3

DSDV Routing over a Multihop Wireless Network of Mobile Computers

Charles E. Perkins
IBM Research

Pravin Bhagwat
University of Maryland

Abstract

An *ad hoc* network is the cooperative engagement of a collection of mobile nodes without the required intervention of any centralized access point. In this chapter we present a design for the operation of such ad hoc networks. The basic idea of the design is to operate each mobile node as a specialized router, which periodically advertises its view of the interconnection topology with other mobile nodes within the network. This amounts to a new sort of routing protocol. We have investigated modifications to the basic Bellman-Ford [Bertsekas+ 1987] routing mechanisms, as specified by the Routing Information Protocol (RIP) [Malkin 1993], making it suitable for a dynamic and self-starting network mechanism as is required by users wishing to utilize ad hoc networks. Our modifications address some of the previous objections to the use of Bellman-Ford, related to the poor looping properties of such algorithms in the face of broken links and the resulting time-dependent nature of the interconnection topology describing the links between the mobile nodes. Finally, we describe the ways in which the basic network-layer routing can be modified to provide MAC-layer support for ad hoc networks.

Note: From *Mobile Computing,* Imielinski, T., and Korth, H. F. (eds.), pp. 183–206—Chapter 6 by C. E. Perkins and P. Bhagwat. Norwood, Mass.: Kluwer, 1996.

3.1 INTRODUCTION

Recently there has been tremendous growth in the sales of laptop and portable computers. These smaller computers, despite their size, can be equipped with hundreds of megabytes of disk storage, high-resolution color displays, pointing devices, and wireless communications adapters. Moreover, since many of these small (in size only) computers operate for hours with battery power, users are free to move about at their convenience without being constrained by wires.

This is a revolutionary development in personal computing. Battery-powered, untethered computers are likely to become a pervasive part of our computing infrastructure. As mobile computers become handy, for whatever purposes, sharing information between them will become a natural requirement. Currently such sharing is made difficult by the need for users to perform administrative tasks and set up static, bidirectional links between their computers. However, if the wireless communications systems in the mobile computers support a broadcast mechanism, much more flexible and useful ways of sharing information can be imagined. For instance, any number of people could conceivably enter a conference room and agree to support communications links between themselves, without necessarily engaging the services of existing equipment in the room (i.e., without requiring any existing communications infrastructure). Thus, one of our primary motivations is to allow the construction of temporary networks with no wires and no administrative intervention required. In this chapter, such an interconnection between mobile computers will be called an *ad hoc* network, in conformance with current usage within the IEEE 802.11 subcommittee [IEEE 1997].

Ad hoc networks differ significantly from existing networks. First, the topology of interconnections may be quite dynamic. Second, most users will not wish to perform any administrative actions to set up such a network. To provide service in the most general situation, we do not assume that every computer is within communication range of every other computer. This lack of complete connectivity would certainly be a reasonable characteristic of, say, a population of mobile computers in a large room that rely on infrared transceivers to effect their data communications.

From a graph theoretic point of view, an ad hoc network is a graph, $G(N, E(t))$, which is formed by denoting each of the N mobile hosts by a node and drawing an edge between two nodes if they are in direct communication range of each other. The set of edges, $E(t)$, so formed is a function of time and keeps changing as nodes in the ad hoc network move around. The topology defined by such a network can be very arbitrary, as there are no constraints on where mobiles can be located with respect to each other.

Routing protocols for existing networks [Malkin 1993, McQuillan+ 1980, Schwartz+ 1980] have not been designed specifically to provide the kind of self-starting behavior needed for ad hoc networks. Most protocols exhibit their least desirable behavior when presented with a highly dynamic interconnection topology. Although we thought that mobile computers could naturally be modeled as *routers,* it was also clear that existing routing protocols would place too heavy a computational burden on each one. Moreover, the convergence characteristics of existing routing protocols did not seem good enough to fit the needs of ad hoc networks. Lastly, the wireless medium differs in important ways from wired media, which requires that we make modifications to whichever routing protocol we might choose to experiment with. For instance, mobile computers may well have only a single network interface adapter, whereas most existing routers have network interfaces to connect two separate networks together. Because we had to make many changes anyway, we decided to follow our ad hoc network model as far as we could. We ended up with a substantially new approach to the classic distance-vector routing.

