Dartmouth Computer Science Technical Report TR2009-641

A Combined Routing Method for Ad Hoc Wireless Networks

Soumendra Nanda
Department of Computer Science
Dartmouth College

Zhenhui Jiang
SYSTRA USA

David Kotz
Institute for Security Technology Studies
Department of Computer Science
Dartmouth College

Corresponding author: snanda@cs.dartmouth.edu

Abstract— **Several simulation and real world studies show that certain ad hoc routing protocols perform better than others under specific mobility and traffic patterns. In order to exploit this phenomena, we propose a novel approach to adapt a network to changing conditions; we introduce "a combined routing method" that allows the network to seamlessly swap from one routing protocol to another protocol dynamically, while routing continues uninterrupted. By creating a thin new virtual layer, we enable each node in the ad hoc wireless network notify each other about the protocol swap and we do not make any changes to existing routing protocols. To ensure that routing works efficiently after the protocol swap, we reuse information from the previous protocol's routing table while initializing the data structures for the new routing protocol. We study the feasibility of our technique and the overheads incurred while swapping between AODV, ODMRP and APRL under different network topologies and traffic patterns through detailed simulations. Our results show that the swap latency is related to the nature of the destination protocol and the topology of the network. We also find that the control packet ratio of a routing protocol during and after a swap is close to that of the protocol running before a swap, thus indicating that our approach does not add excessive overhead.**

I. Introduction

A mobile ad hoc network (MANET) is a collection of moving computers connected by wireless links. By routing packets cooperatively among the nodes, these nodes can communicate with each other without any infrastructure. Thus, ad hoc networks are often proposed for use in emergency situations, such as disaster environments and military conflicts. It is important that ad hoc networks should react to network topological changes and traffic demands quickly and efficiently, and respect the inherent bandwidth and energy constraints [26]. Several researchers compare the performance of different ad hoc routing algorithms [21], [10], [5], [17]. Their results share a common theme; they all found that each routing algorithm can outperform the others in specific conditions, depending on the workload, terrain, network characteristics, or node mobility pattern.

Gray et al. compared four different routing algorithms [10]: AODV [27], ODMRP [16], APRL [15] and STARA [11], [12]. The authors used both simulations and real testbed experiments and found that under different wireless network conditions the relative performance was not the same. For example, ODMRP's message delivery ratio is better than AODV's ratio outdoors, while AODV has a higher message delivery ratio indoors [10]. Broch et al. [5] compared DSDV, TORA, DSR and AODV. They found that DSDVs routing overhead was almost constant with respect to mobility rate while TORA, DSR and AODVs routing overhead dropped as the mobility rate dropped. Lee et al. [17] compared ODMRP, AMRoute [4], CAMP [8], AMRIS [31], and flooding. They found that "in a mobile scenario, mesh-based protocols (ODMRP) outperformed tree-based protocols (AODV)", but they also pointed out that ODMRP showed "a trend of rapidly increasing overhead as the number of senders increased" [17]. Nanda et al. compared LAR, MLAR, AODV and AOMDV in extensive simulations in 2D and 3D mobility patterns [21]. They also found distinct advantages for one protocol over the other in different relative traffic and mobility conditions.

Ad hoc wireless network routing protocols are usually divided into two groups: Proactive (Table Driven) and Reactive (On-Demand) routing [28]. Proactive routing protocols compute the routes in advance while reactive routing protocols compute the routes only when necessary. Both have advantages and disadvantages. Thus several hybrid routing protocols have been proposed to combine both proactive and reactive routing modes [13], [23], [25]. The Zone Routing Protocol (ZRP) [13] divides the network into overlapping, variable-size zones. Routing within a zone uses proactive algorithms and routing between zones uses reactive algorithms. There are some other hybrid routing algorithms that combine proactive and reactive routing algorithms, such as HARP [23] and SHARP [25]. To reduce overhead, these hybrid methods group nearby nodes and use proactive routing algorithms within groups and use reactive routing algorithms between groups. Chen et al. proposed adaptive routing using clusters, which improves throughput by up to 80% [6]. Belding-Royer proposed hierarchical protocols to reduce overhead and gain more scalability [3]. However, since that technique uses higher-level topological information, the route to a destination might not be optimal, and the extra topological information itself requires more memory. Hoebeke et al. proposed an adaptive multi-mode routing algorithm [14]. Their method improved efficiency by switching to different protocols. To achieve this efficiency, however, they introduced many new components for the routing algorithm, which increased the complexity of each algorithm. Their proposed solution added a statistical component at the network layer and they collected non-local statistics through periodic broadcasting of a hello message to neighbors.

A common aspect of previous efforts is the attempt to modify the routing algorithm itself by combining multiple protocols into a new hybrid protocol that outperforms previous methods. Rather than creating a new adaptive routing algorithm, we aim to achieve better performance by dynamically switching to the best existing protocol for the current wireless network conditions. In this paper we focus on the mechanism for switching protocols, rather than the policy for choosing when to switch. Specifically, we develop and evaluate "the combined routing method," which is our mechanism to enable a network of nodes to switch dynamically to a new routing protocol. To the best of our knowledge, such a scheme that does not require any modification to existing routing protocol implementations, has never been studied or even proposed for wired or wireless networks.

