

# Ad Hoc Bridging Protocol

Kartik Gopalan

Navodaya Garepalli

Ping Yang

Florida State University Florida State University Binghamton University

kartik@cs.fsu.edu

garepall@cs.fsu.edu

pyang@cs.binghamton.edu

**Abstract**—This paper introduces a new and simple wireless ad hoc routing protocol called *Ad hoc Bridging Protocol* (ABP). ABP distinguishes itself from earlier proposals in that it uses no explicit control messages for route discovery, setup, and maintenance, while making minimal use of implicit (data-like) control messages that require no special processing at intermediate nodes. ABP is inspired from the self-learning plug-and-play nature of traditional transparent bridges in wired LANs. Like wired bridges and many earlier ad hoc protocols, nodes in ABP learn and adapt routes to different destinations on-the-fly by observing the packets sent by those destinations. As a result, ABP is completely decentralized, self-starting, and automatically adapts to the widely varying network conditions. However, a major difference is that ABP does not attempt to prevent routing loops at all costs. Consequently, ABP does not construct or maintain any overarching control infrastructure for loop prevention, such as extensive network-wide spanning trees in traditional LAN bridges, destination sequence numbers in AODV, or source-routing in DSR. The essential idea is to couple the backward learning technique from transparent bridging, with an associated refresh mechanism, and the use of identification field to effectively limit the impact of any transient loops. We also prove that transient loops, if and when they occur, do not last longer than a bounded short time interval and do not negatively impact the network performance. Simulation results demonstrate that, in spite of its simplicity, ABP can deliver quite competitive network performance in comparison to AODV and DSR protocols while significantly reducing the protocol complexity.

## I. INTRODUCTION

State-of-the-art wireless ad hoc routing protocols rely upon the use of explicit control messages (possibly piggybacked) with associated extensive control infrastructure to set up and maintain loop-free routes. While the on-demand nature of these protocols and several optimizations contribute to minimize control overhead, the fact remains that a large fraction of protocol complexity itself arises from the need to maintain this control infrastructure and process the explicit control information.

We propose a new wireless ad hoc routing protocol, called *Ad hoc Bridging Protocol* (ABP), a simple and effective mechanism to perform ad hoc routing that neither uses any *explicit* control messages nor maintains any control infrastructure in an attempt to prevent routing loops. ABP does not aim to optimize for the network throughput. In fact, sometimes the use of explicit control messages can lead to better network throughput. Rather, this paper demonstrates that there is another as-yet-overlooked ad hoc routing technique that greatly reduces protocol complexity by not relying upon any overarching control infrastructure. What's more, our results show that ABP delivers very competitive network performance, if not better, in comparison to the popular AODV [1] and DSR [2] protocols.

The key idea behind ABP is to combine the on-demand nature of ad hoc routing protocols with the simplicity of *backward learning* that has long been relied upon in wired-LAN bridges. Specifically, if a node  $X$  receives a packet that originated at  $D$  from an immediate neighbor  $Y$ , then  $X$  can infer that  $D$  must be reachable along the direction of  $Y$ . Subsequently,  $X$  can

forward packets destined for  $D$  to its next hop neighbor  $Y$ . If no information about a destination  $D$  is available, then  $X$  resorts to flooding the packets destined to  $D$ . This forwarding process is completely transparent to end-nodes because no explicit route setup or maintenance messages are exchanged. The primary source of overhead is the need to construct and maintain a network-wide *spanning tree* infrastructure so as to prevent the formation of routing loops.

Just as in wired bridging and many wireless ad hoc protocols, nodes in ABP use backward learning to set up and adapt routes to different destinations on-the-fly by observing the packets sent by those destinations. However, unlike conventional wireless ad hoc protocols, ABP nodes implicitly learn routes from data packets themselves and avoid using explicit control messages. In the case of unidirectional traffic flow, ABP selectively uses minimal number of *implicit* (or data-like) control messages that do not require any special processing in the network interior. At the same time, ABP also does not go out of its way to prevent routing loops at all cost but limits their impact by design, if and when such loops arise. Hence ABP eliminates the overhead of constructing and maintaining an extensive infrastructure for loop prevention (such as network-wide spanning trees in wired-LAN bridges, destination sequence numbers in AODV, or source-routing in DSR). We prove that any transient loops, which may potentially arise, do not last beyond a bounded small duration. Moreover, packets do not traverse around transient loops more than once. Additional features of ABP include no need for promiscuous mode operation, no *Hello* messages between neighbors, and elimination of ARP (Address Resolution Protocol) request/reply messages. Thus, ABP is completely decentralized, self-starting, and automatically adapts to the widely varying network conditions.

## II. NETWORK MODEL

We consider a collection of possibly mobile nodes operating together in an ad hoc network without any infrastructure support. ABP nodes have two levels of identity. First is the link level (or layer-2) address which is visible only to immediate neighbors of the node. Second is the network level (or layer-3) address which a node uses to identify and establish communication sessions with any other node (not necessarily a neighbor). Without loss of generality, in this paper we assume that layer-2 address is the 6-byte 802.11 MAC address and the layer-3 address is the 4-byte IPv4 address (or IP address). ABP nodes do not assume any form of hierarchical or structured relationship among the different layer-3 or layer-2 addresses of nodes in the ad hoc network. All nodes in the network can participate in ABP. A node does not need to have any prior knowledge of the identity, connectivity,

or location, of other nodes in the network except, of course, the layer-3 IP address of the nodes with which it explicitly wishes to communicate. A node dynamically learns and unlearns the layer-2 MAC address identity of its neighbors only when it needs to communicate with/via those nodes. Nodes do not exchange any periodic Hello messages to maintain neighbor information. The backward learning mechanism assumes a bidirectional ability to communicate among immediate neighbors. Hence ABP operates on top of link layer protocols, such as the 802.11 family of protocols, which support link-level acknowledgment mechanism for packet transmission between neighbors. Link-level acknowledgments help ABP nodes to detect and avoid unidirectional links, which may occur at times due to differing antenna or propagation patterns. The only capabilities required from the network interface hardware is the ability to send/receive unicast packets (with link-level acknowledgments) and layer-2 broadcast packets (without link-level acknowledgments). Promiscuous mode operation of network interface is not required in ABP.

### III. ABP OVERVIEW

ABP is a completely on-demand ad hoc routing protocol, in which nodes dynamically learn routes by applying the concept of *backward learning* from data packets in transit. One important consequence of not using any network-wide control infrastructure or explicit control messages is that the ABP operation at each node is fully independent. Every node makes completely local routing decisions that require no co-ordination whatsoever with other nodes. ABP is positioned as a thin protocol layer in the networking stack between the Layer-3 (IP layer) and the Layer-2 (MAC layer). Although forwarding is based on destination IP address, the ABP routing is slightly different from traditional IP routing. As we will detail in Section IV, the next-hop address for a destination IP in the routing table is the layer-2 MAC address of the next-hop rather than its IP address. ABP nodes do not care about the IP-to-MAC address mappings of other nodes, except as needed to make forwarding decisions.

In the absence of any control infrastructure for loop prevention, ABP does not try to avoid loop formation at all times. Rather the ABP nodes utilize the Identification and time-to-live (TTL) fields carried within each packet's IP header to contain the impact of any potential loops that may arise. (In non-IP networks, packets that do not carry these pieces of embedded information can be handled by encapsulating them within a small header that carries a TTL and a unique identification value.) Even if transient routing loops emerge under some situations, we prove that they last for a bounded small duration. Similar mechanisms to get around the restrictions imposed by spanning tree in wired-LAN bridges has been advocated by Perlman [3].