3.2 OVERVIEW OF ROUTING METHODS

In our environment, the problem of routing is essentially the distributed version of the shortest-path problem [Schwartz+ 1980]. Each node in the network maintains for each destination a preferred neighbor (a *next hop*). Each data packet contains a destination node identifier in its header. When a node receives a data packet, it forwards the packet to the preferred neighbor for its destination. The forwarding process continues until the packet reaches its destination. The manner in which route tables are constructed, maintained, and updated differs from one routing method to another. Popular routing methods, however, attempt to achieve the common objective of routing packets along the optimal path. The next-hop routing methods can be categorized as two primary classes: *link-state* and *distance-vector.*

3.2.1 Link-State

The link-state approach is closer to the centralized version of the shortest-path computation method. Each node maintains a view of the network topology with a cost for each link. To keep these views consistent, each node periodically broadcasts the link costs of its outgoing links to all other nodes using a protocol such as flooding. As a node receives this information, it updates its view of the network topology and applies a shortest-path algorithm to choose its next hop for each destination. Some of the information about the link costs at any particular node can be incorrect because of long

propagation delays, a partitioned network, and so forth. Such inconsistent views of network topologies might lead to formation of routing loops. These loops, however, are short-lived, because they disappear in the time it takes a message to traverse the diameter of the network [McQuillan+ 1980].

3.2.2 Distance-Vector

In traditional distance-vector algorithms, every node i maintains, for each destination x, a set of distances $\{d_{ij}(x)\}$ for each node j that is a neighbor of i. Node i treats neighbor k as a next hop for a packet destined for x if $d_{ik}(x)$ equals $min_j\{d_{ij}(x)\}$. The succession of next hops chosen in this manner leads to x along the shortest path. To keep the distance estimates up to date, each node monitors the cost of its outgoing links and periodically broadcasts, to all of its neighbors, its current estimate of the shortest distance to every other node in the network.

The distance-vector algorithm just described is the classical Distributed Bellman-Ford (DBF) algorithm [Bertsekas+ 1987]. Compared to link-state algorithms, it is computationally more efficient, is easier to implement, and requires much less storage space. However, it is well known that this algorithm can cause the formation of both short-lived and long-lived loops [Cheng+ 1989]. The primary cause for formation of routing loops is that nodes choose their next hops in a completely distributed fashion on the basis of information that may be stale and therefore incorrect. Almost all proposed modifications to the DBF algorithm [Jaffe+ 1982, Garcia-Luna-Aceves 1989, Merlin+ 1979] eliminate the looping problem by forcing all nodes in the network to participate in some form of internodal coordination protocol. Such internodal coordination might be effective when topological changes are rare. However, within an ad hoc mobile environment, enforcing any internodal coordination mechanism will be difficult because of the rapidly changing topology of the underlying routing network.

Simplicity is one of the primary attributes that make any one routing protocol preferred over others for implementation within operational networks. The Routing Information Protocol (RIP) [Malkin 1993] is a well-known example. Despite the *counting-to-infinity* problem, it has proven to be very successful within small internetworks. The usefulness of RIP within an ad hoc environment, however, is limited, as it was not designed to handle rapid topological changes. Furthermore, the techniques of *split-horizon* and *poisoned-reverse* [Malkin 1993] are not useful within the wireless environment for devices that have a single network interface to a restricted broadcast transmission medium. For these reasons, our design goal has been a routing method for ad hoc networks that preserves the simplicity of RIP yet at the same time avoids the looping problem. Our approach is to tag each route table entry with a sequence number so that nodes can quickly

distinguish stale routes from the new ones and thus avoid formation of routing loops.

3.3 DESTINATION-SEQUENCED DISTANCE-VECTOR PROTOCOL

Consider a collection of mobile computers, which may be far from any base station, that can exchange data along changing and arbitrary paths of interconnection. The computers must also exchange control messages so that all computers in the collection have a (possibly multihop) path along which data can be exchanged. The solution must remain compatible with operation in cases where a base station is available. By the methods outlined in this chapter, not only will we see routing as solving the problems associated with ad hoc networks, but we will also describe ways to perform such routing functions at layer 2, which traditionally has not been utilized as a protocol level for routing.

3.3.1 Protocol Overview

Packets are transmitted between the nodes of the network using route tables stored at each node. Each route table, at each of the nodes, lists all available destinations and the number of hops to each. Each route table entry is tagged with a sequence number that is originated by the destination node. To maintain the consistency of route tables in a dynamically varying topology, each node periodically transmits updates, doing so immediately when significant new information is available. Since we do not assume that the mobile hosts are maintaining any sort of time synchronization, we also make no assumption about the phase relationship of the update periods between the mobile hosts. These packets indicate which nodes are accessible from each node and the number of hops necessary to reach them, following traditional distance-vector routing algorithms. It is not the purpose of this chapter to propose any new metrics for route selection other than the freshness of the sequence numbers associated with the route; cost or other metrics might easily replace the number of hops in other implementations. We permit packets to be transmitted containing either layer-2 (MAC) addresses or layer-3 (network) addresses.