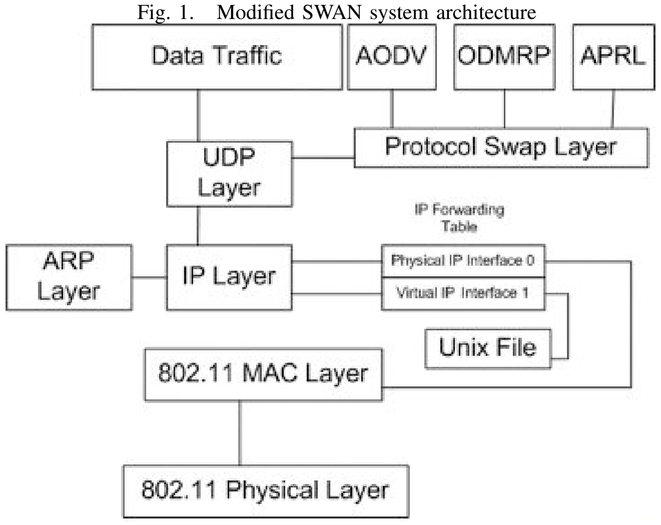
To simplify our combined method, we assume that we already know these existing protocols' characteristics in different environments, and that some mechanism exists to choose the best routing protocol based on the current network traffic pattern. As discussed in Section VI, we could use a centralized method to gather statistics about current network state, identify the traffic pattern, and then select a correct new protocol accordingly using techniques proposed by Qui et al [20].

In ad hoc networks, each node acts both as a host and a router. We thus use the term "node" instead of "host" or "router". We also use the two terms "routing algorithm" and "routing protocol" interchangeably. In Section II, we introduce three different routing algorithms, AODV, ODMRP, and APRL. We describe the differences among these three protocols and compare their performance. We also introduce SWAN, a simulator on which our experiments run. In Section III, we propose a method to switch among the three routing algorithms and discuss the implementation issues of this approach. In Section IV, we explain our experimental setup; in Section V we study the performance of this approach and in Section VI we discuss our results. In Section VII, we summarize and draw conclusions.

II. Background

We ran our routing protocols on the Dartmouth Simulator for Wireless Ad hoc Networks (SWAN) [30]. SWAN is built on the parallel discrete event driven simulator DaSSF [7], which is a C++ implementation of the Scalable Simulation Framework (SSF) [29]. DaSSF is particularly optimized for high performance when simulating large telecommunication systems [18] since DaSSF is able to simulate a network model that contains thousands of nodes. SWAN implements two layers of the 802.11 protocol: a pseudo-protocol-session for the physical layer and a protocol session for the MAC layer. SWAN also includes IP and ARP layers ported from the SSFNet [29] simulator code. A convenient feature of SWAN is that we can dynamically configure the protocol stack using the DML language. The protocol stack of the whole system, illustrated in Figure 1, is composed of five layers. Our routing protocols AODV, ODMRP, APRL are above the UDP layer.

Fig. 1. Modified SWAN system architecture



We used existing implementations of AODV, ODMRP and APRL from the Dartmouth ActComm project [1]. All three routing protocols are implemented in user space on Linux, and they use an IP tunnel and UDP sockets to perform their routing. An "IP tunnel" is a virtual network device that connects a UNIX device file and a network interface. Each node has a virtual IP address associated with the tunnel network interface, and a physical IP address associated with the real network interface in the node's IP forwarding table (Figure 1). At first, the application sends packets using the virtual IP address of the desired destination node. Then the packets are forwarded to the UNIX device file through the IP tunnel. After that, the routing engine converts the virtual IP address to a physical IP destination address, and finds the physical IP address of the next hop according to its routing table and pushes the packets down to the IP layer. These packets with a physical IP address are forwarded to the real network interface instead of the virtual network interface. The original virtual-addressed packet is thus wrapped in an IP packet addressed to the physical IP address of the next hop in the IP layer, in effect, tunneling the virtual network into the physical network. When a packet arrives, the simulator notifies the routing engine about this event and then the routing engine unwraps the packet and checks the virtual address to see whether the packet has reached the destination or needs to be forwarded again. Finally, when a packet arrives at the destination, the simulator notifies the routing engine and the routing engine writes the packet to the UNIX device file for delivery to the application. The system can be used for both simulation as well as actual field experiments [10].

A. Ad hoc On-demand Distance Vector Routing (AODV)

The Actcomm AODV implementation is an extension from the originally proposed AODV [27], adding the broadcast HELLO message. This implementation is capable of both unicast and broadcast routing. There are four types of control packets: RREQ, RERR, HELLO, and RREP. The first three are sent by broadcast, while RREP is by unicast.

