Virtual Wire for Managing Virtual Dynamic Backbone in Wireless Ad Hoc Networks

Bo Ryu* Jason Erickson† Jim Smallcomb† Son Dao§

*Teledesic, Bellevue, WA
† Raytheon Systems Company, El Segundo, CA
§ HRL Laboratories, Malibu, CA

Abstract

We present a novel distributed algorithm that employs the concept of *virtual wire* for the purpose of achieving reliable and scalable routing in wireless mobile ad hoc networks. Virtual wire allows for a virtual backbone to be dynamically maintained such that it is composed of reliable, relatively stable, and high-bandwidth links even under an environment with highly fluctuating wireless channels. The resulting structure, called *Virtual Dynamic Backbone* (VDB) which resembles a cellular-like infrastructure and is similar to *spine* [3] (but different in its construction and functionality), serves as a common backbone for unicast and multicast routing. This cellular-like structure, combined with virtual wire and efficient link quality estimation algorithm, yields a number of attractive properties such as: (i) simplified unicast and multicast routing, (ii) fast recovery under link quality degradation, (iii) bandwidth-efficient flooding, and (iv) easier QoS management. Most importantly, we achieve all these advantages with a small overhead, making this algorithm highly applicable for bandwidth-scarce wireless environment.

1 Introduction

Highly dynamic wireless ad hoc networks refer to those in which link quality frequently fluctuates due to continuous node mobility and/or jamming. It is a challenging task to develop routing protocols aimed at such networks that are reliable and scalable with small overhead. Recently, interest in developing such ad hoc routing protocols from the networking research community has been growing considerably, yielding various proposals [3, 4, 5, 6, 7, 8, 9, 10, 11] both within and outside the Mobile Ad hoc NETworking (MANET) working group of the Internet Engineering Task Force (IETF). In general, each of these algorithms is designed to address some, but not all, of the following requirements:

- unicast routing
- multicast routing
- QoS routing
- low overhead
- responsiveness (under frequent topology changes)
- reliability (such as link quality-based routing)
- scalability

In this work, we explore a potential solution to achieve *all* of the above requirements by introducing the concept of *virtual wire*, and incorporating it into *Virtual Dynamic Backbone* (VDB). A VDB, which is equivalent to "Spine" [2, 3, 1] in terms of its structure (but different in its creation and maintenance), is a small set of connected mobile nodes such that when formed, each node is either part of, or one-hop away from the VDB. For example, in Fig. 1, the set {3,4,7,8,11,12,13,17} comprises a VDB. The advantages

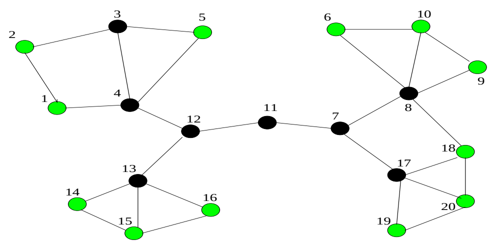

Figure 1: An example of virtual dynamic backbone (VDB).

provided by VDB include:

1. Bandwidth used for flooding (or global broadcast) is minimized;
2. Multicast routing is simplified since the VDB serves as a multicast backbone;
3. QoS routing can be simplified since a VDB is composed of relatively stable nodes and links.

Virtual Wire is defined as a virtual link made up of a stream of very small messages, namely *Virtual Wire Messages* (VWMs), generated and locally broadcast by each and every node. At this point, these VWMs just look like beacon messages used in many ad hoc routing protocols. But the key novelty is that VDB nodes further forward these VWMs, making them *multi-hop* broadcast messages. Duplicates are detected and dropped via sequence number and

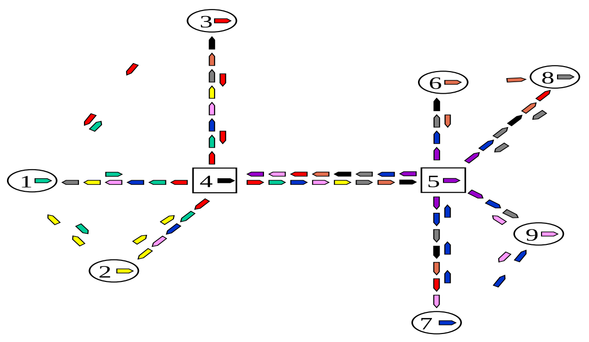

Figure 2: An illustration of *Virtual Wire*. Each and every node generates virtual wire messages (VWMs) at the same rate, say r. Two VDB nodes (nodes 4 and 5) forward received VWMs to their neighbors as local broadcast. This makes VDB nodes transmit VWMs at higher rate (than r) which generally depends on the number of neighbors and the maximum distance each VWM can travel over VDB. In this example, nodes 4 and 5 transmit VWMs at the rate of $8r$. Assuming the size of a VWM being 20 bytes and $r = 0.1$ VWM/sec, the bandwidth overhead per each VDB node (4 and 5 in this example) is only 128 bits/sec. Duplicate VWMs are detected using sequence number checking and dropped so that forwarding is done only once for each VWM.

source address so that forwarding is done only once for each VWM. Non-VDB nodes do not engage in forwarding. Figure 2 illustrates this novel concept. At first, virtual wire appears unacceptably wasteful for bandwidth-scarce environment such as wireless ad hoc networks since it is broadcast intensive. However, note that VDB nodes aggregate VWMs generated/forwarded by its neighbors, resulting in transmitting VWMs at a much higher rate. This allows each node to generate VWMs at a low rate (e.g., one VWM every 10 sec), substantially reducing the overhead caused by VWMs. In addition, we limit the distance (hops) VWMs travel over the VDB (say 3-5 hops) in its design. Consequently, the resulting overhead is considerably lower than it first seems.

