PARALLEL SIMULATION OF WIRELESS ADHOC NETWORKS

***RAMAN CHADHA, @DR. SATYA DEV GARG,# SACHIN AHUJA**

**Research Scholar, Monad University*

@TCS Consultant, Noida

#A.P. VCE, Meerut

ABSTRACT

The Study of Parallel Simulation of Wireless Adhoc Networks (PSWANs) is a simple and well-organized routing designed for specifically use in multi-hop wireless ad hoc networks of mobile nodes. PSWANs allows the network to be completely self-organizing and self-configuring, without the required for any offered network administration.[1] The protocol is composed of the two mechanisms of path detection and Route preservation, which work together to allow nodes to discover and maintain source routes to arbitrary destinations in the ad hoc network. The use of source routing allows packet routing to be slightly loop-free, avoids the need for up-to-date routing information in the intermediate nodes through which packets are forwarded, and allows nodes forwarding or overhearing packets to cache the routing information in them for their own future use. All aspects of the protocol operate entirely on-demand, allowing the routing packet. We have evaluated the operation of CRWAN through detailed reproduction on a range of group and message patterns, and through execution and momentous experimentation in a physical outdoor ad hoc networking. In this chapter, we describe the design of CRWAN and offer a review of some of our simulation and test bed execution outcome for the protocol.

INTRODUCTION

The Study of Parallel Simulation of Wireless Adhoc Networks (PSWANs) is a simple and well-organized routing designed for specifically use in multi-hop wireless ad hoc networks of mobile nodes. Using PSWANs the network is completely self-organizing and self-configuring, requiring no existing network communications or administration. Network nodes (computers) cooperate to promote packets for each other to allow message over multiple "hops" between nodes not straight within wireless broadcast range of one another. As nodes in the network shift on or link or depart the network, and as wireless transmission conditions such as sources of interference change, all routing is automatically determined and maintained by the PSWAN.[2] Since the number or sequence of intermediate hops needed to achieve any destination may change at any moment, the resultant network topology may be quite rich and fast changing. The PSWANs protocol allows nodes to energetically determine a *source route* across several network hops to any destination in the ad hoc network. Each data packet sent then carries in its header the complete, ordered list of nodes through which the packet must pass, following packet routing to be insignificantly loop-free and avoiding the need for up-to-date routing information in the intermediate nodes through which the

packet is forwarded. By including this source route in the header of each data packet, other nodes forwarding or overhearing any of these packets may also easily cache this routing information for future use. The scope of our research includes protocol design, implementation, performance evaluation, and usage-based validation, spanning areas ranging roughly from portions of the ISO Data Link layer (layer 2) through the Presentation layer (layer 6).[3] In designing PSWAN, we sought to create a routing protocol that had very low overhead yet was able to react quickly to changes in the network, providing highly reactive service to help ensure successful delivery of data packets in spite of node movement or other changes in network conditions. The protocol specification for DSR has also been submitted to the Internet Engineering Task Force (IETF), the principal protocol standards development body for the Internet, and is currently one of the protocols under consideration in the IETF Mobile Ad Hoc Networks (MANET) Working Group for adoption as an Internet Standard for IP routing in ad hoc networks [MANET].[4] This chapter describes the design of the PSWAN protocol and provides a summary of some of our current imitation and test bed implementation results for PSWAN. In Section 2 of this chapter discusses our assumptions in the design of PSWAN. In Section 3, we present the design of the PSWAN protocol and describe the resulting important properties of this design. In particular, we describe here the design of the two mechanisms that make up the operation of PSWAN. Section 4 then summarizes some of our simulation results for PSWAN. Finally, we discuss related work in Section 5 and present conclusions in Section 6.

2 ASSUMPTIONS

We assume that all nodes wishing to communicate with other nodes within the ad hoc network are willing to participate fully in the protocols of the network. In particular, each node participating in the network should also be willing to forward packets for other nodes in the network. We refer to the minimum number of hops necessary for a packet to reach from any node located at one extreme edge of the ad hoc network to another node located at the opposite extreme, as the *diameter* of the ad hoc network. We assume that the diameter of an ad hoc network will often be small (e.g., perhaps 5 or 10 hops), but may often be greater than 1. Packets may be lost or corrupted in transmission on the wireless network. A node receiving a corrupted packet can detect the error and discard the packet. Nodes within the ad hoc network may move at any time without notice, and may even move continuously, but we assume that the speed with which nodes move is moderate with respect to the packet transmission latency and wireless transmission range of the particular underlying network hardware in use. In particular, PSWAN can support very rapid rates of arbitrary node mobility, but we assume that nodes do not continuously move so rapidly as to make the flooding of every individual data packet the only possible routing protocol. We assume that nodes may be able to enable *promiscuous* receive mode on their wireless network interface hardware, causing the hardware to deliver every received packet to the network driver software without filtering based on link-layer destination address. Although we do not require this facility, it is, for example, common in current LAN hardware for broadcast media