### IV. ROUTING TABLE AND DATA PACKETS

Each ABP node maintains its forwarding knowledge in a *soft-state* routing table. Each routing table entry contains the following basic information: (1) Destination IP Address ( $D$ ), (2) Next Hop MAC Address ( $m$ ), (3) A list of alternative next hop MAC addresses, (4) Valid flag ( $v$ ), and (5) Expiration time ( $e$ ). Only routing entries that are *valid* can be used for packet forwarding.

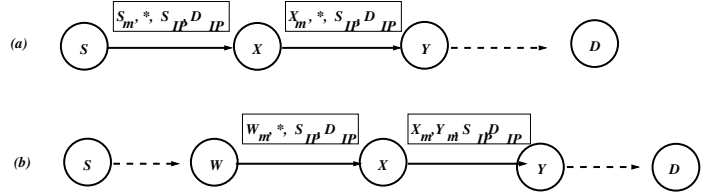


Fig. 1. Illustration of route discovery mechanism.

Routing entries can be invalid during the times when the network is recovering after a failure or mobility of a neighbor.

A routing entry gets *refreshed* each time a packet from the corresponding destination  $D$  is received from the next-hop MAC address  $m$ . In other words, reverse traffic from the destination  $D$  updates its forward routing entries at intermediate nodes. Expiration Time field indicates the amount of time left before an *un-refreshed* routing entry can be purged.

We represent the relevant IP header fields of a packet by the quadruple  $[s_m, d_m, S_{IP}, D_{IP}]$ . Here  $s_m$  and  $d_m$  represent the source and destination MAC addresses of immediate neighbors exchanging the packet and  $S_{IP}$  and  $D_{IP}$  represent the original source and final destination IP addresses. A  $*$  represents a broadcast MAC or IP address.

### V. ROUTE DISCOVERY

The route discovery process in ABP has similarities to that in AODV protocol which also employs backward learning. However, AODV performs learning using explicit control messages, such as RREQ and RREP, rather than using data packets. Figure 1 illustrates the basic route setup mechanism in ABP. Source  $S$  initiates communication with destination  $D$  for the first time by flooding its first data packet with header  $[S_m^*, *, S_{IP}, D_{IP}]$ . This data packet travels towards  $D$ , setting up routing table entries for  $S$  at intermediate nodes. Any intermediate node  $X$ , that receives the packet from  $S$ , processes the packet in two phases: the *learning* phase and the *forwarding* phase.

Note here that there is an inherent tradeoff between flooding a first data packet (as in ABP) versus flooding a first control packet (as in AODV and DSR). On one hand, flooding a (potentially large) data packet could be more expensive than flooding a (smaller) control packet. On the other hand, by eliminating the extensive loop prevention infrastructure and explicit control messages, ABP greatly simplifies the protocol complexity. Additionally, in practice, many communication sessions begin with an initial bidirectional exchange of small packets (such as in TCP) which further reduces the flooding cost during route setup. Finally, ABP significantly limits the impact of network-wide broadcasts via mechanisms described later in Section V-C and V-D.

#### A. Learning Phase

In the learning phase, node  $X$  learns about the direction in which node  $S$  is reachable. First consider the case in Figure 1(a) where  $X$  is an immediate neighbor of  $S$ , but is not aware of this fact and does not have any prior routing information for  $S$  in its routing table. When  $X$  receives the first broadcast packet with header  $[S_m^*, *, S_{IP}, D_{IP}]$ , it immediately infers that a node

with layer-3 IP address  $S_{IP}$  can be reached in the direction of an immediate neighbor with layer-2 MAC address  $S_m$ . The node  $X$  then enters this newly learnt forwarding information in its routing table as a valid routing entry for  $S_{IP}$ . Note that  $X$  does not care that  $S_{IP}$  and  $S_m$  are the same neighbor's IP and MAC addresses respectively. All  $X$  cares is that  $S_{IP}$  is reachable in the direction of  $S_m$ .

Consider the second case in Figure 1(b) when  $X$  is not an immediate neighbor of  $S$  and does not have any prior routing entry for  $S_{IP}$ . Rather  $X$  receives a packet with header  $[W_m, *, S_{IP}, D_{IP}]$ , forwarded from another neighbor  $W$ , which does not know the next hop to  $D$ . Here  $X$  will learn that  $S_{IP}$  is reachable in the direction of  $W_m$ .

Finally consider the third case, again in Figure 1(b), when node  $X$  already has a valid routing table entry for  $S$ . Assume the packet with header  $[W_m, *, S_{IP}, D_{IP}]$  is received for the first time from a neighbor  $W$ , which  $X$  already knows is a valid next hop for  $S_{IP}$ . In this case the only new information that node  $X$  can learn from the packet is that  $S_{IP}$  is still reachable along the direction of  $W_m$  and can refresh its routing table entry for  $S$  by resetting the expiration time field. On the other hand, if the current next hop for  $S_{IP}$  is not  $W_m$  then  $X$  can infer that either the earlier known route to  $S_{IP}$  has failed, or  $S$  has moved to a new location, or simply a better route to  $S$  may have appeared. In this case,  $X$  *re-learns* the new next-hop to  $S_{IP}$ , as described later in Section VI.

Another function of the learning phase is to learn alternative routes. Suppose an intermediate node  $X$  receives multiple copies of the same packet, which originated from  $S$ , from different neighbors. The first copy of the packet will be used by  $X$  to learn the *primary* next hop towards  $S$ . Any subsequent copy of the packet will be recognized by  $X$  as a duplicate (using mechanism is described later in Section VII). Before dropping the duplicates,  $X$  will extract the source MAC address from the duplicates as *alternative* next hop towards  $S$  which can be used in case the primary next hop fails.

One consequence of the backward learning mechanism described above is that ABP automatically populates the routing table with the mappings between destination IP address and next-hop MAC address. *Hence ABP eliminates the need to separately invoke the Address Resolution Protocol (ARP) to resolve these mappings*, which would otherwise introduce additional network traffic in the form of ARP Request/Reply messages. Another consequence is that neighbors can learn of each others' existence only when they need to forward data packets. There is no particular need for nodes to proactively maintain neighbor membership information using Hello messages.

## B. Forwarding Phase

In the forwarding phase, node  $X$  first checks if the packet is destined to itself, i.e.  $D_{IP} = X_{IP}$ . If so, the packet is delivered to the IP layer (layer-3) within  $X$ . If not, then  $X$  determines the next hop for destination  $D$ . If no valid routing entry exists for  $D_{IP}$  then, as shown in Figure 1(a),  $X$  re-broadcasts the data packet to its neighbors with a modified header  $[X_m, *, S_{IP}, D_{IP}]$ . If  $X$  already has a valid routing en-

try for  $D_{IP}$  with the next hop neighbor  $Y$  then, as shown in Figure 1(b),  $X$  forwards the data packet to  $Y$  with a modified header  $[X_m, Y_m, S_{IP}, D_{IP}]$ . *Hence, if parts of the network already have valid routes to  $D$  then those routes are reused.*

If, for some reason, the transmission to next hop node  $Y$  fails, then  $X$  invalidates the routing entry and attempts to send the data packet to one of its neighbors  $Z$  in the *alternative* next hop list of routing entry for  $D$  with a modified header  $[X_m, Z_m, S_{IP}, D_{IP}]$ . If transmissions to none of the alternative next hops succeed, then  $X$  simply re-broadcasts the packet with a modified header  $[X_m, *, S_{IP}, D_{IP}]$ .