Virtual wire brings several key benefits:

1. it significantly simplifies routing since there is no need to exchange routing tables
2. it provides a means to measure and maintain the most up-to-date link quality not only between adjacent nodes but also between those that are more than a single hop away;
3. it is used to create and maintain a stable VDB, i.e., no other messages are needed for managing VDB;
4. it can be easily extended to support asymmetric routing as well; and
5. it makes the VDB look like a cellular network, making node mobility much more manageable.

The last benefit follows from the fact that each node continuously monitors VWMs transmitted by each of its neighbors, which permits early detection of link quality degradation and allows the node to take appropriate actions before the link actually breaks down. In Fig. 2, suppose a non-VDB node 6 moves away from its VDB attachment point (node 5) and towards another VDB node 4. As it gets closer to node 4, it will hear VWMs transmitted by node 4 better than those by node 5. This will allow node 6 to choose node 4 as the new VDB attachment point even if the link quality between node 6 and node 5 remains marginal. We call this *virtual handoff*, following the same notion of handoff in a cellular network.

This paper is organized as follows. Section 2 defines some notations used throughout this paper. Section **??** describes three key components of the distributed algorithm that creates and manages a VDB using VWMs: backbone selection process, backbone connection process, and backbone maintenance process. Section 4 describes an overview of supporting unicast and multicast routing over VDB. Section 5 discusses various issues related to the algorithm such as overhead, routing optimality and scalability, QoS support, and support for asymmetric routing. Finally, Section 6 summarizes the work.

2 Notations and Basic Rules

- At any time, each node is in one of the three colors: *black* (B) when it becomes a part of VDB (VDB node), *green* (G) when it has at least one black node as its neighbor with acceptable link quality, and *white* (W) when it has no black node as its neighbor. Only black nodes forward packets.
- LQ_{ab} represents the link quality from node a to node b observed at b (i.e., outbound link quality from a to b at a, or inbound link quality from a to b at b).
- $\delta(v)$ denotes the number of neighbors of node v with *acceptable* inbound-only or both inbound and outbound link qualities, referred to as *degree*. When v is white, only inbound link quality with its neighbors is taken into account for calculating $\delta(v)$. When green, depending on the color of its neighbors, either inbound-only (for white and green neighbors) or both inbound and outbound (for black neighbors) link qualities must be acceptable to be counted for $\delta(v)$. When black, both link qualities are taken into account except for white neighbors.
- Each node generates Virtual Wire Messages (VWMs) at independent and identically distributed random intervals following truncated exponential distribution. This permits unbiased sampling of link quality.
- A VWM contains the following fields:
  - *Source Node ID ($\mathcal{S}$):* The address of the node which originates this VWM.
  - *Forwarding Node ID ($\mathcal{F}$):* The address of the VDB node by which this message is forwarded.
  - *VAP Node ID ($\mathcal{V}$):* The address of the black node which the forwarding node requests to become or remain as its VAP.
  - *Link Quality Node ID ($\mathcal{N}$):* The address of the node to which link quality feedback is provided by the forwarding node.
  - *Link Quality Feedback (LQ):* Link quality feedback for node $\mathcal{N}$. If $\mathcal{N} = n$ and $\mathcal{F} = f$, this field is taken from LQ_{nf} of the node f's NIT, inbound link quality for neighbor n at f. Once this value

is received by node n, it becomes the outbound link quality for its neighbor f.

- *Hop Count (HC):* The number of hops that this message has been forwarded.
- *Time-To-Live (TTL):* The maximum number of hops that this message can be forwarded.
- *Source Sequence Number:* In combination with the source node ID, this uniquely identifies a VWM per source node.
- *Transmitter Sequence Number:* In combination with the forwarding node ID, this uniquely identifies a VWM per forwarding node.
- *Color (C):* Color of the node *transmitting* this message. (one of White, Green, or Black)
- *Degree (δ):* Degree of the *forwarding* node

- $N_1(v)$ denotes the one-hop neighbors of a node v.
- Let $u \in N_1(v)$. The node v maintains a table called *neighborhood information table* (NIT) which contains its one-hop neighbors ($N_1(v)$), color of each neighbor, $\delta(u)$, inbound and outbound link qualities (LQ), and the expected VWM (transmitter) sequence number from the corresponding neighbor.

  For the purpose of illustration, Figure 3 shows an example of node coloring and two-way link quality. The node 3 will have the NIT as given in Table 1.