including wireless, and some of our optimizations can take advantage of its availability. Use of promiscuous mode does increase the software overhead on the CPU, but we believe that wireless network speeds are more the inherent limiting factor to performance in current and future systems. PSWAN can easily be used without the optimizations that depend on promiscuous receive mode, or can be programmed to only periodically switch the interface into promiscuous mode. Wireless communication ability between any pair of nodes may at times not work equally well in both directions, due for example to differing antenna or propagation patterns or sources of interference around the two nodes.[5] That is, wireless communications between each pair of nodes will in many cases be able to operate *bi-directionally*, but at times the wireless link between two nodes may be only *uni-directional*, allowing one node to successfully send packets to the other while no communication is possible in the reverse direction. Although many routing protocols operate correctly only over bidirectional links, PSWAN can successfully discover and forward packets over paths that contain uni-directional links. Some MAC protocols, however, such as MACA , MACAW, or IEEE , limit unicast data packet transmission to bi-directional links, due to the required bidirectional exchange of RTS and CTS packets in these protocols and due to the link-level acknowledgement feature in IEEE 802.11; when used on top of MAC protocols such as these, PSWAN can take advantage of additional optimizations, such as the route reversal optimization described below.

Each node selects a *single* IP address by which it will be known in the ad hoc network. Although a single node may have many different physical network interfaces, which in a typical IP network would each have a different IP address, we require each node to select one of these and to use only that address when participating in the DSR protocol. [6]This allows each node to be recognized by all other nodes in the ad hoc network as a single entity regardless of which network interface they use to communicate with it.

3 PSWAN PROTOCOL DESCRIPTION

3.1 Overview and Important Properties of the Protocol

The PSWAN protocol is composed of two mechanisms that work together to allow the discovery and maintenance of source routes in the ad hoc network:

 Route Discovery is the mechanism by which a node **S** wishing to send a packet to a destination node **D** obtains a source route to **D**. Route Discovery is used only when **S** attempts to send a packet to **D** and does not already know a route to **D**.

 Route Maintenance is the mechanism by which node **S** is able to detect, while using a source route to **D**, if the network topology has changed such that it can no longer use its route to **D** because a link along the route no longer works. When Route Maintenance indicates a source route is broken, **S** can attempt to use any other route it happens to know to **D**, or can invoke Route Discovery again to find a new route. Route Maintenance is used only when **S** is actually sending packets to **D**. Route Discovery and Route Maintenance each operate entirely *on demand*. In particular, unlike other protocols, CRWAN requires *no* periodic packets of *any kind* at *any level* within the network. For example, CRWAN does not use any periodic

routing advertisement, link status sensing, or neighbor detection packets, and does not rely on these functions from any underlying protocols in the network. [7]This entirely on-demand behavior and lack of periodic activity allows the number of overhead packets caused by CRWAN to scale all the way down to *zero*, when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packet overhead of PSWAN *automatically* scales to only that needed to track the routes currently in use. In response to a single Route Discovery, a node may learn and cache multiple routes to any destination. This allows the reaction to routing changes to be much more rapid, since a node with multiple routes to a destination can try another cached route if the one it has been using should fail. The operation of PSWAN are designed to allow uni-directional links and asymmetric routes to be easily supported. In particular, as noted in Section 2, in wireless networks, it is possible that a link between two nodes may not work equally well in both directions, due to differing antenna or propagation patterns or sources of interference. PSWAN allows such uni-directional links to be used when necessary, improving overall performance and network connectivity in the system. PSWAN also supports internetworking between different types of wireless networks, allowing a source route to be composed of hops over a combination of any types of networks available . For example, some nodes in the ad hoc network may have only short-range radios, while other nodes have both short-range and long-range radios; the combination of these nodes together can be considered by CRWAN as a single ad hoc network.

3.2 Basic CRWAN Route Discovery

When some node **S** originates a new packet destined to some other node **D**, it places in the header of the packet a *source route* giving the sequence of hops that the packet should follow on its way to **D**. Normally, **S** will obtain a suitable source route by searching its *Route Cache* of routes previously learned, but if no route is found in its cache, it will initiate the Route Discovery protocol to dynamically find a new route to **D**. In this case, we call **S** the *initiator* and **D** the *target* of the Route Discovery. For example, Figure 1 illustrates an example Route Discovery, in which a node **A** is attempting to discover a route to node **E**. To initiate the Route Discovery, **A** transmits a ROUTE REQUEST message as a single local broadcast packet, which is received by (approximately) all nodes currently within wireless transmission range of **A**. Each ROUTE REQUEST message identifies the initiator and target of the Route Discovery, and also contains a unique *request id*, determined by the initiator of the REQUEST. Each ROUTE REQUEST also contains a record listing the address of each intermediate node through which this particular copy of the ROUTE REQUEST message has been forwarded. This route record is initialized to an empty list by the initiator of the Route Discovery.