To complete the picture, by the time packet from  $S$  reaches  $D$ , the intermediate nodes in the network (including  $D$ ) learn about the route to reach  $S$ . Similarly, when the receiving user-level application at node  $D$  responds back with its own data to its peer application at node  $S$ , the intermediate nodes (including  $S$ ) learn about how to reach  $D$  as well.

## C. Handling Silent Endpoints

One of the distinguishing features of ABP is the manner in which it handles connections that involve silent endpoints. Many real-world network applications engage in bidirectional communication. For instance, the popular TCP protocol would, at a minimum, involve at least one end transmitting data and the other end responding back with ACKs. On the other hand, there also exist applications where only one of the end-points might actively send traffic. In such cases, the backward learning mechanism described above would maintain routing information for only the active sender and not for the passive receiver. This creates an undesirable situation where all data packets to the silent receiver might be flooded.

To rectify this situation, the ABP layer at the receiver node  $D$  monitors whether packets from  $S$  are being received for some *ACTIVITY\_INTERVAL* duration (say 80% of maximum route expiration time) without the application at  $D$  sending any data back to  $S$ . If so, the ABP layer at  $D$  transmits a ***dummy IP packet*** to  $S_{IP}$  with zero byte data payload. As the dummy IP packet from  $D_{IP}$  to  $S_{IP}$  travels through the network, intermediate nodes learn the routing information for  $D_{IP}$ . Thus future data packets from  $S$  to  $D$  are unicasted rather than flooded through the network. Since the dummy IP packet is not meant for any specific application at  $S$ , it is silently discarded by the IP layer at  $S$ . ABP layer at  $D$  stops sending dummy IP packets once it observes that  $S$  is not sending any more data packets.

The dummy IP packet is an *implicit control message* for the following two reasons. First, the dummy IP packet is treated just like any other data packet and forwarded by the intermediate nodes without any special processing. Secondly, this packet is created only under the exceptional circumstance when one of end-points is passive for a long period.

## D. Quelling Back-to-back Broadcasts

When TCP is used for communication, the session would normally begin with a transport level 3-way handshake, which would help to set up routing information for both source  $S$  and destination  $D$ . However, for non-TCP sessions, the application

initiating the communication at source  $S$  might, in some cases, begin by sending multiple back-to-back packets to the destination  $D$ . In case the intermediate nodes do not have routing information for  $D$ , all the initial back-to-back packets from  $S$  to  $D$  might be flooded through the network. This situation would persist till either application at  $D$  responds with its own data or the ABP layer at  $D$  sends a dummy IP packet to  $S$ . When this situation happens, it may result in undesirable back-to-back network-wide broadcasts for a small duration of time.

To mitigate this situation, we can apply an optimization at the ABP layer of the initiator  $S$ . After sending out the first packet, the ABP layer at  $S$  buffers the remaining back-to-back outgoing packets till a data packet is received from  $D$ . Since node  $D$  is guaranteed to transmit either an application data packet or a dummy IP packet within a bounded `ACTIVITY_INTERVAL`, the space set aside at  $S$  for this buffer can be bounded and small. After receiving a packet from  $D$ ,  $S$  can be certain that routing state is set up in the network for both end-points and  $S$  can now safely send out the remaining buffered packets as well.

## VI. ROUTE MAINTENANCE

### A. Refreshing Soft Routing State Using Reverse Traffic

The route refresh mechanism employed by ABP is particularly different from the refresh mechanisms employed by other protocols such as AODV. Each routing entry in ABP routing table has an associated lifetime, which indicates the time duration after which the a non-refreshed routing entry will be deleted. Assume that a node  $X$ 's routing table contains a valid routing entry for a destination address  $D_{IP}$  with next hop as node  $Y$ . The lifetime field of this routing entry is refreshed to a maximum lifetime value of `MAX_ROUTE_LIFETIME` each time a packet *originating* from node  $D$  is forwarded by node  $Y$  to node  $X$ . In other words,  $X$  can reaffirm the fact that  $D$  is still reachable along the direction of its neighbor  $Y$  when it receives a packet from  $Y$  that has a *source* IP address of  $D_{IP}$ .

*The unique and important aspect of the above route refresh mechanism in ABP is that the **reverse** traffic originating from  $D$  is used to refresh the **forward** routing information towards  $D$  in the network.* This in contrast to other routing protocols such as AODV in which packets *destined* towards  $D$  are used to refresh the routing information for  $D$ . ABP's refresh mechanism maintains more accurate routing state because only the reverse traffic generated by node  $D$  itself can accurately indicate the current forward direction in which  $D$  can be reached (assuming the use of bidirectional communication links). On the other hand, with the route refresh mechanism in AODV, just because packets destined towards  $D$  are being forwarded using  $D$ 's current routing entry does not necessarily mean that the current routing entry is accurate or should be refreshed. This is especially true during node mobility or failures when only the packets originating from  $D$ , rather than packets destined towards  $D$ , can provide accurate information about  $D$ 's reachability.

### B. Handling Mobility and Network Failures

Ongoing communication sessions can be adversely affected whenever (1) a node fails, (2) a communication link degrades, or (3) nodes move inducing communication failure. Failure can be detected whenever a node attempts to transmit a unicast packet to one of its neighbors and, in return, does not receive a corresponding link-level acknowledgment. Consider Figure 2 in which there is a failure along a route from  $S$  to  $D$  between neighbors  $X$  and  $Y$ . When  $X$  detects a failure during unicast transmission of a packet to its neighbor  $Y$ , it first checks every routing entry that contains  $Y_m$  as the next hop address. If such a routing entry contains an alternative next hop, then the alternative is assigned as the primary next hop. If there is no alternative next hop, the corresponding routing entry is marked as invalid. For example, in Figure 2  $X$  has four other neighbors  $L$ ,  $M$ ,  $N$  and  $O$  besides  $Y$ . Of these four,  $L$  happens to be an alternative next hop towards destination  $D$ . Upon failure of communication with  $Y$ ,  $X$  first assigns  $L$  as the primary next hop and unicasts the packet for  $D$  to  $L$ . This is shown in case (a). If the communication with  $L$  fails as well, then there are no more alternative next hops and  $X$  floods the packet to all its neighbors hoping that at least one of them can reach  $D$ . This is shown in case (b). The scope of this flood is naturally limited by the value of TTL present in the packet at the time it was received by  $X$ .

Part of the network which is affected by the failure (including node  $X$ ) needs to quickly re-learn about new routes to the destination  $D$ . In case  $D$  is still alive and reachable,  $D$  is guaranteed to either transmit data packets or dummy IP packets towards all nodes with whom it is communicating (including  $S$ ). Two possibilities follow. First is that node  $X$  lies along the path of these packets being sent by  $D$  after the failure. In fact,  $X$  may receive packets from  $D$  via any of its neighbors  $L$ ,  $M$ ,  $N$ ,  $O$ , or  $Y$ , and needs to select one as the next hop for  $D$ . However  $X$  needs to be careful in selecting a stable next hop to  $D$  because  $D$  might be moving. To learn a stable route,  $X$  observes all non-duplicate packets from  $D$  that arrive via each neighbor for a duration of `MAX_ROUTE_LIFETIME` after the failure. At the end of this duration,  $X$  selects that neighbor as the next hop from which it receives the latest non-duplicate packet sent by  $D$ . The rationale for this choice is that the latest packet received is more likely to accurately reflect the current direction in which  $D$  is reachable. The second possibility is that  $X$  may not receive any of the packets sent by  $D$  after a failure. Consequently,  $D$ 's routing entry at node  $X$  will expire within `MAX_ROUTE_LIFETIME` duration. This would mean that either  $D$  is unreachable or alternate routes have been found that bypass  $X$ . In either case, routing entry for  $D$  at  $X$  would no longer be required.