B. On-Demand Multicast Routing Protocol (ODMRP)

ODMRP is a multicast on-demand routing protocol. There are two types of control packets: Join Query and Join Reply [2]. Join Query is sent by broadcast and Join Reply is sent by unicast. Both Join Query and Join Reply contain the originator and multicast group ID addresses. ODMRP uses multicast groups to keep member information. For each known node M in the whole network, ODMRP maintains a multicast group for that M, where the multicast group ID is Ms IP address. Each ODMRP node has two data structures in addition to the routing table: a multicast group table and a message cache. The multicast group table contains all the multicast groups for a node. The message cache is used to detect routing loops. The multicast group table contains expiration time and information about whether it knows a route to M or if it should receive data originated from M. Although ODMRP is a multicast protocol, similar to Bae et al. [9], we use it only as a unicast protocol.

C. Any Path Routing without Loops (APRL)

APRL is a unicast, proactive routing protocol [15]. There are two types of control packets: Beacon and PDVN. A node periodically broadcasts beacons to its neighbors. Each beacon contains the route information known by the sender. Ping Destination Via Neighbor (PDVN) packets are used to confirm the routes that the node receives in beacons. Upon startup, each node broadcasts a beacon message to its neighbors so that each nodes routing table only contains the destinations of its neighbors. After initializing the routing table with only its neighbor's information, each node broadcasts its own routing table to its neighbors periodically. If there is no route to a packet's required destination, the data packet is discarded; unlike AODV there is no route-request mechanism.

III. Implementation

In this section, we discuss our method to combine the three different protocols. Simply speaking, we insert a new layer between the routing protocols and the UDP layer (or equivalent layer on some other infrastructure). We call this new layer the Protocol-Swap Layer. Thus, the change in protocol is transparent to the lower layer (in our case, the UDP layer).

A. The combined method

Because we insert a new layer (the protocol-swap layer) between the routing layer and UDP layer, any control packet generated by the routing protocols is intercepted by the protocol-swap layer where the packet is wrapped with additional information; namely, the protocol type and the epoch number (Figure 2). These two extra fields specify the current protocol type and the freshness of the protocol respectively. For any received control packet, we first check the additional information at the protocol-swap layer, and then forward the control packet to the appropriate routing protocol (subject to some details discussed below).

Fig. 2. Packet format (top = old; bottom = new)

MAC	IP	UDP	Control Packet

MAC	IP	UDP	Epoch Num.	Control Packet

Because of the insertion of the protocol-swap layer shown in Figure 1, the protocol type and epoch number are transparent to the routing protocol layer. The advantage of this layer is that we only have to change the interface with the new protocol-swap layer and can reuse the routing part of the existing routing protocol codes. We also encapsulate all the code for swapping protocols in the protocol-swap layer. Our combined method adds little overhead because:
1) The two fields are only 4 bytes each, which is small yet enough to prevent wraparound ambiguity. Since a control packet is composed of a MAC header, an IP header, a UDP header, and a control packet body, the extra two fields do not use much extra bandwidth.
2) Only control packets are wrapped with the protocol type and epoch number while data packets remain the same as before.
3) During run time, only the routing table of the current protocol is maintained, and the other combined protocols' routing tables are empty. So the combined method does not use extra memory for additional routing tables.
4) Our implementation of the protocol-swap layer does not set up a virtual connection to other protocol-swap layers, which means this method does not add any new control packets.
5) We could add a traffic-monitor component in this layer. For example, if a node detects that the ratio of route requests is higher than normal, it might decide whether to swap to another routing protocol. This topic is beyond the scope of this paper, but we will discuss this topic in the section on future work.

B. The problems we need to solve

To implement the combined method, which can allow networks to swap from one protocol to another, there are three problems to solve:

1) Who determines when to swap, and how?
2) How is the swap decision communicated to all nodes?
3) How should each node adjust its internal tables?

While it is beyond the scope of this paper to determine when a swap should occur, we discuss a potential solution in Section VI-D. We assume for the rest of this paper that a single master node can initiate a protocol swap and notify all the nodes about the swap and focus on designing a practical and efficient technique to implement our combined method.

C. How is the swap decision communicated?

The master node communicates its decision to its neighbors by sending a control packet with the new protocol type and epoch number. The master node increments the epoch number and changes the protocol type every time it decides to swap. After the neighbors change to the new protocol, all their future

control packets will use this protocol type field and epoch number, thus diffusing the news. We do not add the protocol-swap layer header to data packets, however, because data packets do not need to know which routing protocol is used to find a path to the destination. Even if two nodes are using different routing protocols, they can still send data packets to each other, and the network can continue forwarding packets even while a swap is in progress. The mechanism for protocol swap requires each node to record its own notion of the current local protocol type and epoch number. It then compares the protocol-swap layer header of incoming packets to determine whether a new epoch has occurred and thus it is time to switch to a new protocol. There are two cases to consider:
Case 1: The received protocol number is the same as the local protocol number. *Case 1a:* The received epoch number is lower than its local epoch number; the node will discard the packet. *Case 1b:* The epoch number is equal; process the packet. *Case 1c:* The received epoch number is larger than the local epoch number; the node will update its local epoch number to be the received epoch number, and process the packet.
Case 2: The received protocol number is different from the local protocol number. *Case 2a:* The received epoch number is lower than or equal to its local epoch number; the node will discard the packet. *Case 2b:* The received epoch number is larger than the local epoch number; the node will update its local epoch number to be the received epoch number, swap to the received protocol, then process the packet.