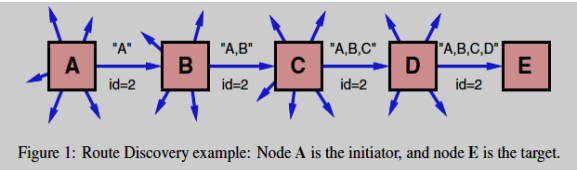

Figure 1: Route Discovery example: Node **A** is the initiator, and node **E** is the target.

When another node receives a ROUTE REQUEST, if it is the target of the Route Discovery, it returns a ROUTE REPLY message to the initiator of the Route Discovery, giving a copy of the accumulated route record from the ROUTE REQUEST; when the initiator receives this ROUTE REPLY, it caches this route in its Route Cache for use in sending subsequent packets to this destination. Otherwise, if this node receiving the ROUTE REQUEST has recently seen another ROUTE REQUEST message from this initiator bearing this same request id, or if it finds that its own address is already listed in the route record in the ROUTE REQUEST message, it discards the REQUEST. Otherwise, this node appends its own address to the route record in the ROUTE REQUEST message and propagates it by transmitting it as a local broadcast packet (with the same request id).

In returning the ROUTE REPLY to the initiator of the Route Discovery, such as node **E** replying back to **A** in Figure 1, node **E** will typically examine its own Route Cache for a route back to **A**, and if found, will use it for the source route for delivery of the packet containing the ROUTE REPLY. Otherwise, **E** may perform its own Route Discovery for target node **A**, but to avoid possible infinite recursion of Route Discoveries, it must piggyback this ROUTE REPLY on its own ROUTE REQUEST message for **A**. It is also possible to piggyback other small data packets, such as a TCP SYN packet , on a ROUTE REQUEST using this same mechanism. Node **E** could also simply reverse the sequence of hops in the route record that it trying to send in the ROUTE REPLY, and use this as the source route on the packet carrying the ROUTE REPLY itself. For MAC protocols such as IEEE 802.11 that require a bi-directional frame exchange as part of the MAC protocol , this route reversal is preferred as it avoids the overhead of a possible second Route Discovery, and it tests the discovered route to ensure it is bi-directional before the Route Discovery initiator begins using the route. However, this technique will prevent the discovery of routes using uni-directional links. [8]In wireless environments where the use of uni-directional links is permitted, such routes may in some cases be more efficient than those with only bi-directional links, or they may be the only way to achieve connectivity to the target node.

When initiating a Route Discovery, the sending node saves a copy of the original packet in a local buffer called the *Send Buffer*. The Send Buffer contains a copy of each packet that cannot be transmitted by this node because it does not yet have a source route to the packet's destination. Each packet in the Send Buffer is stamped with the time that it was placed into the Buffer and is discarded after residing in the Send Buffer for some timeout period; if necessary for preventing the Send Buffer from overflowing, a FIFO or other replacement strategy can also be used to evict packets before they expire. While a packet remains in the Send Buffer, the node should occasionally initiate a new Route Discovery for the packet's destination address. However, the node must limit the rate at which such new Route Discoveries for the same address are initiated, since it is possible that the destination node is

not currently reachable. In particular, due to the limited wireless transmission range and the movement of the nodes in the network, the network may at times become partitioned, meaning that there is currently no sequence of nodes through which a packet could be forwarded to reach the destination. Depending on the movement pattern and the density of nodes in the network, such network partitions may be rare or may be common. If a new Route Discovery was initiated for each packet sent by a node in such a situation, a large number of unproductive ROUTE REQUEST packets would be propagated throughout the subset of the ad hoc network.