Note that in other protocols, such as DSR and AODV, if a link failure occurs and there is no alternative route, then a Route Error (RERR) message is sent back to the source. While, this approach has its benefits, it also requires intermediate nodes to recognize and process explicit RERR messages, contributing to greater protocol complexity. Hence there is an inherent tradeoff between the level of protocol complexity and control message processing.

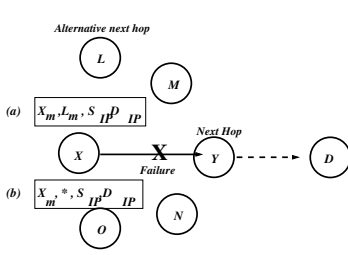


Fig. 2. Two cases when the communication link between  $X$  and  $Y$  fails.

**Re-Learning Even Without Failures:** In Figure 2,  $X$  can relearn a better route to  $D$  even in the absence of failure with  $Y$  ( $X$ 's valid next hop to  $D$ ). For instance,  $X$  might receive a non-duplicate first-time packet from  $D$  via another neighbor  $M$  and not via  $Y$ . At this point, node  $X$  invalidates its current route to  $D$  via  $Y$ .  $X$  then relearns a better route to  $D$  by observing all non-duplicate first-time packets from  $D$  that arrive via its neighbors for  $\text{MAX\_ROUTE\_LIFETIME}$ .

### C. Soft State Re-learning after Node Reboot

A failure or reboot of a ABP node  $Y$  may induce neighboring nodes to re-learn their routes that passed through  $Y$ . However, once  $Y$  reboots and comes back up, it does not need any re-synchronization with its neighbors. Rather it can immediately start learning and participating in the routing process again. This is in contrast to AODV in which a node is forced to stay silent for a period of time after reboot in order to flush the routing state of its neighbors [4].

## VII. DAMPING DUPLICATES WITHOUT A SPANNING TREE

Wired-LAN bridges construct and maintain a network-wide spanning tree to prevent routing loops because the Ethernet header does not carry any hop-count or TTL information [3]. A spanning-tree is neither feasible nor advisable in wireless ad hoc networks due to the higher degree of node mobility and non-determinism in transmissions. In fact, spanning trees are being viewed unfavorably even in wired environments [3,5].

ABP nodes do not construct or maintain any spanning trees. Instead they damp the propagation of duplicate packets by relying on the Identification and TTL fields carried inside the IP header of every packet. A simple form of duplicate damping mechanism could be constructed as follows. Corresponding to each IP address  $S_{IP}$  in its routing table, a ABP node  $X$  can remember the largest Identification value that it has seen in packets with  $S_{IP}$  in their source IP field. A duplicate copy of a packet that arrives at node  $X$  will carry an Identification value that is less than or equal to the one already recorded by the node (after accounting for wrapping around of the 16-bit Identification value). Such a packet can potentially be discarded.

However, the above simple scheme might also mistakenly discard re-ordered packets. A very simple technique to mitigate this situation is that every ABP node remembers the Identification values of last  $k$  packets seen from source node  $S$ . A packet is dropped only if its value is among the last  $k$  remembered val-

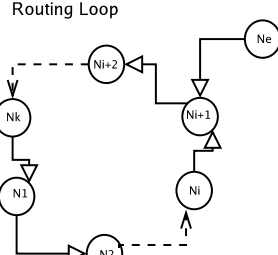


Fig. 3. Complete transient loops disappear in bounded time.

ues or is smaller than all of them. This technique tolerates re-ordering of packets within a window of  $k$  packets. In evaluations, we observe that a window size as small as  $k = 3$  is more than sufficient to detect and damp all duplicates even in the presence of packet reordering. The worst-case per-node state requirement in an  $N$ -node network would be  $O(kN)$  (IP Identification field being unique across all connections within a host). Another alternative for duplicate detection with low state overhead could be the use of Bloom Filters [6]. Also note that, unlike the intricate mechanism of destination sequence numbers in AODV, which are used to prevent routing loops altogether and maintain route freshness, the IP Identification field is used in ABP to simply detect and damp duplicate packets.

## VIII. TRANSIENT ROUTING LOOPS IN ABP

One side-effect of not relying on a spanning tree is that transient routing loops may arise for some destinations when network topology is changing. We define a *complete* transient loop for a destination  $D$  as one in which each node in the loop has a valid routing entry towards the next hop node in the loop. On the other hand, a *partial* transient loop for a destination  $D$  is one in which at least one node in the loop does not have a valid routing entry for  $D$ . Consequently, such a node would broadcast the packets destined to  $D$  and the next node in the loop picks up the broadcast packet.

### A. How Do Complete Transient Loops Form?

Throughout our simulations of ABP, we have not yet observed any instance where complete transient loops have actually formed. However, under certain precise timing synchronizations and traffic assumptions, it is theoretically possible to construct scenarios where complete routing loops can potentially form in ABP. For example, consider the scenario given in Figure 4, which shows a wireless ad hoc network that may have many nodes. Our focus is specifically on 9 nodes:  $D$ ,  $N1$ – $N6$ ,  $X$ , and  $Y$ . In the following discussion, we will show how a complete transient routing loop for node  $D$  can arise among nodes  $N1$ – $N4$ . Let's start with the scenario in Figure 4–(a). Assume that a routing entry for destination node  $N6$  exists at node  $X$  with next hop as  $N1$ . Let's represent this routing entry by the notation  $X \xrightarrow{N6} N1$ . Similarly there are three other routing entries, namely,  $N1 \xrightarrow{N6} N2$ ,  $N2 \xrightarrow{N6} N4$ ,  $Y \xrightarrow{N5} N4$ ,  $N4 \xrightarrow{N5} N3$ , and  $N3 \xrightarrow{N5} N1$ . There may be other routing entries at different nodes, which we ignore for our present discussion. Now assume that, at time  $t_1$ , a packet  $P_1$  originating from  $D$  and destined to  $N6$  is forwarded by  $X$  to  $N1$ . Due to backward learning,  $N1$  learns that  $D$  is reachable from  $X$  and inserts a new routing entry  $N1 \xrightarrow{D} X$  in its routing table. Assume that, at the same time  $t_1$ , another packet  $P_2$  originating from  $D$  and destined to  $N5$  is forwarded by  $Y$  to  $N4$ . Thus  $N4$  applies backward learning and inserts a new routing entry  $N4 \xrightarrow{D} X$  in its routing table. The resulting state is shown in Figure 4–(b). Next, node  $N1$  forwards packet  $P_1$  to  $N2$  at time  $t_2$ . Now  $N2$  applies backward learning and inserts a new routing entry  $N2 \xrightarrow{D} N1$  in its routing