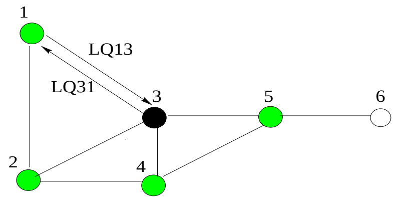

Figure 3: An example of node coloring and two-way link quality notation.

u ($\in N_1(3)$)	Color	$\delta(u)$	LQ_{u3}	LQ_{3u}	Seq. Nr
1	G	2	15	10	31
2	G	3	14	12	327
4	G	3	15	15	15
5	G	3	10	13	88

Table 1: Neighborhood information table (NIT) of node 3 corresponding to Fig. 3. A 16-level link quality ranging from 0 to 15 is assumed.

- Each node (say u), black or green, has at least one black neighbor v via which u is attached to VDB (except the case where VDB consists of a single node). We refer to this black node v as *virtual attachment point* (VAP) of u, or $v = V(u)$.[1] As will be discussed in Section 4.1, VAPs play the role of next-hop routers in an IP network; each node choose the VAP that gives the "best" route to the destination where "best" depends on routing metrics used. As will be discussed in Section 4.1, each node keeps a list of primary gateways as its VAP list (VAP_{list}).

[1] v is also called a dominator of u [3].

- Once a node receives a VWM, it computes its inbound link quality for each of its neighbors [12]. Once a node turns green or black, it maintains outbound link quality with each of its VAPs based on the LQ update provided by its respective VAP.
- Each black node determines whether it needs to remain as a VAP (black) serving its neighbors by maintaining VAP timer T_{VAP}. Whenever a black node detects its address in the VAP Node ID field of the received VWM, it resets T_{VAP} to `VAP_TIMEOUT`. If no request is received until T_{VAP} expires, the black node changes its color to green (after choosing its own VAP from its NIT).

3 Management of Virtual Dynamic Backbone

Because VDB nodes heavily engage in forwarding VWMs, it is desirable to keep the size of VDB as small as possible to conserve bandwidth. In the context of graph theory, this is equivalent to finding a Minimum Connected Dominating Set (MCDS) [2]. Below, we propose a distributed algorithm that creates and maintains a VDB that does not require global (or partially global) exchange of local topology information. This algorithm consists of three components: Backbone Selection Process yielding a dominating set, Backbone Connection Process connecting disjoint VDBs, and Backbone Maintenance Process dealing with node mobility and link quality fluctuation. Simulation results, which will be reported elsewhere, show that this algorithm yields a VDB close to an MCDS in its size.

3.1 Backbone Selection Process (BSP)

In a nutshell, BSP chooses nodes with the maximum degree in their neighborhood as VDB nodes. This process is repeatedly performed at each node until the colors of all the neighbors in its NIT and itself are either green or black. This process can be realized as follows.

When a node turns on its power, it starts building its NIT based on the received VWMs from its neighbors. Each time it receives a VWM, it updates its NIT and resets T_{BSP} if a change in NIT occurs (in terms of color and/or degree). The timer T_{BSP} is used to trigger BSP. As soon as its T_{BSP} expires, it node decides whether it is eligible to be part of VDB by comparing its degree with that of its one-hop neighbors. If its degree is highest, it changes its color to black and starts forwarding VWMs. Minimum ID is used to break tie between neighbors with the highest δ of equal value.

If a white node w detects a black neighbor b (either a new neighbor or color change of an existing neighbor) with $(LQ_{bw}, LQ_{wb}) \geq$ `LQ_THRESHOLD` before T_{BSP} expires, w changes its color to green and starts sending its VWMs with its VAP Node ID set to b, i.e., $\mathcal{V} = b$ since $b = V(w)$. Once it changes to green, w checks if there is at least one white neighbor. If so, it resets T_{BSP} since it may need to support that white node by becoming black. After T_{BSP} expires, if it still has at least one white neighbor and its degree is highest among the remaining neighbors excluding black neighbor(s), it changes to black. We illustrate the BSP in Figure 4.

The BSP algorithm above yields the following crucial property: *for any black node, there exists another black node to which the distance is at most three hops.* In other words, after BSP is complete, a black node can reach another black node with at most three hops (i.e. two green nodes between them). Stated differently, a partially formed VDB can

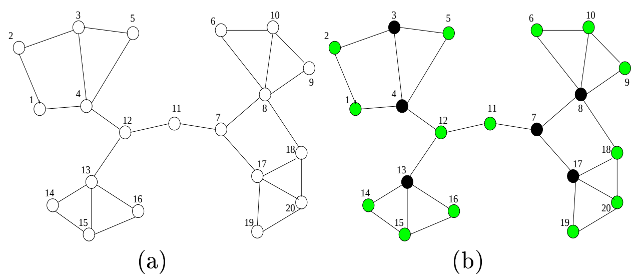

Figure 4: Illustration of Backbone Selection Process. (a) every node is white; (b) after neighborhood discovery and backbone selection processes, nodes 4, 8, 13, and 17 become black in the first round. Then, after the second round, nodes 3 and 7 become black to support nodes 2 and 11, respectively.

be connected to a nearby VDB if one or two green nodes between them turn black. For example, in Fig. 4, a partially formed VDB {3,4} can reach another partially formed VDB {13} via node 12, and reach VDB {7,8,17} via nodes 11 and 12. This property can be readily proved by rejecting the conclusion (no black node exists within more than three hops from a black node) and finding that an assumption is violated (a node cannot be green without a black neighbor). This property is essential for connecting disjoint VDBs, which is described in the next section.