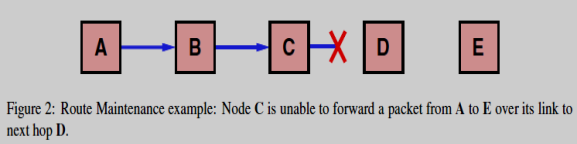

Figure 2: Route Maintenance example: Node C is unable to forward a packet from A to E over its link to next hop D.

reachable from this node. In order to reduce the overhead from such Route Discoveries, we use exponential back-off to limit the rate at which new Route Discoveries may be initiated by any node for the same target. If the node attempts to send additional data packets to this same node more frequently than this limit, the subsequent packets should be buffered in the Send Buffer until a ROUTE REPLY is received, but the node must not initiate a new Route Discovery until the minimum allowable interval between new Route Discoveries for this target has been reached. This limitation on the maximum rate of Route Discoveries for the same target is similar to the mechanism required by Internet nodes to limit the rate at which ARP REQUESTs are sent for any single target IP address .

3.3 Basic PSWAN Route Maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that the packet has been received by the next hop along the source route; the packet is retransmitted (up to a maximum number of attempts) until this confirmation of receipt is received. For example, in the situation illustrated in Figure 2, node **A** has originated a packet for **E** using a source route through intermediate nodes **B**, **C**, and **D**. In this case, node **A** is responsible for receipt of the packet at **B**, node **B** is responsible for receipt at **C**, node **C** is responsible for receipt at **D**, and node **D** is responsible for receipt finally at the destination **E**. This confirmation of receipt in many cases may be provided at no cost to DSR, either as an existing standard part of the MAC protocol in use (such as the link-level acknowledgement frame defined by IEEE 802.11, or by a *passive acknowledgement* (in which, for example, **B** confirms receipt at **C** by overhearing **C** transmit the packet to forward it on to **D**). If neither of these confirmation mechanisms are available, the node transmitting the packet may set a bit in the packet's header to request a CRWAN specific software acknowledgement be returned by the next hop; this software acknowledgement will normally be transmitted directly to the sending node, but if the link between these two nodes is uni-directional, this software acknowledgement may travel over a different, multi-hop path. If the packet is retransmitted by some hop the maximum number of times and no receipt confirmation is received, this node returns a ROUTE ERROR message to the original sender of the packet, identifying the link over which the packet could not be forwarded. For

example, in Figure 2, if **C** is unable to deliver the packet to the next hop **D**, then **C** returns a ROUTE ERROR to **A**, stating that the link from **C** to **D** is currently "broken." Node **A** then removes this broken link from its cache; any retransmission of the original packet is a function for upper layer protocols such as TCP. For sending such a retransmission or other packets to this same destination **E**, if **A** has in its Route Cache another route to **E** (for example, from additional ROUTE REPLYs from its earlier Route Discovery, or from having overheard sufficient routing information from other packets), it can send the packet using the new route immediately. Otherwise, it may perform a new Route Discovery for this target.

3.4 Additional Route Discovery Features

3.4.1 Caching Overheard Routing Information

A node forwarding or otherwise overhearing any packet may add the routing information from that packet to its own Route Cache. In particular, the source route used in a data packet, the accumulated route record in a ROUTE REQUEST, or the route being returned in ROUTE REPLY may all be cached

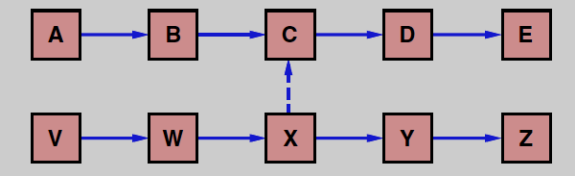

Figure 3: Limitations on caching overheard routing information: Node C is forwarding packets to E and overhears packets from X.

by any node. Routing information from any of these packets received may be cached, whether the packet was addressed to this node, sent to a broadcast (or multicast) MAC address, or received while the node's network interface is in promiscuous mode. One limitation, however, on caching of such overheard routing information is the possible presence of uni-directional links in the ad hoc network (Section 2). For example, Figure 3 illustrates a situation in which node **A** is using a source route to communicate with node **E**. As node **C** forwards a data packet along the route from **A** to **E**, it can always add to its cache the presence of the "forward" direction links that it learns from the headers of these packets, from itself to **D** and from **D** to **E**. However, the "reverse" direction of the links identified in the packet headers, from itself back to **B** and from **B** to **A**, may not work for it since these links might be uni-directional. If **C** knows that the links are in fact bi-directional, for example due to the MAC protocol in use, it could cache them but otherwise should not. Likewise, node **V** in Figure 3 is using a different source route to communicate with node **Z**. If node **C** overhears node **X** transmitting a data packet to forward it to **Y** (from **V**), node **C** should consider whether the links involved can be known to be bi-directional or not before caching them. If the link from **X** to **C** (over which this data packet was received) can be known to be bi-directional, then **C** could cache the link from itself to **X**, the link from **X** to **Y**, and the link from **Y** to **Z**. If all links can be assumed to be bi-directional, **C** could also cache the links from **X** to **W** and from **W** to **V**. Similar considerations apply to the routing information that

might be learned from forwarded or otherwise overheard ROUTE REQUEST or ROUTE REPLY packets.