D. How to actually swap protocols?

To swap, we need to initialize the new destination protocol's routing table and other data structures by using those of the current protocol. The primary goal when changing protocols is to build the routing table for the new protocols and to initialize, using as much information as possible from the routing table of the old protocol. We consider all six different cases for the swap: a) AODV to ODMRP or APRL, b) ODMRP to AODV or APRL, and c) APRL to AODV or ODMRP.

E. Reuse prior routing table entries

To take advantage of the prior protocols routing information, we reuse the entries in the prior routing table. However, the entries in the routing tables of AODV, ODMRP, and APRL are different, which complicates our effort to copy the entries between routing protocols. We copy any similar fields of two entries and choose a reasonable value for the fields that are different. It is important to note that AODV, ODMRP and APRL all have two key fields for routing: the destination IP address and the next-hop IP address. These two fields determine the next hop for forwarding packets to the destination. Since all these routing protocols use these two fields to determine any route, it is correct to copy these two IP addresses from the prior routing table entry to the new routing table entry. The other fields are used to determine the current status of the routes. AODV, ODMRP, and APRL keep different status of the routes for routing, so it might not be correct to reuse them in the new protocol, but we can carefully select a valid default value. We omit the details here for lack of space but present them in a thesis [32] and also comment on the correctness and the drawbacks of these default values. One key advantage of our reuse of prior routing table entries is that we are able to immediately use the old route after the swap, eliminating most of the potential costs of a swap.

F. Key data structures

To perform the swap, we must not only change the routing table, but each protocol's special associated data structures as well. We discuss each such data structure in turn.

AODV Precursor List: This data structure contains all the upstream nodes that use the node itself towards the same destinations. If the node determines that any one of its links is broken, as a hint it sends a RERR packet to those neighbors who are in its precursor list. When we swap to AODV, it is safe to leave the precursor list empty, because this data structure will be rebuilt when nodes later send out RREQ.

AODV Packet Queue: The source node queues any data packets that are yet to be sent in per-destination packet queues. When we swap from AODV to another protocol, we discard the packets in these queues and they are lost. We assume that some other mechanism (such as TCP) will realize that these packets did not reach their destinations and will resend those data again. The packet queue is AODV's unique data structure; other protocols do not have a queue for data packets. If we swap to AODV, we can simply create empty packet queues.

AODV RREQ Packet Cache: This data structure is used to store recently received RREQ packets to avoid loops. It may be created as empty when we swap to AODV, and may be discarded when we swap from AODV.

ODMRP Message Cache: This data structure is used to store recently received Join Query packets to avoid loops. It may be created as empty when we swap to ODMRP, and may be discarded when we swap from ODMRP.

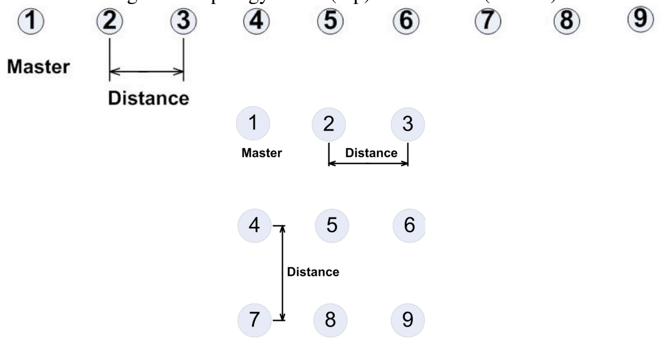
ODMRP Multicast Group Table: This data structure is used to maintain a list of multicast groups in which this node is a member and is checked when receiving a Join Query. If this node is in the multicast group, then it should accept the Join Query packet and send back a Join Reply. ODMRP has to rebuild the multicast group table via Join Queries when we swap to ODMRP. This data structure may be created as empty when we swap to ODMRP, and may be discarded when we swap from ODMRP.

APRL has no additional data structures, so there is nothing extra to do when swapping to or from APRL.

IV. Experiments

Our goal is to measure the overhead (in terms of time and traffic) due to a protocol swap. We chose a static network, which means all the nodes were preset to a certain position and would not move during the experiments. The effective transmission distance of the simulated node's radio was 73m. We ran the protocol for 200 seconds and the swap occurred

Fig. 3. Topology: Line (top) and Lattice (bottom)



at 100 seconds. We selected two types of network topology (Figure 3): 'line' and 'square lattice'.

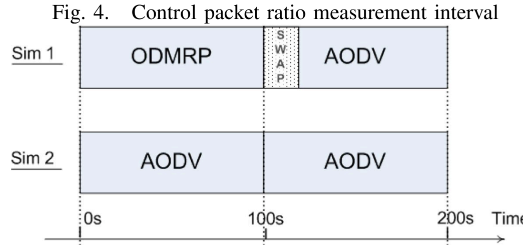
We select two network sizes: 9 nodes and 49 nodes. We select two traffic speeds: 1 data packet originated per node per second, or 1 data packet originated per node every 5 seconds. Each data packet's destination is chosen uniformly among the rest of the nodes. We run each parameter combination 5 times, each time with a different random seed for SWAN; we report the average result.