3.2 Backbone Connection Process (BCP)

The set of black nodes resulting from the BSP may not be connected. For example, there may be disjoint groups of backbone nodes where only nodes inside the group are connected. The BCP is employed to connect these disjoint VDBs. As mentioned before, it can be easily proven that the BSP will guarantee that at most two green nodes will exist between two nearby disjoint backbone nodes. This means that each green node is responsible for detecting whether it needs to become black to connect disjoint groups of backbone nodes.

The main idea behind the BCP is that if VDB were connected, a green node g will hear two identical VWMs (in terms of source ID and source sequence number) from each of its neighbors: *direct* VWM, and *forwarded* VWM via its black neighbors. This is from the property that VWMs are forwarded by VDB nodes. Once a node becomes green, it begins to closely monitor all the VWMs *originating* from all of its *one-hop* neighbors. If g detects that more than a fixed number of duplicate VWMs have not been received for a neighbor n via one of its black neighbor b, g is eligible to become black by connecting n and b.

Note that each VWM carries the source node sequence number field. A green node maintains a sequence number table (SNT) whose entry consists of two fields: neighbor node ID, and the source sequence numbers of VWMs for which no duplicates have been received (yet) from this neighbor. When a green node receives a VWM, it first checks if the source sequence number of this VWM exists in its corresponding SNT entry. If so, indicating that it is a duplicate, the green node discards from the SNT all the sequence numbers that are equal to or smaller than the current one. Otherwise, indicating that it is either a new one or it has been previously discarded, the green node adds this sequence number to the corresponding SNT entry. If it accumulates more than the fixed number of sequence numbers (say 5), the green node is eligible to be part of the VDB by connecting this neighbor and the black neighbor it is monitoring.

Figure 5 illustrates BCP. First, node 12 detects that it is not receiving VWMs generated by node 4 (13) via 13 (4), and thus becomes black. In the same way, node 11 detects that nodes 12 and 7 are disconnected.

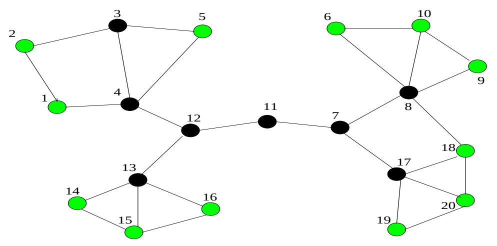

Figure 5: Illustration of Backbone Connection Process. Node 11 (or 12 whichever detects first) asks node 12 to become black to connect nodes 7 and 12.

3.3 Backbone Maintenance Process (BMP)

As nodes freely move, join (power-on), or leave (power-off) the network, it is very likely that the size of VDB becomes sub-optimal (i.e., too many black nodes compared to the entire network size, leading to high redundancy and unnecessary bandwidth consumption due to VWM forwarding). For example, some black nodes may not be qualified to be part of the VDB any more, or some green nodes are more qualified to become black than their VAPs. Since it is crucial to keep the size of VDB very small to reduce the overhead of VWM forwarding, the BMP must enable unnecessary black nodes to be detached from VDB, and qualified green nodes to join VDB. This will be achieved by ensuring that VDB consists of nodes with *high* degree. Another goal of BMP is to prevent dramatic changes in the VDB structure. If a color or degree change of a node triggers too many route or color changes of nearby nodes, it will undermine benefits of having VDB since packets in transit may get dropped or it may lead to longer routes or packet drops.

We first consider conditions that keep the size of VDB small:

- If a green node's VAP is deleted from its NIT, it is also removed from the VAP_{list}. When VAP_{list} becomes empty, the green node chooses a black neighbor (or green if none) with the highest degree as its new VAP, subject to LQ constraint.
- If no neighbors depend on a black node b as their VAP, the VAP timer T_{VAP} will expire. Then, b turns green after choosing one of its black (or green if none) neighbors with highest degree and acceptable two-way LQ as its VAP.
- If a green node g detects either a new green or black neighbor who has higher degree than its current VAP(s), g asks this neighbor to become its VAP subject to two-way LQ constraint.

Second, in order to keep the structure of VDB stable under high node mobility and link quality fluctuation, we introduce the concept of "virtual handoff". The virtual handoff is considered between a green node and each of its VAPs only.

When a green node n detects that the inbound link quality for its VAP drops below `LQ_THRESHOLD`, n deletes this VAP from its VAP_{list}. If this was the last VAP and there is another black neighbor with increasing LQ and above the threshold, choose this black neighbor as its new VAP. If there is no other black neighbor, choose a green neighbor with the highest degree as its new VAP subject to inbound link quality condition.