3.4.2 Replying to ROUTE REQUESTs using Cached Routes

A node receiving a ROUTE REQUEST for which it is not the target, searches its own Route Cache for a route to the target of the REQUEST. If found, the node generally returns a ROUTE REPLY to the initiator itself rather than forwarding the ROUTE REQUEST. In the ROUTE REPLY, it sets the route record to list the sequence of hops over which this copy of the ROUTE REQUEST was forwarded to it, concatenated with its own idea of the route from itself to the target from its Route Cache. However, before transmitting a ROUTE REPLY packet that was generated using information from its Route Cache in this way, a node must verify that the resulting route being returned in the ROUTE REPLY, after this concatenation, contains no duplicate nodes listed in the route record. For example, Figure 4 illustrates a case in which a ROUTE REQUEST for target **E** has been received by node **F**, and node **F** already has in its Route Cache a route from itself to **E**. The concatenation of the accumulated route from the ROUTE REQUEST and the cached route from **F**'s Route Cache would include a duplicate node in passing from **C** to **F** and back to **C**.

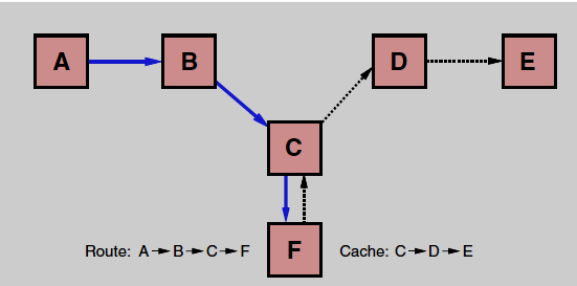

Figure 4: A possible duplication of route hops avoided by the Chaotic Routing limitation on replying to ROUTE REQUESTs from the Route Cache.

Node **F** in this case *could* attempt to edit the route to eliminate the duplication, resulting in a route from **A** to **B** to **C** to **D** and on to **E**, but in this case, node **F** would not be on the route that it returned in its own ROUTE REPLY. DSR Route Discovery prohibits node **F** from returning such a ROUTE REPLY from its cache for two reasons. First, this limitation increases the probability that the resulting route is valid, since **F** in this case should have received a ROUTE ERROR if the route had previously stopped working. Second, this limitation means that a ROUTE ERROR traversing the route is very likely to pass through any node that sent the ROUTE REPLY for the route (including **F**), which helps to ensure that stale data is removed from caches (such as at **F**) in a timely manner. Otherwise, the next Route Discovery initiated by **A** might also be contaminated by a ROUTE REPLY from **F** containing the same stale route. If the ROUTE REQUEST does not meet these restrictions, the node (node **F** in this example) discards the ROUTE REQUEST rather than replying to it or propagating it.

3.4.3 Preventing ROUTE REPLY Storms

The ability for nodes to reply to a ROUTE REQUEST based on information in their Route Caches, as described in Section 3.4.2, could result in a possible ROUTE REPLY "storm" in some cases. In particular, if a node broadcasts a ROUTE REQUEST for a target node for which the node's neighbors have a route in their Route Caches, each neighbor may attempt to send a ROUTE REPLY, thereby wasting bandwidth and possibly increasing the number of network collisions in the area. For example, in the situation shown in Figure 5, nodes **B**, **C**, **D**, **E**, and **F** all receive **A**'s ROUTE REQUEST for target **G**, and each have the indicated route cached for this target. Normally, they would all attempt to reply from their own Route Caches, and would all send their REPLYs at about the same time since they all received the broadcast ROUTE REQUEST at about the same time. [9] If a node can put its network interface into promiscuous receive mode, it should delay sending its own ROUTE REPLY for a short period, while listening to see if the initiating node begins using a shorter route first. That is, this node should delay sending its own ROUTE REPLY for a random period d = Hx(h -1+ r) where h is the length in number of network hops for the route to be returned in this node's ROUTE REPLY, r is a random number between 0 and 1, and H is a small constant delay (at least twice the maximum wireless link propagation delay) to be introduced per hop. This delay effectively randomizes the time at which each node

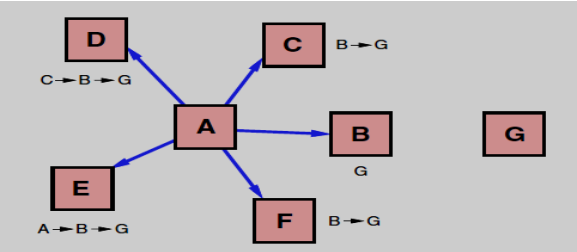

Figure 5: A ROUTE REPLY storm could result if many nodes all reply to the same ROUTE REQUEST from their own Route Caches. The route listed next to each node shows the route to destination G currently listed in that node's Route Cache.

sends its ROUTE REPLY, with all nodes sending ROUTE REPLYs giving routes of length less than h sending their REPLYs before this node, and all nodes sending ROUTE REPLYs giving routes of length greater than h sending their REPLYs after this node.