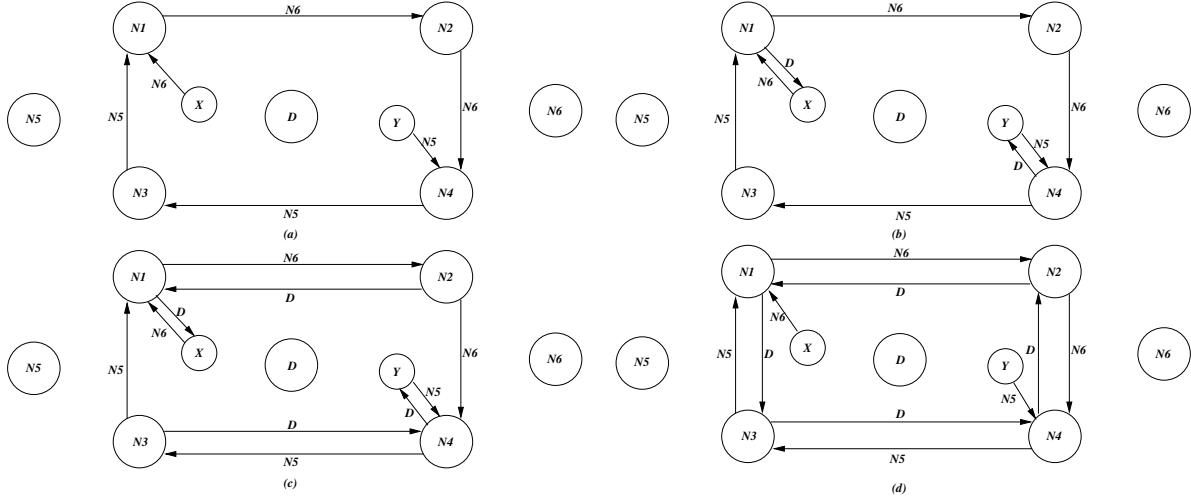


Fig. 4. Example of a loop formation scenario under certain timing synchronization and traffic constraints.

table. Similarly, N4 forwards packet  $P_2$  to N3 at the same time  $t_2$  and N3 applies backward learning to insert a new routing entry  $N3 \xrightarrow{D} N4$  in its routing table. The resulting state is shown in Figure 4-(c). Finally, N2 forwards packet  $P_1$  to N4 at time  $t_3$ . Now N4 applies backward learning and re-learns a new route to D via N2, i.e. it replaces the existing routing entry for D with a new entry  $N4 \xrightarrow{D} N2$  in its routing table. Similarly, N3 forwards packet  $P_2$  to N1 at the same time  $t_3$  and N1 applies backward learning to re-learn a new route to D via N3, i.e. it replaces the existing routing entry for D with a new entry  $N1 \xrightarrow{D} N3$  in its routing table. The resulting network state is shown in Figure 4-(d) in which a complete transient routing loop towards D is created as follows:  $N1 \rightarrow N3 \rightarrow N4 \rightarrow N2 \rightarrow N1$ .

Note that, in the above example, certain timing synchronization in arrival times of packets  $P_1$  and  $P_2$  at different nodes plays a key role in the formation of the complete routing loop. Such timing synchronizations and resulting loops have not yet been observed in our simulations and appear highly unlikely to occur in practice.

### B. Limiting the Impact of Transient Loops

Algorithms such as AODV and DSR incorporate intricate mechanisms (based on destination sequence numbers or source routes) to guarantee that *complete* transient loops are never formed. On the other hand, *partial* transient loops are almost impossible to avoid whenever any intermediate node needs to broadcast a packet to an unknown destination. ABP does not attempt to prevent transient loops at all costs. Rather, if and when transient loops are indeed formed, their impact on network performance is highly minimized due to following three factors.

1. ABP guarantees that complete transient loops last for less than a provably short bounded time interval (We prove this claim below).
2. Since ABP nodes perform duplicate packet detection (Section VII), no packet will traverse a loop more than once. Rather a packet is discarded once it revisits any node.
3. Successive ABP nodes decrement the TTL field in each packet and discard them when TTL reaches zero.

While (2) and (3) guarantee that packets do not proliferate unchecked even when complete or partial transient loops are formed, (1) guarantees that complete transient loops do not last forever,

**Theorem 1:** Lifetime of a complete transient loop for any destination D is less than or equal to MAX\_ROUTE\_LIFETIME.

**Proof:** Consider the complete loop in Figure 3 for destination D with  $k$  nodes  $N_1$  to  $N_k$ , where the next hop for destination D at node  $N_i$  is  $N_{i+1}$  and at node  $N_k$  is  $N_1$ . By definition, node D itself cannot be part of the complete loop and hence is outside the loop. Furthermore, in order for node  $N_i$  to maintain  $N_{i+1}$  as its next hop entry for D,  $N_{i+1}$  must forward a packet with source IP address  $D_{IP}$  to  $N_i$  within MAX\_ROUTE\_LIFETIME interval of its learning the next hop. Two cases arise. In the first case, node D does not send a packet through any node in the complete loop for MAX\_ROUTE\_LIFETIME duration. Thus  $N_{i+1}$  cannot forward a packet with  $D_{IP}$  as source IP address to  $N_i$ . Consequently  $N_i$ 's routing entry for D will expire and become invalid, thus breaking the complete loop within MAX\_ROUTE\_LIFETIME duration. In the second case, node D does send a packet through node  $N_{i+1}$  within MAX\_ROUTE\_LIFETIME interval. Since node D is outside the loop, the packet from D must enter the loop from some external node  $N_e$ . Without loss of generality, assume that  $N_e$  is a neighbor of node  $N_{i+1}$  that we are presently considering. Thus node D's packet arrives at  $N_{i+1}$  from  $N_e$ . However,  $N_e$  is different from  $N_{i+2}$ , which is the current next hop for D at node  $N_{i+1}$  (since  $N_e$  is outside the loop). Thus, as described in Section VI,  $N_{i+1}$  will switch to maintenance mode to re-learn its next hop for D by invalidating its routing entry for D via  $N_{i+2}$ . This too would break the complete loop within MAX\_ROUTE\_LIFETIME interval. ■

It is easy to see from the above argument that a complete loop soon decays into a partial loop in which one or more nodes do not have a valid routing entry towards D and broadcast the packets destined to D. However the partial loop also lasts only till the broadcasting node re-learns a new route to D, which would

happen quickly if the broadcasting node lies along the path of data/dummy-IP packets arriving from  $D$ . Furthermore,  $D$  is *guaranteed* to send either a data or a dummy IP packet within `ACTIVITY_INTERVAL`, as described in Section VI-C.

## IX. PERFORMANCE EVALUATION

One might expect that the performance of a protocol that doesn't rely upon extensive control infrastructure would be lower than one that does. This is because the additional control processing could potentially be used towards increasing the data transfer efficiency. In this section we show that, in spite of its simplicity, the performance of ABP is very competitive, if not better, when compared to AODV and DSR protocols.

We used the discrete event Network Simulator(NS2) [7] to evaluate the performance of ABP protocol. Simulations were run for different networks with 50 nodes. The number of connections were varied from 10 to 100, where each connection is set up between a randomly selected node pair. We used the mobility scenarios from the Monarch Project [8] in which the mobility speed of the nodes was varied between from 0 to 20 meters/sec. Nodes move within a rectangular field of 1500m x 300m according to the Random Waypoint Model. Each simulation is run for 900-simulated seconds. Pause times were varied from 0s (Highly mobile) to 900s (static). We used both TCP-based connections, (for bidirectional communication), and UDP-based Constant Bit Rate (CBR) connections (for unidirectional communication). The maximum TCP window was set to 20 packets, with maximum packet size set to 512 bytes. The UDP CBR sources sent traffic at 4 packets/sec with 64 byte packets. Each data point is averaged over five simulation runs with a different random seed to vary the connection establishment pattern.