Finally, when two adjacent black nodes begin to lose its link, the BCP will cure this soon-to-be broken link if green nodes are available around them. In this way, a VDB always attempts to connect itself *locally* when disjoint VDBs occur. Hence, the BCP algorithm is an important part of maintaining a stable VDB.

4 Routing

4.1 Unicast Routing

Since VWMs carry source and forward node IDs, each node can readily maintain a unicast routing table based on the received VWMs forwarded by its black neighbors. Upon receiving a VWM, each node updates the corresponding routing entry: destination, primary gateway, secondary gateway, routing metric, and *stale timer* (T_{st}). The source of the received VWM becomes the destination, the black node which has forwarded this VWM with the (second) best routing metric becomes the primary (secondary) gateway, and T_{st} represents the remaining time after which this route becomes stale. Note that routes for one-hop neighbors do not have primary gateways since they can be reached directly. Currently, Hop Count is the only cost metric for routing subject to link quality between gateway and sender. In the future, other metrics such as delay or bandwidth will be taken into account when determining routes with QoS support.

An example of a routing table at node 17 in Figure 6 is given in Table 2. It is assumed that the TTL value for VWMs is 3 so that VWMs travel no more than three hops.

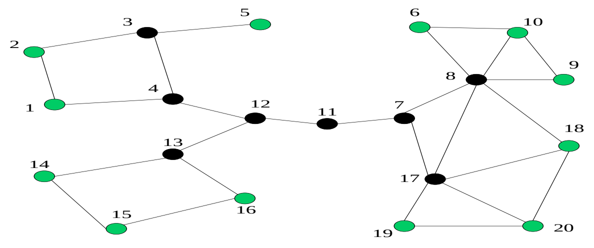

Figure 6: State Transition Diagram.

Destination ($\mathcal{D}$)	*Primary*			*Secondary*		
	G	HC	T_{st}	G	HC	T_{st}
18	-	1	15	7	2	21
19	-	1	23	-	-	-
20	-	1	25	-	-	-
7	-	1	28	8	2	29
8	-	1	22	7	2	27
11	7	2	9	8	3	15
12	7	3	18	8	4	10
9	8	2	13	7	3	22
10	8	2	22	7	3	25
6	8	2	26	7	3	22

Table 2: Example of a routing table at node 17 corresponding to Fig. 6 for destinations with up to three hops away.

Suppose node n has received a VWM with ($\mathcal{S}$, $\mathcal{F}$, HC) = (s, f, h). The following basic rules are then applied for routing table maintenance:

- If there is no route for destination s, a primary route is created for s with f as the primary gateway and distance h. The corresponding secondary route is empty.
- If a primary route for s exists, but not the secondary route, and f is different from the primary gateway, a secondary route is created for s.
- If both primary and secondary routes for s exist, and the received VWM contains a new gateway f with h better than the secondary but larger than primary route, the secondary route is replaced with f and h.
- If both primary and secondary routes for s exist, and the received VWM contains a new gateway f with h better than the primary route, the primary route is replaced with f and h, and the now-old primary route replaces the secondary route, including the stale timer.
- If both primary and secondary routes for s exist, and the received VWM does not contain a new gateway but has better hop count, only the hop count of the corresponding route is updated.
- For all route updates, corresponding stale timer T_{ST} is set to `ROUTE_STALE_TIMEOUT` (except when the old primary route becomes the secondary route).
- When a VAP is removed from the NIT, all the corresponding routes with this VAP as gateway (either primary or secondary) are also removed. If the route being removed is primary, then the corresponding secondary route becomes primary.

Since VWMs are TTL-limited, the above approach gives routes to destinations which are likely to be located closely. To address routing for the destinations whose VWMs are not heard, each node maintains a small table called *route cache*, separate from the routing table, which contains route information (gateway and hop count) for destinations who are not in the routing table (since destination is too far away).

A Route Query Message (RQM) is used to find a route to the destination when it is not in its routing table and no route cache exists. This message is broadcast over VDB. Any black node receiving an RQM keeps a record of ($\mathcal{S}$, $\mathcal{F}$, $\mathcal{D}$) for a fixed amount of time ($\mathcal{D}$ is the node ID of the destination). If no corresponding Route Response Message (RRM) is received during that time, this record is thrown away. The first black node which has the destination within its routing table returns then Route Response Message (RRM). Another words, if an RQM is received with ($\mathcal{S}$, $\mathcal{F}$) = (s, f), the corresponding RRM is generated with ($\mathcal{D}$, $\mathcal{N}$) = (s, f). All the intermediate black nodes receiving an RRM configures the route and stores in its Route Cache.

4.2 Multicast Routing

With VDB, multicast support is readily achieved since VDB can be used as multicast backbone. Any multicast tree constructed is a subset of VDB. There are two components for

Status	*Actions*
VAP_{up} not set, and VAP_{down} configured	Forward if first MJM for G. Otherwise, do not forward.
Only VAP_{down} configured	Do not forward
Only VAP_{up} configured	Forward if first MJM for G. Otherwise, do not forward.
Both VAP_{up} and VAP_{down} configured	Do not forward.