3.4.4 ROUTE REQUEST Hop Limits

Each ROUTE REQUEST message contains a "hop limit" that may be used to limit the number of intermediate nodes allowed to forward that copy of the ROUTE REQUEST. As the REQUEST is forwarded, this limit is decremented, and the REQUEST packet is discarded if the limit reaches zero before finding the target. We currently use this mechanism to send a *nonpropagating* ROUTE REQUEST (i.e., with hop limit 0) as an inexpensive method of determining if the target is currently a neighbor of the initiator or if a neighbor node has a route to the target cached (effectively using the neighbors' caches as an extension of the initiator's own cache). If no ROUTE REPLY is received after a short timeout, then a *propagating* ROUTE REQUEST (i.e., with no hop limit) is sent. We have also considered

using this mechanism to implement an *expanding ring* search for the target [Johnson 1996a]. For example, a node could send an initial nonpropagating ROUTE REQUEST as described above; if no ROUTE REPLY is received for it, the node could initiate another ROUTE REQUEST with a hop limit of 1. For each ROUTE REQUEST initiated, if no ROUTE REPLY is received for it, the node could double the hop limit used on the previous attempt, to progressively explore for the target node without allowing the ROUTE REQUEST to propagate over the entire network. However, this expanding ring search approach could have the effect of increasing the average latency of Route Discovery, since multiple Discovery attempts and timeouts may be needed before discovering a route to the target node.

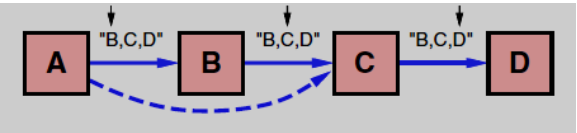

Figure 6: Node C notices that the source route to D can be shortened, since it overheard a packet from A intended first for B.

3.5 Additional Route Maintenance Features

3.5.1 Packet Salvaging

After sending a ROUTE ERROR message as part of Route Maintenance as described in Section 3.3, a node may attempt to *salvage* the data packet that caused the ROUTE ERROR rather than discarding it. To attempt to salvage a packet, the node sending a ROUTE ERROR searches its own Route Cache for a route from itself to the destination of the packet causing the ERROR. If such a route is found, the node may salvage the packet after returning the ROUTE ERROR by replacing the original source route on the packet with the route from its Route Cache. The node then forwards the packet to the next node indicated along this source route. For example, in Figure 2, if node **C** has another route cached to node **E**, it can salvage the packet by applying this route to the packet rather than discarding the packet.

When salvaging a packet in this way, the packet is also marked as having been salvaged, to prevent a single packet being salvaged multiple times. Otherwise, it could be possible for the packet to enter a routing loop, as different nodes repeatedly salvage the packet and replace the source route on the packet with routes to each other. An alternative mechanism of salvaging that we have considered would be to replace only the unused suffix of the original route (the portion in advance of this node) with the new route from this node's Route Cache, forming a new route whose prefix is the original route and whose suffix is the route from the Cache. In this case, the normal rules for avoiding duplicated nodes being listed in a source route are sufficient to avoid routing loops.

3.5.2 Automatic Route Shortening

Source routes in use may be automatically shortened if one or more intermediate hops in the route become no longer necessary. This mechanism of automatically shortening routes in use is somewhat similar to the use of passive acknowledgements. In particular, if a node is able to overhear a packet carrying a source route (e.g., by operating its network interface in

promiscuous receive mode), then this node examines the unused portion of that source route. [10] If this node is not the intended next hop for the packet but is named in the later unused portion of the packet's source route, then it can infer that the intermediate nodes before itself in the source route are no longer needed in the route. For example, Figure 6 illustrates an example in which node **C** has overheard a data packet being transmitted from **A** to **B**, for later forwarding to **C**; the arrow pointing to one node in the source route in each packet indicates the intended next receiver of the packet along the route. In this case, this node (node **C**) returns a *gratuitous* ROUTE REPLY message to the original sender of the packet (node **A**).