A. Metrics

We compare the performance of our combined method with plain AODV, APRL, and ODMRP. We used two metrics: the time to complete a protocol swap and the ratio of control packets per data packet sent from the UDP layer.

Metric 1: Time to complete a protocol swap. The swap time starts when the master node decides to swap, and ends when all the nodes in the network have updated their local protocol number and local epoch number. The metric is thus the difference between the swap end time and swap start time. This metric measures the *swap latency*.

Fig. 4. Control packet ratio measurement interval



Metric 2: Ratio of unicast and multicast control packets per data packet sent from UDP layer i.e, *the control packet ratio*. This metric helps us evaluate the efficiency of the destination protocol after swap.

AODV has four control packets: HELLO, RREQ, RREP, RERR. RREP is unicast and the rest are multicast. APRL has two control packets: BEACON, PDVN. PDVN is unicast and BEACON is multicast. ODMRP has two control packets: Join Query, Join Reply. Join Reply is unicast and Join Query is multicast. In all cases, the control packet ratio is the number of (unicast and multicast) control packets divided by the number of data packets sent from UDP layer.

TABLE I
TEST CONFIGURATIONS

Config.	Layout	Nodes	Dist.	Master node's neighbors
1	Line	9	20m	3
2	Line	49	20m	3
3	Square	9	25m	8
4	Square	49	30m	8

TABLE II
ASSOCIATION WITH NETWORK CONNECTIVITY

Config.	Layout	Nodes	Dist.	Max Swap Latency
1	Line	9	20m	10.004 sec
2	Line	49	20m	35.048 sec
3	Square	9	25m	1.813 sec
4	Square	49	30m	6.014 sec

We measure the control-packets ratio in three intervals using two simulations: 1) the interval after swap of the destination protocol; 2) the first half interval of different simulation of just the destination protocol; 3) the second half interval of the same simulation destination protocol. For example, in Figure 4, simulation 1 represents a swap from ODMRP to AODV starting at t=100s. Thus AODV is our destination protocol. Simulation 2 is a simulation of just the destination protocol for 200 seconds. So we compare the control packet ratio for the following three intervals of time: 1) the destination protocol after the swap from simulation 1 (time when the nodes finish the swap until end of simulation at t = 200s). 2) the destination protocol for first 100 seconds from sim 2, and 3) the destination protocol for second 100 seconds from sim 2.

B. Environment

We chose four configurations as shown in Table I. Referring back to Figure 3 and recalling the effective communication distance (73m), several nodes are in range of each node, including the master node. Although all nodes were connected directly or indirectly, we can see in Configuration 3 that all nodes were connected within the transmission range of each other, but in other configurations multi-hop communication was required.

V. SIMULATION RESULTS

We used the metrics we defined above to measure the efficiency of the swap for a given destination protocol. We identify the maximum or average swap latency to the same destination protocol from two sources. For example, in Table III, we use the average swap latency from ODMRP to AODV and APRL to AODV as the swap latency for AODV.

A. Swap latency

1) Association with network connectivity: Table II and Table III both show that swap latency is associated with the network connectivity for each type of swap. The highest connectivity (Configuration 3) has the best swap latency and

TABLE III
AVERAGE SWAP LATENCY (S) WITH DIFFERENT NETWORK TOPOLOGIES

Configuration	To AODV	To ODMRP	To APRL
9 node Line	0.988	2.620	7.518
49 node Line	3.945	4.6146	31.045
9 node Square	0.050	1.233	0.012
49 node Square	2.493	2.006	11.021

TABLE IV
AVERAGE SWAP LATENCY (S) WITH DIFFERENT TRAFFIC PATTERNS

	To AODV	To ODMRP	To APRL
Avg. Latency (Low Traffic)	1.788	1.435	12.029
Avg. Latency (High Traffic)	1.950	3.801	12.769

lowest connectivity (Configuration 2) has the worst swap latency.

2) Association with the network traffic: Table IV shows that for AODV and APRL, the swap latency were similar with heavy and low traffic workloads. For ODMRP, the heavy traffic swap latency is nearly twice as fast as the low traffic latency. Since ODMRP is a purely reactive routing protocol, it only sends Join Query when it needs to. So a busier traffic pattern generates more control traffic and thus spreads the news about the swap. But AODV and APRL both periodically broadcast message to its neighbors, so the swap interval is more dependent on the broadcast interval and not the traffic load.

3) Association with the destination protocol: Table V shows that AODV and ODMRP completed the swap quickly, while APRL was relatively slow. After a swap, ODMRP needs to broadcast Join Query packets to maintain its multicast group membership information. Similarly, AODV needs to broadcast RREQ if there is no route to the destination in the routing table after swap. But APRL drops the packets if it can not find route information in its routing table. Those route query broadcasting packets make swaps to AODV and ODMRP fast.