Table 3: MJM forwarding rules at CP.

multicast support over VDB: (i) group membership management, and (ii) routing.

In an IP network, local group management is done by Internet Group Membership Protocol. We use a similar technique for our purpose. A multicast group is identified as a unique multicast address. For each multicast group, a black node maintains local group membership information based on two special messages: group join (MJM) and leave messages (MLM).

We adopt the "broadcast-and-prune" approach of Distance Vector Multicast Routing Protocol (DVMRP) of Internet to achieve multicast support over VDB. Each black node maintains multicast routing table consisting of four fields: group address, local members, upstream multicast VAP (VAP_{up}), and downstream multicast VAP(VAP_{down}). Note that the notion of "up" and "down" is with respect to the node that generates the first MJM, regardless of whether it is a source or receiver of the group. This implies that there is only one VAP_{up} but may be multiple VAP_{down}s for a given group. However, our multicast routing algorithm is designed such that any member can act as a source.

1. When a node wishes to join a group G, it generates MJM with (group ID, connection point) = (G, CP) where a connection point is chosen from VAP_{list}.

2. When a black node b receives an MJM for a group G from its neighbor for which it serves as CP, b updates its local membership table. Depending on the multicast routing table status, b decides whether to forward this message further as summarized in Table 3.

3. Any black node b_2 receiving a *forwarded* MJM by a neighbor black node b_1 updates its multicast routing table described in Table 4.

Status	*Actions*
None of VAP_{up} and VAP_{down} configured	$VAP_{up} = b_1$ and forward over VDB if no local members. Otherwise, $VAP_{down} = b_1$ and do not forward.
Only VAP_{up} configured	$VAP_{down} = b_1$ and forward to VAP_{up} if no local members. Otherwise, $VAP_{down} = b_1$ and do not forward.
Only VAP_{down}(s) present	Add b_1 to VAP_{down} list if new. Do not forward.
Both VAP_{up} and VAP_{down} present	Add b_1 to VAP_{down} list if new. Do not forward.

Table 4: MJM forwarding rules at other black nodes.

4. When a node n changes its VAP via which that it has joined a group G, it must generate a new MJM to the new VAP. The old VAP will remove this neighbor when it is deleted from its NIT.

5. When a node n leaves a group G, it generates MLM to its CP b via which it has joined G. Upon receiving this message, b deletes n from its local membership table. If n is the last local member for D and b is at the end of the tree (i.e, only one multicast VAP is configured), it prunes itself from the tree by forwarding the leave message to that multicast VAP and delete it from the multicast routing table. Otherwise (either there are still local members or more than one multicast VAP present), it does not forward the MLM.

6. If a black node b_1 receives a *forwarded* MLM from one of its multicast VAPs, say b_2, it removes b_2 from its multicast routing table. If b_1 has local members for G or there are remaining multicast VAPs (either up or down), the received MLM is not forwarded. Otherwise, b_1 forwards it to other remaining multicast VAPs.

7. With the information of local membership, upstream gateway, and downstream gateway(s) maintained as described above, a multicast routing is done by simply forwarding multicast packets to appropriate multicast VAPs.

We illustrate these rules in Figure 7. Suppose node 1 is the first member to join group G. The join message MJM initiated by node 1 is broadcast over VDB, and all the black nodes except node 3 configure their VAP_{up} accordingly [Fig. 7-(a)]. Now, node 6 joins G via node 7. Since node 7 already has $VAP_{up} = 5$ configured for G, it sends this join message only to node 5, which in turn forwards it to its VAP_{up}, which is node 3 [Fig. 7-(b)]. Node 10 now joins as the third member of the group. Node 9 forwards this join message to its $VAP_{up} = 7$, which configures node 9 as its VAP_{down}. This time, node 7 does not need to forward this join message further since it already did when its local member (node 6) joined [Fig. 7-(c)]. Figure 7-(d) shows how a multicast datagram is forwarded over VDB with node 1 as a source. Each intermediate node in the tree receiving multicast packets from one direction forwards them to the other direction. If there are multiple VAP_{down}s and multicast packets arrive via one of them, they are forwarded to both VAP_{up} and all other VAP_{down}s. Nodes that have only one multicast VAP configured (edges of the tree) do not forward multicast datagrams.[2] When node 1 leaves G, it sends the MLM to node 3 [Fig. 7-(d)]. Since node 1 is the last local member and node 3 knows it is at the edge of the tree, node 3 no longer needs to be part of the tree. Hence, it forwards the leave message to its downstream VAP (node 5) and deletes it from the table. Similar to node 3, node 5 removes itself from the tree by forwarding the received MLM to node 7 and delete 7 from the table. Node 7 unassigns node 5 as its upstream VAP, but does not forward this leave message further since it has a local member. After this step, the multicast tree for G consists of nodes 7 and 9 only. Finally, when nodes 6 and 10 leave the group, all the multicast routing tables of nodes 7 and 9 become empty. The multicast routing table for node 8 becomes empty after timeout since it has not been part of the tree.