3.5.3 Increased Spreading of ROUTE ERROR Messages

When a source node receives a ROUTE ERROR for a data packet that it originated, this source node propagates this ROUTE ERROR to its neighbors by piggybacking it on its next ROUTE REQUEST. In this way, stale information in the caches of nodes around this source node will not generate ROUTE REPLYs that contain the same invalid link for which this source node received the ROUTE ERROR. For example, in the situation shown in Figure 2, node **A** learns from the ROUTE ERROR message from **C**, that the link from **C** to **D** is currently broken. It thus removes this link from its own Route Cache and initiates a new Route Discovery (if it doesn't have another route to **E** in its Route Cache). On the ROUTE REQUEST packet initiating this Route Discovery, node **A** piggybacks a copy of this ROUTE ERROR message, ensuring that the ROUTE ERROR message spreads well to other nodes, and guaranteeing that any ROUTE REPLY that it receives (including those from other node's Route Caches) in response to this ROUTE REQUEST does not contain a route that assumes the existence of this broken link. We have also considered, but not simulated, a further improvement to Route Maintenance in which a node, such as **A** in Figure 4, that receives a ROUTE ERROR will forward the ERROR along the same source route that resulted in the ERROR.

3.5.4 Caching Negative Information

In some cases, CRWAN could potentially benefit from nodes caching "negative" information in their Route Caches. For example, in Figure 2, if node **A** caches the fact that the link from **C** to **D** is currently broken (rather than simply removing this hop from its Route Cache), it can guarantee that no ROUTE REPLY that it receives in response to its new Route Discovery will be accepted that utilizes this broken link. A short expiration period must be placed on this negative cached information, since while this entry is in its Route Cache, **A** will otherwise refuse to allow this link in its cache, even if this link begins working again. Another case in which caching negative information in a node's Route Cache might be useful is the case in which a link is providing highly variable service, sometimes working correctly but often not working. This situation could occur, for example, in the case in which the link is near the limit of the sending node's wireless transmission range and there are significant sources of interference (e.g., multipath) near the receiving node on this link.

3.6 Multicast Routing with PSWAN

PSWAN does not currently support true multicast routing, but does support an approximation of this that is sufficient in many network contexts. Through an extension of the Route Discovery mechanism, PSWAN supports the controlled flooding of a data packet to all nodes in the ad hoc network that are within some specified number of hops of the originator; these nodes may then apply destination address filtering (e.g., in software) to limit the packet to those nodes subscribed to the packet's indicated multicast destination address. While this mechanism does not support pruning of the broadcast tree to conserve network resources, it can be used to distribute information to all nodes in the ad hoc network subscribed to the destination multicast address.[11] This mechanism may also be useful for sending application level packets to all nodes in a limited range around the sender.

4. PSWAN EVALUATION

This section summarizes some of our experiences evaluating PSWAN through detailed studies using discrete event simulation, and through implementation and actual operation and experience with the protocol in an ad hoc networking testbed environment.

5 RELATED WORK

Research in the area of routing in multi-hop wireless ad hoc networks dates back at least to 1973, when the U.S. Defense Advanced Research Projects Agency (DARPA) began the Packet Radio Network (PRNET)
project [Jubin 1987]. PRNET and its successor, the Survivable Adaptive Networks (SURAN) project [Lauer 1995], generated a substantial number of fundamental results in this area. With the increasing capabilities and decreasing costs of small, portable computers such as laptops and PDAs (Personal Digital Assistants), and with the increasing availability of inexpensive wireless network interface devices such as wireless LAN interfaces packaged as PCMCIA PC Cards, a growing number of other research projects in ad hoc networking have developed, some of which are described in other chapters of this book. In our discussion of related work here, we concentrate on research specifically related to the PSWAN protocol. As noted in Section 1, the design specification for CRWAN has also been submitted to the MANET (Mobile Ad Hoc Networks) Working Group of the IETF (Internet Engineering Task Force) in their efforts to standardize a protocol for routing of IP packets in an ad hoc network [Broch 1999a, MANET].

The original motivation in the design of CRWAN came from the operation of the Address Resolution Protocol (ARP) used in the TCP/IP suite of protocols in the Internet. ARP is used on Ethernets and other types of networks to find the link-layer MAC address of a node on the same subnet as the sender. A node sending a packet to a local IP address for which it does not yet have the MAC address cached, broadcasts an ARP REQUEST packet on the local subnet link, giving the IP address of the node it is looking for; that node responds with an ARP REPLY packet, giving its MAC address, and all other nodes ignore the REQUEST.[12] If all

nodes in an ad hoc network are within wireless transmission range of each other, this is the only routing protocol needed for the ad hoc network. Our original implementation of PSWAN in 1997 also was structured as an extension of ARP, integrated into the existing ARP implementation in the FreeBSD Unix kernel [FreeBSD], using an extension of the ARP REQUEST and ARP REPLY packet formats; as described in Sections 3.8 and 4.2, however, we ultimately decided to operate DSR at the network layer rather than at the link layer, to allow routing between different heterogeneous networks all forming a single ad hoc network. CRWAN is also similar in approach to the source routing discovery mechanism used in the IEEE 802 SRT bridge standard , and related mechanisms have also been used in other systems including FLIP [Kaashoek 1993] and SDRP [Estrin 1995]. However, in wired networks, a bridge can copy such an explorer packet from one network interface onto each of its other interfaces (i.e., to each other link to which this bridge is attached) and be sure that the explorer packet will flood the network in an orderly and complete way.