TABLE V
AVERAGE SWAP LATENCY (S) OVER ALL TESTED CONFIGURATIONS

	To AODV	To ODMRP	To APRL
Average Latency	1.8692	2.6186	12.3995

TABLE VI
MEASURED SWAP LATENCY (S) WITH THE DESTINATION PROTOCOL OVER ALL TESTED CONFIGURATIONS

Config. Format= Number of nodes/Layout/Traffic-pattern
Sq = Square; Ln = Line; Lo= Low Traffic; Hi= High Traffic

No.	Config.	To AODV	To ODMRP	To APRL
a	9/Ln/Hi	0.928	1.645	5.032
b	9/Ln/Lo	1.048	3.595	10.004
c	9/Sq/Hi	0.098	0.652	0.009
d	9/Sq/Lo	0.001	1.813	0.014
e	49/Ln/Hi	1.235	2.593	27.043
f	49/Ln/Lo	6.655	6.635	35.048
g	49/Sq/Hi	4.890	0.849	16.032
h	49/Sq/Lo	0.095	3.162	6.010

Fig. 5. AODV control packet ratio

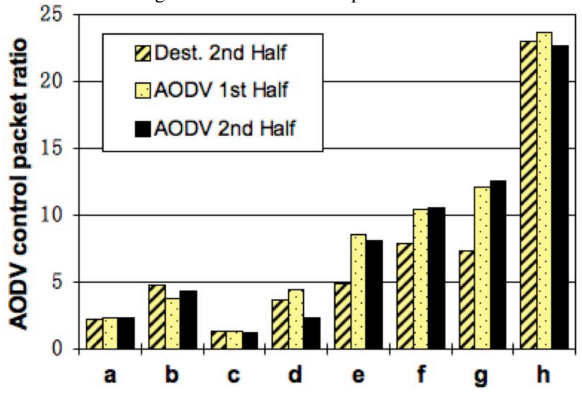


Fig. 6. ODMRP control packet ratio

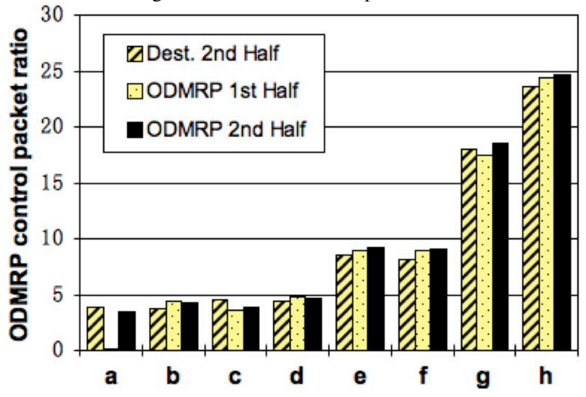
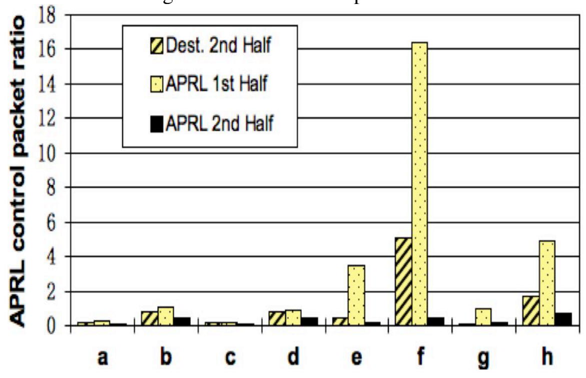


Fig. 7. APRL control packet ratio



In Table II, we can see that in high connectivity square configurations (rows 3 and 4), AODV and APRL swapped quickly. This is because they both periodically broadcast a message to neighbors. We can also see in Table VI that AODV and ODMRP completed their swaps quickly even in the low connectivity line networks. But APRL's swap time was highly related to the network topology. For example, AODV's swap latency range was from 0.001 to 6.665 seconds and ODMRP's ranged from 0.652 to 6.635 seconds. APRL's swap latency range was from 0.009 to 35.048 seconds, however, because APRL only broadcasts beacons to its neighbors periodically. So the swap latency depends on the period of these broadcasts rather than on the traffic patterns as in AODV or ODMRP.

B. The control packet ratio

In Figures 5, 6 and 7 we show the control packets ratio of eight types of network configurations corresponding to the same ones listed in columns 1 and 2 of Table VI. For each configuration and each type of swap, we can see that the control packets ratio after swap (the first bar in each group of three) was not the largest one of each group in most cases. Even if the control packets ratio after swap was the largest one of each group, the variance between the three bars was low. After the swap, in most cases, the destination routing protocol performed almost transparently, much as it would without a swap and data packets were routed successfully and with very little control overhead. For APRL the third bars are always small because APRL does not send many control packets in a static network once routes have stabilized and all destinations are reachable.

VI. Discussion

A. Latency to complete a protocol swap

First, we found that the swap latency depends on the network connectivity: highly connected networks had a better swap latency, because news of the swap had fewer hops to traverse. We also found that traffic workload influenced the swap latency of reactive routing protocols like ODMRP. Lastly, swap latency also depends on the characteristics of the destination protocol. Protocols that are reactive depend on the data traffic to generate control packets and thus propagate news of the swap; proactive protocols depend on periodic broadcasts to spread the news. If a protocol performs routing without flooding, the swap latency was long (particularly in less connected networks). In this case, a node might need to send back an empty control message to inform a sender about a new epoch and routing protocol if it receives any out-dated control packets from the sender.