[2]Note that both nodes 6 and 7 can also transmit multicast datagrams over the same tree.

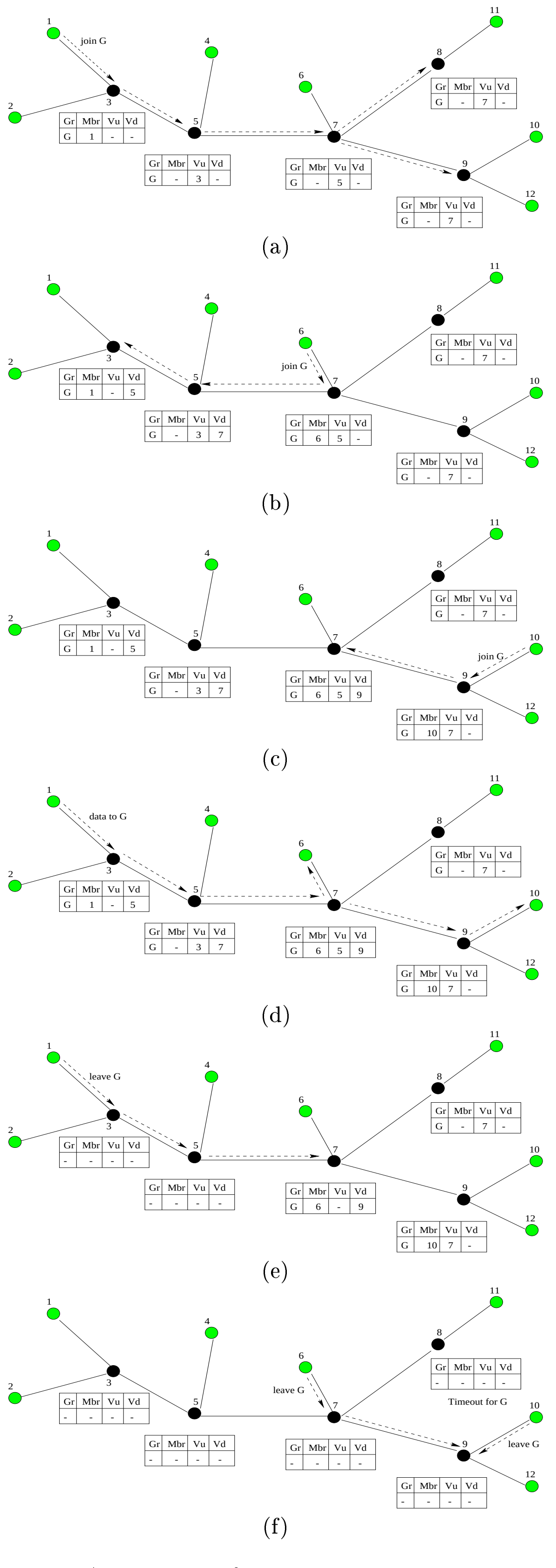

Figure 7: An example of setting up and tearing down a multicast tree over VDB (Gr: group address, Mbr: local members, V_u: upstream VAP, V_d: downstream VAP).

5 Issues

In this section, we sketch several key issues requiring further study.

5.1 Overhead

The overhead resulting from the use of virtual wire can be decomposed into two components: bandwidth consumption and power consumption. We briefly analyze each overhead.

Suppose the size of VWM is 20 bytes (as implemented in [12]) and each node generates VWMs at the rate of 1 per 10 sec. If a TTL value is chosen such that each black node forwards VWMs generated by about 100 to 300 nodes, the bandwidth consumed by VWMs per each black node is given by $1.6 \sim 4.8$ kbps. This overhead appears acceptable as long as this constitutes less than 5-10 % of the raw channel bandwidth, considering the benefits achieved by the use of VWMs. The more important overhead concern is the case in which mobility and link quality dynamics result in a cluster of black nodes. If N black nodes are adjacent to each other, the bandwidth required for forwarding VWMs will be proportional to N. In this case, there is a good chance that VWMs will take the most bandwidth available, leaving little for actual data routing. We are looking at several techniques that can prevent this from happening.

Another important issue concerning VWM overhead is power consumption, especially for black nodes. Since each black node needs to process and forward 10-30 VMWs per second, power consumption is likely to be an important factor to be considered. But with the advance in low-power DSP technology in recent years, this may be addressed via power-efficient implementation of radios.

5.2 Routing Sub-Optimality, Redundancy, and Scalability

The algorithm presented in the earlier sections is fully *distributed*, i.e., no global, or even partially global information is exchanged between nodes to construct and maintain a VDB. We chose this approach over centralized (or partially centralized) algorithms (such as link state routing) since the exchange of global information requires some form of reliable delivery of topology/routing updates, which may not be trivial in highly dynamic environment. By managing a VDB based on local information and unreliable multi-hop forwarding of VWMs, we avoid the use of reliable packet delivery. The trade-off is that the size of VDB may be sub-optimal, resulting in sub-optimal routing. We are currently investigating the performance of the algorithm in terms of the size of VDB constructed for a variety of topologies.