Routing information is advertised by broadcasting or multicasting the packets that are transmitted periodically and incrementally as topological changes are detected—for instance, when nodes move within the network. Data is also kept about the length of time between the arrival of the *first* and the arrival of the *best* route for each particular destination. On the basis of this data, a decision may be made to delay advertising routes that are about to change, thus damping fluctuations of the route tables. The advertisement of possibly unstable routes is delayed to reduce the number

of rebroadcasts of possible route entries that normally arrive with the same sequence number.

3.3.2 Route Advertisements

The DSDV protocol requires each mobile node to advertise, to each of its current neighbors, its own route table (for instance, by broadcasting its entries). The entries in this list may change fairly dynamically over time, so the advertisement must be made often enough to ensure that every mobile computer can almost always locate every other mobile computer in the collection. In addition, each mobile computer agrees to relay data packets to other computers upon request. This agreement places a premium on the ability to determine the shortest number of hops for a route to a destination; we want to avoid disturbing mobile hosts unnecessarily if they are in sleep mode. In this way a mobile computer may exchange data with any other mobile computer in the group even if the target of the data is not within range for direct communication. If the notification about other mobile computers that are accessible from any particular computer in the collection is done at layer 2, DSDV will work with whatever higher-layer (e.g., network layer) protocol might be in use.[1]

All the computers interoperating to create data paths between themselves broadcast the necessary data periodically, say, once every few seconds. In a wireless medium, it is important to keep in mind that broadcasts are limited in range by the physical characteristics of the medium, in ways that are difficult to characterize precisely. This is different from the situation with wired media, which usually have a much more clearly defined range of reception.

3.3.3 Route Table Entry Structure

The data broadcast by each mobile computer will contain its new sequence number and the following information for each new route:

- The destination's address
- The number of hops required to reach the destination
- The sequence number of the information received regarding that destination, as originally stamped by the destination

Within the headers of the packet, the transmitted route tables will also contain the hardware address and (if appropriate) the network address of the mobile computer transmitting them. The route tables will also include

[1] But, see Chapter 1, Section 1.1.2, regarding problems with address resolution.

a sequence number created by the transmitter. Routes with more recent sequence numbers are always preferred as the basis for forwarding decisions, but they are not necessarily advertised. Of the paths with the same sequence number, those with the smallest metric will be used. By the natural way in which the route tables are propagated, the sequence number is sent to all mobile computers, which may each decide to maintain a routing entry for that originating mobile computer.

Routes received in broadcasts are also advertised by the receiver when it subsequently broadcasts its routing information; the receiver adds an increment to the metric before advertising the route, as incoming packets will require one more hop to reach the destination (namely, the hop from the transmitter to the receiver). Again, we do not explicitly consider here the changes required to use metrics that do not use the hop count to the destination.

Wireless media differ from traditional wired networks because asymmetries produced by one-way "links" are more prevalent. Receiving a packet from a neighbor therefore does not indicate the existence of a single-hop data path back to that neighbor across the wireless medium. To avoid problems caused by such one-way links, no mobile node may insert routing information received from a neighbor unless that neighbor shows that it can receive packets from the mobile node. Thus, our routing algorithms effectively use only links that are bidirectional.

One of the most important parameters to be chosen is the time between broadcasting the routing information packets. However, when any new or substantially modified route information is received by a mobile node, the new information will be retransmitted soon (subject to constraints imposed for damping route fluctuations), effecting the most rapid as possible dissemination of routing information among all of the cooperating mobile nodes. This quick rebroadcast introduces a new requirement for our protocols to converge as soon as possible. It would be calamitous if the movement of a mobile node caused a storm of broadcasts, degrading the availability of the wireless medium.