ABP does not use any explicit control messages, ARP requests/replies or Hello messages. In the presence of unidirectional data traffic, ABP nodes generate implicit data-like control messages called *dummy* IP packet. In these simulations, we count the dummy IP packets as the control packets for ABP. The `MAX_ROUTE_LIFETIME` is set to 5 seconds and the dummy IP packets are generated by the ABP layer at a node after 4 seconds of silence on the part of the network application. For AODV and DSR, we count the Route Request (RREQ), Route Reply (RREP), Route Error (RERR), ARP requests/replies, and Hello messages as the control overhead. For DSR, we *do not* count the source routing information carried in packet headers towards the control overhead. The control packets are counted on a hop-to-hop basis rather than on end-to-end basis. Thus, if a control packet is forwarded or broadcasted  $k$  times, then it is counted as  $k$  control messages.

For fair comparison, in AODV simulations we enabled the Local Repair optimization (which attempts to repair a failed link locally before returning a RREQ message to source) and the Link Layer Detection optimization (which suppresses the transmission of "Hello" messages). Similarly, the simulation of DSR included the optimization of piggybacking RERR and RREP messages onto RREQ messages and the optimization of populating route cache by listening in promiscuous mode. ABP and AODV simulations do not use promiscuous mode operation.

We focus upon three metrics in our evaluations to bring out the advantages as well as disadvantages of ABP. (1) *R/S* is the ratio of the number of packets received by the destination to the number of packets sent by the source. (2) *C/R* is the ratio of number of control packets to the number of data packets received by destination. It signifies the number of control packets sent for each data packet received. (3) *Number of control packets* represents the total number of control messages transmitted during the entire simulation. We note that, among the three metrics, R/S represents the most accurate measure of protocol performance. This is because R/S incorporates the impact of both control messages (both non-piggybacked and piggybacked) as well as data packets (both unicast and broadcast) on the overall protocol performance. The other two metrics account only for explicit and implicit non-piggybacked control messages.

### A. Performance with TCP Traffic

**R/S:** Figures 5, 6, and 7 show that DSR performs best in terms of R/S values, followed by ABP and AODV. In Figure 5, as the pause time increases, the R/S value approaches 1 (no packet loss). In Figure 6, as the number of connections increases, channel contention and packet losses increase resulting in a drop in R/S. As in the previous case, DSR performs best, followed by ABP and finally AODV. Figure 7 shows a universal decrease in R/S with increase in mobility speed. The difference in R/S between DSR and ABP is less than 1%.

It can be seen from these figures that reducing the protocol complexity in ABP does not automatically translate into better delivery ratio (R/S) when compared against DSR. This is because reduction in control message overhead is offset to some extent by increase in both unicast and broadcast data traffic, and this reflects in the final R/S numbers. However, ABP's delivery ratio is still quite competitive with AODV and DSR.

**C/R:** Figures 8, 9, and 10 show the impact of various parameters on C/R values. In all the three figures, ABP maintains a low value of C/R around 0.2 because it exploits the bidirectional nature of TCP traffic to maintain routing state without exchanging excessive control messages. DSR and AODV show higher value of C/R for smaller pause times than for larger pause times. These figures show that larger pause times, smaller number of connections and lower mobility speed, all contribute to less dynamic network and smaller C/R. ABP experiences little variation in C/R whereas variation in C/R is large for both AODV and DSR. It is notable that DSR, which relies upon source routing, performs better than ABP by a difference in C/R of 0.05 for large pause times. Figure 10 shows that increasing the mobility speed of nodes results in greater increase in C/R for DSR than for ABP and AODV. One possible reason is that DSR employs source routing as opposed to ABP and AODV that use backward learning. Consequently, DSR is unable to benefit from the forward routing information carried in the reverse traffic.

**Number of Control Packets:** Figure 11 shows that number of control messages in ABP is much lower than AODV and DSR. For 0s pause time, ABP generates around 20K control packets, AODV generates around 90K control packets and DSR generates around 120K control packets. Thus, in comparison, ABP gener-

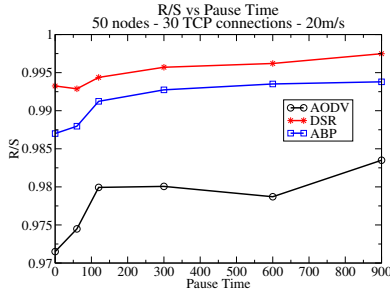


Fig. 5. R/S vs Pause Time for 50 nodes, 30 TCP connections, 20m/s.

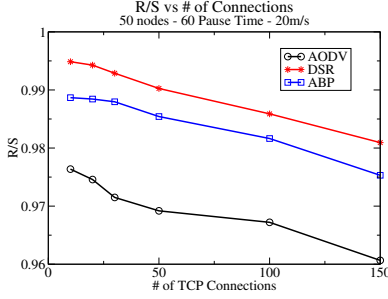


Fig. 6. R/S vs TCP connections for 50 nodes, 60s pause time, 20m/s.

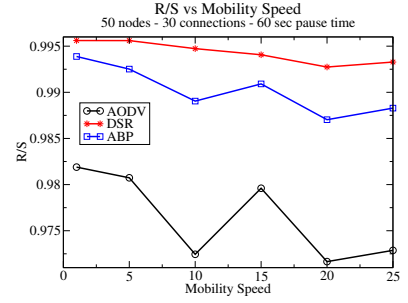


Fig. 7. R/S vs Mobility speed for 50 nodes, 30 TCP connections, 60s pause time.

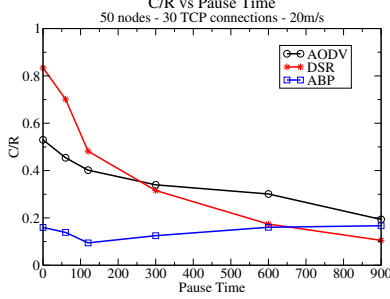


Fig. 8. C/R vs Pause Time for 50 nodes, 30 TCP connections, 20m/s.

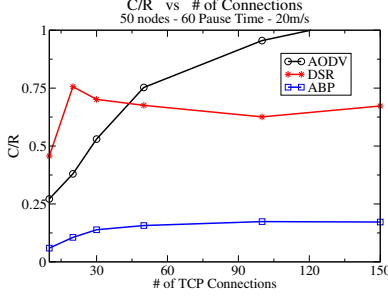


Fig. 9. C/R vs TCP Connections for 50 nodes, 60s pause time, 20m/s.

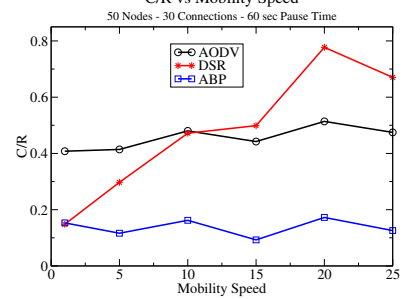


Fig. 10. C/R vs Mobility speed for 50 nodes, 30 TCP connections, 60s pause time.