Secondly, algorithm also does not prevent having a loop in the constructed VDB. Note, however, that this loop does not pose a problem since each black node detects duplicate VWMs and drops them. In fact, a loop in the VDB gives a certain degree of redundancy and better routes for some nodes. For example, in Fig. 5 suppose nodes 5 and 6 are adjacent and the BCP makes both of them black for connecting nodes 3 and 8, in addition to nodes 11 and 12. This gives a good redundancy in case the link between nodes 7 and 11 break down. Furthermore, this redundant VDB link gives a better route between nodes 3 and 8. Without the VDB link between nodes 5 and 6, packets from node 3 to node 8 must travel via nodes 4, 12, 11, and 7, resulting two more hops than the path via nodes 5 and6.

Finally, the use of unreliable delivery adopted in our algorithm better supports scalability in terms of the network

	Spine	VDB w/ Virtual Wire
Algorithm	An approximation to MCDS	An heuristic algorithm based on local information only
LQ Granularity	2 (On/Off)	arbitrary ($\gg 2$)
Signaling	Requires reliable delivery of global (or partially global) broadcast over spine	Unreliable multi-hop broadcast of VWMs
Routing Optimality	Yes	No
Routing	Routing over Spine for backup only	Always route over VDB all the time (except direct neighbors)
Bandwidth Overhead	Low	Medium

Table 5: A comparison between VDB and Spine. (MCDS: Minimum Connected Dominating Set)

size. Since each node always has the most up-to-date information on which nodes it can reach and the algorithm makes a VDB recover locally, we expect this routing algorithm to support a very large ad hoc network (on the order of 1000 nodes) even without using any hierarchy in routing. Simulation study will reveal this claim, which will be reported elsewhere.

5.3 Asymmetric Routing

The algorithm presented here can be readily extended to support asymmetric routing. The current algorithm assumes a good bi-directional link between two nodes, which may not be always the case. A simple extension of allowing different VAPs for inbound and outbound links (direction-sensitive VAPs) is expected to address link asymmetry. This is under study, and the resulting algorithm will be reported elsewhere.

5.4 QoS Support

Since a VDB consists of good links, QoS support is expected to be manageable since all the routing is done over VDB. Seamless QoS support when a change in VDB structure occurs is currently under investigation

5.5 Comparison with "Spine"

Since our VDB is very close to "Spine" [1] in its structure, we make a comparison in Table 5 and delineate the major differences.

6 Conclusion

In summary, the algorithm presented here combines *virtual wire* into *virtual dynamic backbone*, yielding an ad hoc network architecture which consists of *mobile hosts* and *mobile base stations* connected via *virtual links*. We argue that this architecture is suitable for highly dynamic wireless ad hoc networks since it significantly simplifies routing and mobility management even under frequently fluctuating link quality. A fully distributed algorithm that creates and manages a VDB is presented, and algorithms for unicast and multicast routing over VDB are described. Finally, various issues concerning the algorithm in terms of overhead, routing sub-optimality, redundancy, scalability, QoS support, and asymmetric routing have been raised.

We are currently undertaking extensive simulation for evaluating its performance under various topology and node mobility scenarios, the results of which will be reported elsewhere.

Acknowledgement

We are grateful to Bharghavan Vaduvur for his comments and helpful discussions on spine routing. This work was conducted while the first author was with HRL Laboratories.

References

[1] V. Bharghavan, R. Sivakumar, and B. Das. Spine routing in ad hoc networks. submitted, 1998.

[2] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *Proc. ICC'97*, Montreal, Jun 1997.

[3] B. Das, R. Sivakumar, and V. Bharghavan. Rouing in ad-hoc networks using a spine. In *Proc. ICCCN'97*, 1997.

[4] R. Dube, C. Rais, K.-Y. Wang, and S. Tripathi. Signal stability-based adaptive routing (ssa) for ad hoc mobile networks. *IEEE Personal Comm. Mag.*, 4(1), Feb. 1997.

[5] Z. Haas and M. Pearlman. The zone routing protocol (ZRP) for ad hoc networks. Internet Draft, Aug 1998.

[6] M. Jiang, J. Li, and Y.-C. Tay. Cluster based routing protocol (CBRP) functional specification. Internet Draft, Aug 1998.

[7] D. Johnson, D. Maltz, and J. Broch. The dynamic source routing protocol for mobile ad hoc networks. Internet Draft, March 1998.

[8] Y.-B. Ko and N. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proc. MobiCom'98*, Oct. 1998.

[9] G. Lauer. Packet-radio routing. In M. Streenstrup, editor, *Routing in Communications*, pages 351–396. Prentice Hall, Englewood Cliffs, NJ, 1995.

[10] V. Park and M. Corson. Temporally-ordered routing algorithm (TORA) version 1 functional specification. Internet Draft, Aug 1998.

[11] C. Perkins and E. Royer. Ad hoc on demand routing distance vector (AODV) routing. Internet Draft, Aug 1998.

[12] B. Ryu, J. Erickson, J. Smallcomb, and S. Dao. Self-forming, self-maintaining virtual backbone-based radio for wireless ad hoc networks. US Patent application (pending), Jan 1999.