3.3.4 Responding to Topology Changes

Mobile nodes cause broken links as they move from place to place. The broken link may be detected by the layer-2 protocol, or it may be inferred if no broadcasts have been received for a while from a former neighbor. A broken link is described by a metric of ∞ (i.e., any value greater than the maximum allowed metric). When a link to a next hop has broken, any route through that next hop is immediately assigned an ∞ metric and an updated sequence number. Since this qualifies as a substantial route change, such modified routes are immediately disclosed in a broadcast routing information

packet. Building information to describe broken links is the only situation in which the sequence number is generated by any mobile node other than the destination mobile node. Sequence numbers generated to indicate ∞ hops to a destination will be one greater than the last sequence number received from the destination. When a node receives an ∞ metric, and it has an equal or later sequence number with a finite metric, it triggers a route update broadcast to disseminate the important news about that destination. In this way routes containing any finite metric will supersede routes generated with the ∞ metric.

In a very large population of mobile nodes, adjustments will likely be needed for the time between broadcasts of the routing information packets. To reduce the amount of information carried in these packets, two types will be defined. One, called a *full dump,* will carry all of the available routing information. The other, called an *incremental,* will carry only information changed since the last full dump. By design, an incremental routing update should fit in one network protocol data unit (NPDU). The full dump will most likely require multiple NPDUs, even for relatively small populations of mobile nodes. Full dumps can be transmitted relatively infrequently when no movement of mobile nodes is occurring. When movement becomes frequent and the size of an incremental approaches the size of a NPDU, a full dump can be scheduled so that the next incremental will be smaller. It is expected that mobile nodes will implement some means for determining which route changes are significant enough to be sent out with each incremental advertisement. For instance, when a stabilized route shows a different metric for some destination, that is likely to constitute a significant change that needs to be advertised after stabilization. If a new sequence number for a route is received but the metric stays the same, that is unlikely to constitute a significant change.

3.3.5 Route Selection Criteria

When a mobile node receives new routing information (usually in an incremental packet as just described), that information is compared to the information already available from previous routing information packets. Any route with a more recent sequence number is used; routes with older sequence numbers are discarded. A route with a sequence number equal to an existing route is chosen if it has a "better" metric, and the existing route is discarded or stored as less preferable. The metrics for routes chosen from the newly received broadcast information are each incremented by one hop. Newly recorded routes are scheduled for immediate advertisement to the current mobile node's neighbors. Routes that show a more recent sequence number may be scheduled for advertisement at a later time, which time depends on the average settling time for routes to the particular destination under consideration.

Timing skews between the various mobile nodes are expected. The broadcasts of routing information by the mobile nodes are to be regarded as somewhat asynchronous events, even though some regularity is expected. In such a population of independently transmitting agents, some fluctuation can develop using the preceding procedures for updating routes. It may turn out that a particular mobile node receives new routing information in a pattern that causes it to consistently change routes from one next hop to another, even when the destination mobile node has not moved. This happens because there are two ways for new routes to be chosen: They might have a later sequence number, or they might have a better metric. Conceivably, a mobile node can always receive two routes to the same destination, with a newer sequence number, one after another (via different neighbors), but it always gets the route with the worse metric first. Unless care is taken, this will lead to a continuing burst of new route transmittals upon every new sequence number from that destination. Each new metric is propagated to every mobile host in the neighborhood, which propagates to its neighbors, and so on.

One solution is to delay the advertisement of such routes when a mobile node can determine that a route with a better metric is likely to show up soon. The route with the later sequence number must be available for use, but it does not have to be advertised immediately unless it is a route to a previously unreachable destination. Thus, there will be two route tables kept at each mobile node—one for use with forwarding packets and another to be advertised via incremental routing information packets. To determine the probability of imminent arrival of routing information showing a better metric, the mobile node has to keep a history of the weighted average time that routes to a particular destination fluctuate until the route with the best metric is received. Received route updates with infinite metrics are not included in this computation of the settling time for route updates. We hope that such a procedure will allow us to predict how long to wait before advertising new routes.

3.3.6 Operating DSDV at Layer 2

The addresses stored in the route tables will correspond to the layer at which the DSDV ad hoc networking protocol is operated. That is, operation at layer 3 will use network layer addresses for the next hop and destination addresses; operation at layer 2 will use layer-2 medium access control (MAC) addresses.

Using MAC addresses for the forwarding table introduces a new requirement, however. The difficulty is that layer-3 network protocols provide communication based on network addresses, and a way must be provided to resolve these layer-3 addresses into MAC addresses. Otherwise, a broadcast address resolution mechanism would be needed, and a corresponding loss of

bandwidth in the wireless medium would be observed whenever the resolution mechanisms were utilized. This loss could be substantial because such mechanisms would require broadcasts and retransmitted broadcasts by every mobile node in the ad hoc network. Thus, unless special care is taken, every address resolution might produce a glitch in the normal operation of the network, which may well be noticeable to any active users.