B. Control traffic overhead for protocol swap

As the results show, the control-packet ratio after swap was lower than or close to the control-packet ratio of running a protocol without a swap. First, the swap does not require extra control packets to diffuse the swap information or rebuild tables. Second, because we initialize the new routing table using the old routes, we send few route-query packets. This situation is true only in static networks or low mobility networks. In high-mobility networks, there would be more control traffic to rebuild a route to the destination, and more lost data packets. However, the same would be true if the swap had not occurred. Thus we believe that our method efficiently transfers the network from one routing protocol to another, using no new packet types, reusing routing table information where possible, and not excessively increasing control traffic after a swap.

C. An alternative approach

Hoebeke et al. proposed an adaptive multi-mode routing protocol for ad hoc networks [14] but do not provide any performance results. Their adaptive method is similar to our combined method in principle: both want to dynamically swap to another protocol based on current network conditions, but there are three main differences.

1. Their adaptive method introduced a new type of control packet to the existing protocols, which is periodically broadcasted. Our method did not introduce any new message to the existing protocols. Thus, it is relatively easy to combine more protocols if necessary. On the other hand, we need to design $n \times (n-1)$ routing-table converters to combine n protocols.
2. In their method, different protocols share the same routing table, requiring all protocols to be re-implemented to suit a new, common routing table format. Each node also has a neighbor table to keep track of connectivity and neighbors modes (reactive or proactive). In our method, different protocols each maintain their own routing table, and we translate tables when we swap protocols. Thus, we do not need to alter current protocols or limit their design capabilities.
3. We have actually implemented our ideas on SWAN and can use our code for both simulations and real-world field tests. In simulations, we have analyzed the performance of the routing algorithm for swaps between three protocols.

D. Future work

To answer the critical question of when to swap, we are exploring various network monitoring techniques in a complementary project [22]. By identifying the profile of the network in a centralized manner, the master node can decide which routing protocol the network should use (depending on the desired performance criteria). After collecting all the required information, the master node can decide when to swap and to which protocol to swap in an automated manner by using a simulator to simulate the impact of the new protocol on the measured network state. Qui et al. proposed this simulator-based technique [20] to automate management of a wireless mesh network. Combining their approach with ours could provide a first step towards a complete solution.

We note that our results are based on simulations and should be considered tentative pending real-world experimentation. For our initial evaluation, we chose to use static networks to run our simulations and obtain an approximate lower bound for the cost of our approach. We intend to evaluate our technique in mobile scenarios in both simulations and real-world experiments using the emulation capabilities of SWAN [19] in the future.

One tradeoff we wish to explore is whether we should let the protocol-swap layer broadcast an empty dummy packet just to notify its neighbors about the swap. This broadcast should decrease the swap completion time by increasing the speed of disseminating news about the swap. In particular, this optimization would benefit protocols that do not broadcast periodically (such as APRL). We could also add some fields in the packet's header to carry traffic statistics besides protocol type and epoch number. There are many other avenues to explore to create a practical solution such as developing metrics to evaluate the before-and-after effects of each swap, distributing the functionality of the master node, accounting for errors in network measurement, buffering strategies, and local vs. global performance optimizations, to name just a few.

VII. Summary

We describe a method to combine AODV, ODMRP, and APRL (and other protocols) in such a way that we can swap from one protocol to another dynamically. For each pair of protocols, we identify how to initialize each protocol's data structure from the previous entries of the other protocol. We propose two metrics to measure the performance and simulate various network topologies and conditions using SWAN. Our results show that the time to complete a protocol swap depended on the characteristics of the protocol we swap to, the topology of the network, and the traffic on the network. Our combined method swapped slowly for the less-connected networks and for the protocols without flooding (like APRL) but was efficient in all other cases. In our combined routing method, from a software engineering point of view, we efficiently reuse the source code of the existing routing protocol by inserting a new layer to facilitate swaps, without changing any existing protocol implementations.

Acknowledgment

We wish to thank Jason Liu, Yougu Yuan and Bob Gray for their suggestions and help. This research program is a part of the Institute for Security Technology Studies, supported under Award number 2000-DT-CX-K001 from the U.S. Department of Homeland Security, Science and Technology Directorate and by Grant number 2005-DD-BX-1091 from the Bureau of Justice Assistance. Points of view are those of the authors.

References

[1] ActComm Project. http://actcomm.dartmouth.edu

[2] S. H. Bae, S. Lee, W. Su, M.Gerla. The Design, Implementation, and Performance Evaluation of the On-Demand Multicast Routing Protocol in Multihop Wireless Networks. *IEEE Network Magazine*, vol. 14, no. 1, Jan./Feb. 2000, pp. 70-77.

[3] E. Belding-Royer. Multi-Level Hierarchies for Scalable Ad Hoc Routing. *Wireless Networks*, Vol. 9, Issue 5, 2003, pp. 461-478.

[4] E. Bommaiah, M. Liu, A. McAuley, R. Talpade. AMRoute: Adhoc Multicast Routing Protocol. Internet-Draft, draft-talpade-manetamroute-00.txt, Aug. 1998; Work in progress.