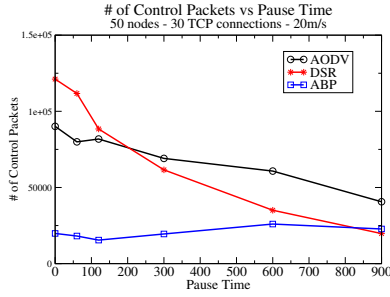


Fig. 11. Control packets vs Pause Time for 50 nodes, 30 TCP connections, 20m/s.

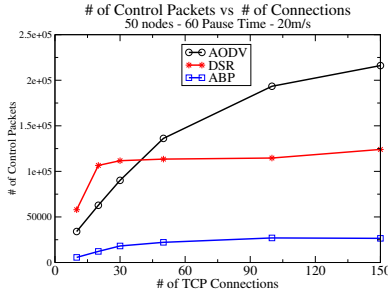


Fig. 12. Control packets vs TCP Connections for 50 nodes, 60s pause time, 20m/s.

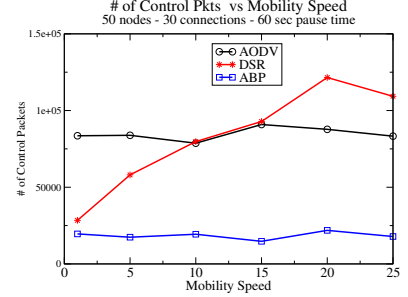


Fig. 13. Control pkts vs Mobility speed. 50 nodes, 30 TCP connections, 60s pause time.

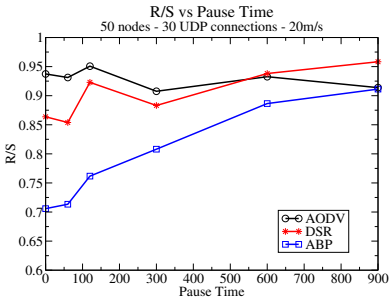


Fig. 14. R/S vs Pause Time for 50 nodes, 30 UDP connections, 20m/s.

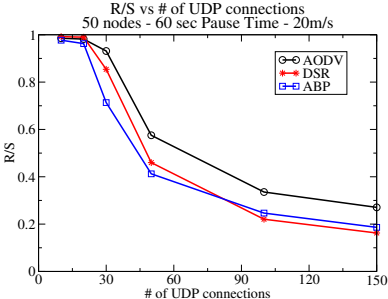


Fig. 15. R/S vs Number of UDP connections for 50 nodes, 60s pause time, 20m/s.

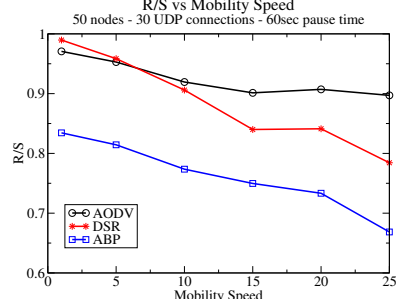


Fig. 16. R/S vs Mobility speed for 50 nodes, 30 UDP connections, 60s pause time.

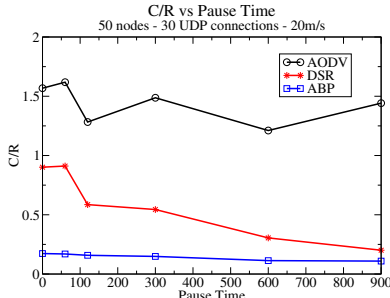


Fig. 17. C/R vs Pause Time for 50 nodes, 30 UDP connections, 20m/s.

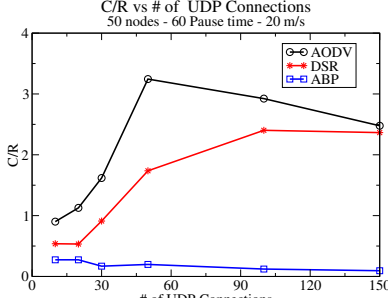


Fig. 18. C/R vs UDP Connections for 50 nodes, 60s pause time, 20m/s.

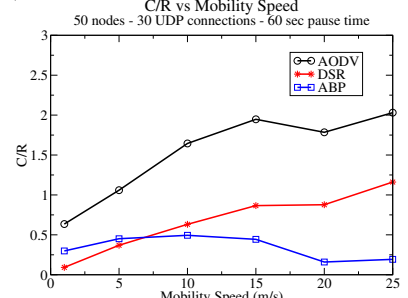


Fig. 19. C/R vs Mobility speed for 50 nodes, 30 UDP connections, 60s pause time.



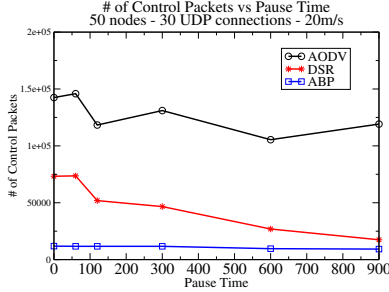


Fig. 20. Control pkts vs Pause Time for 50 nodes, 30 UDP connections, 20m/s.

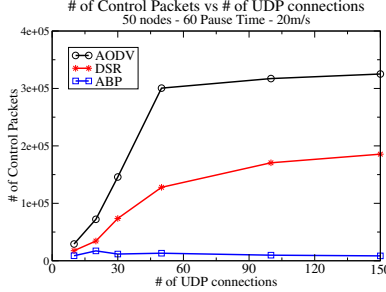


Fig. 21. Control pkts vs UDP Connections for 50 nodes, 60s pause time, 20m/s.

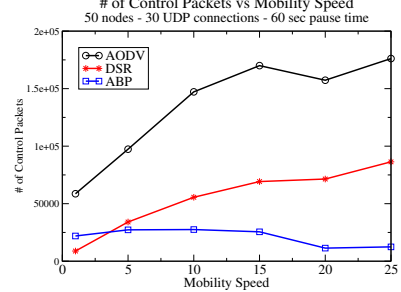


Fig. 22. Control pkts vs Mobility speed. 50 nodes, 30 UDP connections, 60s pause time.

ates 4.5 times fewer control packets than AODV and 6 times fewer control packets than DSR. In Figure 12, we vary the number of TCP connections from 10 to 150, for a fixed 60s pause time and fixed mobility speed of 20m/s. One can see that number of control packets generated by ABP remains much lower than AODV and DSR. As the number of connections increases, AODV generates more control packets than DSR. For 150 connections, AODV generates around 220K control packets, DSR generates around 125K control packets, while ABP generates around 25K control packets. As with C/R, Figure 13 shows that increasing the mobility speed of nodes results in greater increase in number of control packets for DSR than for ABP and AODV.

#### B. Performance with Constant Bit Rate (CBR) UDP Traffic

**R/S:** In Figure 14, ABP achieves an R/S smaller than both AODV and DSR for smaller pause times. As pause time increases, ABP's performance approaches that of AODV. In Figure 15, ABP and DSR achieve almost the same R/S which is smaller than AODV. With increasing mobility speed in Figure 16, AODV tends to achieve higher R/S for UDP traffic compared to both ABP and DSR. When compared against results for TCP traffic, the above results show that, as expected, ABP performs better in the presence of bidirectional traffic, than in the presence unidirectional traffic.

**C/R:** Figure 17 shows that AODV transmits almost 1.5 control packets for each received data packet. Although, DSR has approximately 0.8 C/R at 0s pause time, it reduces as the pause time increases. ABP has a C/R less than 0.2 at any given point of time. Figure 18 shows that, as the number of connections increases, the C/R value drops for ABP. With more number of connections, dummy IP packets are suppressed more often leading to lower C/R value. On the other hand, C/R value increases rapidly for both DSR and AODV. Figure 19 shows similar trends with increasing mobility speed, except that C/R for DSR is slightly smaller than for ABP at very low mobility.