6. CONCLUSION

The Parallel Simulation of Wireless Adhoc Networks (PSWANs) is a simple and well-organized provides excellent performance for routing in multi-hop wireless ad hoc networks. As shown in our detailed simulation studies and in our implementation of the protocol in a real ad hoc network of cars driving and routing among themselves, PSWANs has very low routing overhead and is able to correctly deliver almost all originated data packets, even with continuous, rapid motion of all nodes in the network.

For example, PSWAN does not use any periodic routing advertisement, link status sensing, or neighbor detection packets, and does not rely on these functions from any underlying protocols in the network. This entirely on-demand behavior and lack of periodic activity allows the number of routing overhead packets caused by DSR to scale all the way down to *zero*, when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packet overhead of PSWAN *automatically* scales to only that needed to track the routes currently in use.

7. REFERENCES

[1]. [Bantz 1994] David F. Bantz and Fr´ed´eric J. Bauchot. Wireless LAN Design Alternatives. *IEEE Network*, 8(2):43–53, March/April 1994.

[2] [Bharghavan 1994] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: Media Access Protocol forWireless LAN's. In *Proceedings of the ACMSIGCOMM'94nConference*, pages 212–225. ACM, August 1994. [Braden 1989] Robert T. Braden, editor. Requirements for Internet Hosts—Communication Layers.

[3] [Broch 1998] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols.

[4] In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom'98), pages 85–97, Dallas, TX, October 1998. ACM.

[5] [Broch 1999b] Josh Broch, David A. Maltz, and David B. Johnson. Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks. In *Proceedings of The International Symposium on Parallel Architectures, Algorithms and Networks*.

[6] [Cheshire 1996] Stuart Cheshire and Mary Baker. Internet Mobility 4x4. In *Proceedings of the SIGCOMM '96*, pages 318–329. ACM, August 1996. [Deering 1998] Stephen E. Deering and Robert M. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[7] [Droms 1997] Ralph Droms. Dynamic Host Configuration Protocol. RFC 2131, March 1997. [Dube 1997] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi. Signal Stability-Based Adaptive Routing (SSA) for Ad HocMobile Networks. *IEEE Personal [7] Communications*, 4(1):36–45, February 1997. [Estrin 1995] Deborah Estrin, Daniel Zappala, Tony Li, Yakov Rekhter, and Kannan Varadhan. Source Demand Routing: Packet Format and Forwarding Specification (Version 1). Internet-Draft, January 1995. Work in progress. [Fall 1997] Kevin Fall and Kannan Varadhan, editors. *ns* Notes and Documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. Available from http://www-mash.cs.berkeley.edu/ns/.[Frank 1988] Daniel M. Frank. Transmission of IP Datagrams over NET/ROM Networks. In *ARRL Amateur Radio 7th Computer Networking conference*, pages 65–70. American Radio Relay League, October 1988.

[8] [FreeBSD] FreeBSD Project. FreeBSD Home Page. Available at http://www.freebsd.org/. [Garbee 1987] Bdale Garbee. Thoughts on the Issues of Address Resolution and Routing in Amateur Packet Radio TCP/IP Networks. In *ARRL Amateur Radio 6th Computer Networking [9] Conference*, pages 56–58. American Radio Relay League, August 1987.

AUTHOR BIOGRAPHIES

Raman Chadha having 13 years' experience in teaching and 1 year experience in an industry. His Area of Research includes Data Structures, Programming in 'C', C++ , Artificial Intelligence and Mobile Ad-hoc Networks. Presently working as an Astt. Professor(HOD, IT, MCA) in Vidya College of Engineering, Meerut. He has authored more than 8 books. He is pursuing PhD from Monad University, Village & Post Kastla Kasmabad, Pilakhwa- Tehsil Hapur (U.P), under the guidance of Dr. Satyadev Garg (I.T. Consultant, TCS, Noida). His area of Research is "**PARALLEL SIMULATION OF LARGE-SCALE WIRELESS NETWORKS".**