[5] J. Broch, D. Maltz, D. Johnson, Y. C. Hu, J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proc. of MobiCom 1998*, Oct. 1998, pp. 85-97.

[6] Y. P. Chen, A. Liestman. A Zonal Algorithm for Clustering Ad Hoc Networks. International Journal of Foundations of Computer Science, 14(2), 2003, pp. 305-322.

[7] Dartmouth SSF. http://www.cis.fiu.edu/ liux/research/projects/dassf/

[8] J. J. Garcia-Luna-Aceves, E. L. Madruga. The Core-Assisted Mesh Protocol. *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, Aug. 1999, pp. 1380-1394.

[9] S. Bae, S. Lee, M. Gerla. Unicast Performance Analysis of the ODMRP in a Mobile Ad Hoc Network Testbed. In *Proc. of IEEE International Conference on Computer Communications and Networks (ICCCN)*, Oct. 2000, pp. 148-153.

[10] R. Gray, D. Kotz, C. Newport, N. Dubrovsky, A. Fiske, J. Liu, C. Masone, S. McGrath. Outdoor Experimental Comparison of Four Ad Hoc Routing Algorithms. In *Proc. of the Seventh ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2004)*, Venice, Italy, Oct. 2004.

[11] P. Gupta and P. Kumar. A System and Traffic Dependent Adaptive Routing Algorithm For Ad Hoc Networks. In *Proc. IEEE 36th Conf. on Decision and Control*, San Diego, CA, Dec. 1997, pp. 270-283.

[12] P. Gupta. Design and Performance Analysis of Wireless Network. Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, Aug. 2000.

[13] Z. Haas and M. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol, *IEEE/ACM Trans. Networking*, vol. 9, no. 4, Aug. 2001, pp. 427-438.

[14] J. Hoebeke, I. Moerman, B. Dhoedt, P. Demeester. Adaptive Multimode Routing in Mobile Ad Hoc Networks. In *Proc. of the 9th International Conference on Personal Wireless Communications*, Delft, The Netherlands, Sep. 2004, pp. 107-117.

[15] B. Karp and H. T. Kung. Dynamic Neighbor Discovery and Loop- Free, Multi-Hop Routing for Wireless Mobile Networks. Harvard University, May 1998. Draft.

[16] S. Lee, W. Su, M. Gerla. On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks. *ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communication in Wireless Mobile Networks*, 2000.

[17] S. Lee, W. Su, J. Hsu, M. Gerla, R. Bagrodia. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM)*, Tel Aviv, Israel, Mar. 2000, pp. 565-574.

[18] J. Liu and D. Nicol. DaSSF 3.1 User's Manual. Apr. 2001.

[19] J. Liu, Y. Yuan, D. Nicol, R. Gray, C. Newport, D. Kotz, L. Felipe Perrone. Simulation Validation Using Direct Execution of Wireless Ad-Hoc Routing Protocols. In *18th Workshop on Parallel and Distributed Simulation (PADS04)*, Kufstein, Austria, May 2004, pp. 7-16.

[20] Lili Qiu, Paramvir Bahl, Ananth Rao, Lidong Zhou. Troubleshooting Wireless Mesh Networks. *SIGCOMM Computer Communications Review*, 36(5), 2006, pp. 17–28.

[21] S. Nanda and R. Gray. MLAR in 2D and 3D. In *Proc. of 6th IEEE Wireless Communications and Networking Conference (WCNC)*, Las Vegas, April 2006

[22] S. Nanda and D. Kotz. Mesh-Mon: a multi-radio mesh monitoring and management system. *Computer Communications (2008)*, doi:10.1016/j.comcom.2008.01.046

[23] N. Navid, S. Wu, C. Bonnet. HARP: Hybrid Ad hoc Routing Protocol, *International Symposium on Telecommunications (IST 2001)*, Teheran, Iran, Sep. 2001.

[24] C. Perkins, E. Royer, S. Das. Ad Hoc On Demand Distance Vector Routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, Feb. 1999, pp. 90-100.

[25] V. Ramasubramanian, Z. Haas, E. G. Sirer. SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks. In *Proc. of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Annapolis, Maryland, June 2003. pp. 303-314.

[26] S. Corson, J. Macker. Mobile Ad hoc Networking: Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501, Jan. 1999.

[27] C. Perkins, E. Belding-Royer, S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, July 2003.

[28] E. Royer and C-K. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. *IEEE Personal Communications*, Vol. 6, April 1999, pp. 46-55.

[29] Scalable Simulation Framework. http://www.ssfnet.org/homePage.html

[30] Simulator of Wireless Ad hoc Networks. http://www.eg.bucknell.edu/swan/

[31] C.W. Wu, Y.C. Tay, C-K. Toh. Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) Functional Specification. Internet-Draft, draft-ietf-manet-amris-spec-00.txt, Nov. 1998, Work in progress.

[32] Zhenhui Jiang. A Combined Routing Method for Ad hoc Wireless Networks. MS Thesis, Dartmouth College Computer Science Technical Report TR2005-566.