol", IEEE JSAC, Special Issue on Ad-Hoc Networks, August 1999.

[6] C. Santivanez, "Asymptotic Behavior of Mobile Ad hoc Routing Protocols with respect to Traffic, Mobility and Size", Northeastern University CDSP Tech. Report TR-CDSP-00-52, Oct. 2000.

[7] X. Xiao and L. Ni, "Internet QoS: A Big Picture", IEEE Network, March/April 1999, pp. 8-18.

[8] P. Sholander, A. Yankopolus, P. Coccoli and S. Tabrizi, "Experimental Comparison of Hybrid and Proactive MANET Routing Protocols", MILCOM 2002, Oct. 2002.

[9] The Cycle Soaker tool can be found at http://www.uow.edu.au/~andrew/linux/#zc