**Control Packets:** In Figure 20, AODV generates around 140K control packets for 0s pause time, whereas DSR generates around 75K control packets and ABP generates around 12K control packets. In Figure 21 we can see that, although all routing protocols use fewer control messages for smaller number of connections, the main difference arises when the number of connections increases. For 150 connections AODV uses around 320K control packets and DSR uses around 180K control pack-

ets, while ABP uses less than 20K control packets. Figure 22 shows trends similar to C/R in varying mobility speed.

#### X. RELATED WORK

Numerous ad hoc routing algorithms have been proposed in literature, all of which rely upon the use of extensive control infrastructure for loop prevention and/or explicit control messages that requires special processing in the network interior. Here we review the ones most relevant to this paper. Existing ad hoc routing protocols can be classified as either *table-driven* or *on-demand* protocols. DSDV [9] is a table-driven protocol based on the distance vector algorithm and provides guarantees on loop freedom by use of destination sequence numbers. DSDV uses explicit control messages to proactively exchange link-state information. A number of other table-driven protocols exist in literature [10–13], all of which inherently require the exchange of explicit control messages to proactively maintain routes. *Ad hoc On-Demand Distance Vector (AODV)* [1] routing protocol is an on-demand version of distance vector algorithm that establishes a route only when required. Like DSDV, AODV guarantees loop freedom through use of destination sequence numbers. AODV relies upon explicit control messages such as RREQ, RREP, and RERR messages to construct and maintain routes. *Dynamic Source Routing (DSR)* [2] is an on-demand routing protocol that uses source routing to guarantee loop freedom. DSR uses explicit control messages, such as RREQ and RREP packets, to accumulate the address of each intermediate node during the route discovery phase. The accumulated source route is then appended to each packet's header before transmission. LQSR [14] is a variant of DSR that supports link quality metrics. A number of other on-demand protocols have been proposed in literature, [15–20] to name a few, all of which use explicit control messages. One class of ad hoc routing protocols [21–23] use the hierarchical structure of addressing to make routing decisions. Another class of routing protocols [24–27] use location information to route data packets.

To the best of our knowledge, all the above protocols use explicit control messages and some form of control infrastructure to construct and maintain loop-free routes in the face of node mobility and communication failures. Some of the above mentioned protocols try to minimize the number of control messages by piggybacking explicit control information with data or other control packets. For instance, DSR can piggyback RERR and

RREP messages over RREQ messages and includes the complete source routing information within every packet. In such cases, intermediate nodes still need to understand and process these protocol-specific explicit control headers and need additional control logic to maintain loop-free routes. In contrast, ABP does not rely upon the use of any explicit control infrastructure or control information, whether piggybacked or otherwise, and yet achieves competitive network performance.

## XI. CONCLUSION

In this paper, we have proposed a simple ad hoc wireless routing protocol, called *Ad hoc Bridging Protocol* (ABP), which is inspired from the self-learning plug-and-play nature of transparent bridges in wired LANs. The distinguishing feature of ABP is that it does not use any explicit control messages and does not rely upon any extensive control infrastructure for loop prevention (such as spanning trees, destination sequence numbers, or source routing mechanisms). ABP is particularly suited for routing bidirectional communication sessions with minimal control overhead. To handle unidirectional communication, ABP makes minimal use of implicit (data-like) control messages that require no special processing at intermediate nodes. We prove that transient loops, if and when they form, do not last longer than a bounded short duration and do not negatively impact network performance. We also demonstrate via extensive simulations that, in spite of its simplicity, ABP's performance is quite competitive compared to AODV and DSR. In summary, we have shown that ABP is a remarkably simple, appealing, and effective routing protocol for ad hoc networks.

## REFERENCES

- [1] C.E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," *IEEE Workshop on Mobile Comp Sys and Applications*, 1999.
- [2] David B Johnson and David A Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, vol. 353. Kluwer Academic Publishers, 1996.
- [3] R. Perlman, "Rbridges: Transparent routing," in *Proc. of Infocom*, 2004.
- [4] C. Perkins, E. Belding-Royer, and S. Das, "Ad-hoc on demand distance vector routing," RFC 3561, July 2003.
- [5] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh, "Viking: A multi-spanning-tree ethernet architecture for metropolitan area and cluster networks," in *Proc. of Infocom 2004*, Hong Kong, China, June 2004.
- [6] Alex C. Snoeren, "Hash-based ip traceback," in *Proc. of the SIGCOMM*, 2001.
- [7] "Network simulator ns-2," <http://www.isi.edu/nsnam/ns/>.
- [8] "The monarch project," <http://www.monarch.cs.rice.edu/>.
- [9] C.Perkins and P.Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM'94*, 1994, pp. 234–244.
- [10] S. Murthy and J.J. Garcia-Luna-Aceves, "A routing protocol for packet radio networks," in *Mobile Computing and Networking*, 1995.
- [11] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-tree routing in wireless networks," in *ICNP*, 1999.
- [12] C.A. Santivanez and R. Ramanathan, "Hazy sighted link state routing protocol," BBN Technical Memorandum No. 1301, Aug. 31 2001.
- [13] T. Parker and K. Langendoen, "Guesswork: Robust routing in an uncertain world," in *Mobile Ad-hoc and Sensor Systems*, 2005.
- [14] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proc. of MobiCom*, New York, NY, USA, 2004.
- [15] V.D. Park and M.S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *INFOCOM*, 1997.
- [16] Z.J. Haas, M.R. Pearlman, and P. Samar, "The interzone routing protocol for ad hoc networks," IETF Internet draft, July 2002.
- [17] Y.J. Lee and G.F. Riley, "Dynamic Nix-Vector routing for mobile ad hoc networks," in *Wireless Commn. and Networking Conf.*, 2005.
- [18] I.D. Chakeres, E.M. Royer, and C.E. Perkins, "Dynamic MANET on-demand routing protocol," IETF Internet Draft, October 2005.
- [19] B.S. Manoj, R. Ananthapadmanabha, and C.S.R. Murthy, "Link life based routing protocol for ad hoc wireless networks," in *ICC*, Oct. 2001.
- [20] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," in *Proc. of SIGCOMM*, New York, NY, USA, 2005.
- [21] T-W. Chen and M. Gerla, "Global state routing: A new routing scheme for ad-hoc wireless networks," in *ICC'98, Atlanta, GA*, June 1998.
- [22] A. Iwata, C. Chiang, G. Pei, M. Gerla, and T. Chen, "Scalable routing strategies for ad-hoc wireless networks," *IEEE JSAC*, vol. 17(8), pp. 1369–1379, Aug. 1999.
- [23] J. Eriksson, M. Faloutsos, and S. Krishnamurthy, "Scalable ad hoc routing: The case for dynamic addressing," in *INFOCOM*, 2004.
- [24] Y.B. Ko and N.H. Vaidya, "Location-aided routing in mobile ad hoc networks," in *MOBICOM'98*, October 1998.
- [25] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris, "A scalable location service for geographic ad-hoc routing," in *Mobicom*, 2000.
- [26] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward, "A distance routing effect algorithm for mobility," in *MobiCom*, 1998.
- [27] B. Karp and H.T.Kung, "Greedy perimeter stateless routing for wireless networks," *MobiComm 2000*, pp. 1–8, 2000.