The solution proposed here, for operation at layer 2, is to include layer-3 protocol information along with the layer-2 information. Each destination host would advertise which layer-3 protocols it supports, and each mobile node advertising reachability to that destination would include, along with the advertisement, the information about the layer-3 protocols supported at that destination. This information would have to be transmitted only when it changes, which occurs rarely. Changes would be transmitted as part of each incremental dump. Since each mobile node could support several layer-3 protocols (and many will), this list would have to be variable in length.

3.3.7 Extending Base Station Coverage

Mobile computers will frequently be used in conjunction with base stations, which allow them to exchange data with other computers connected to the wired network. By participating in the DSDV protocol, base stations can extend their coverage beyond the range imposed by their wireless transmitters. When a base station participates in DSDV, it is shown as a default route in the tables transmitted by a mobile node. In this way, mobile nodes within range of a base station can cooperate to effectively extend the base station range to serve other nodes out of that range, as long as those other mobile nodes are close to one of the mobile nodes that are within range.

3.4 EXAMPLES OF DSDV IN OPERATION

Consider MH_4 in Figure 3.1. Table 3.1 shows a possible structure of the forwarding table maintained at MH_4. Suppose that the address[2] of each mobile node is represented as MH_i. Suppose further that all sequence numbers are denoted SNNN_MH_i, where MH_i specifies the computer that created the sequence number and SNNN is a sequence number value. Also suppose that there are entries for all other mobile nodes, with sequence numbers SNNN_MH_i, before MH_1 moves away from MH_2. The install time field helps determine when to delete stale routes. With our protocol, the deletion

[2]If DSDV is operated at level 2, MH_i denotes the MAC address; otherwise, it denotes a level-3 address.

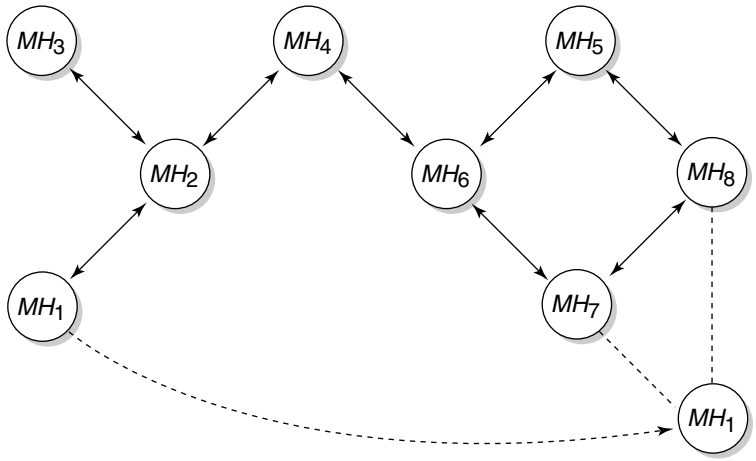

Figure 3.1. Movement in an Ad Hoc Network

Table 3.1. MH_4 Forwarding Table

Destination	Next Hop	Metric	Sequence Number	Install	Stable_Data
MH_1	MH_2	2	S406_MH_1	T001_MH_4	Ptr1_MH_1
MH_2	MH_2	1	S128_MH_2	T001_MH_4	Ptr1_MH_2
MH_3	MH_2	2	S564_MH_3	T001_MH_4	Ptr1_MH_3
MH_4	MH_4	0	S710_MH_4	T001_MH_4	Ptr1_MH_4
MH_5	MH_6	2	S392_MH_5	T002_MH_4	Ptr1_MH_5
MH_6	MH_6	1	S076_MH_6	T001_MH_4	Ptr1_MH_6
MH_7	MH_6	2	S128_MH_7	T002_MH_4	Ptr1_MH_7
MH_8	MH_6	3	S050_MH_8	T002_MH_4	Ptr1_MH_8

of stale routes should rarely occur, as the detection of link breakages should propagate through the ad hoc network immediately. Nevertheless, we monitor for the existence of stale routes and take appropriate action.

From Table 3.1, we can surmise, for instance, that all of the computers become available to MH_4 at about the same time because, for most of them, its Install_Time is about the same. Ptr1_MH_i will all be pointers to null structures because there are no routes in Figure 3.1 that are likely to be superseded or to compete with other possible routes to any particular destination.

Table 3.2 shows the structure of the advertised route table of MH_4.

Now suppose that MH_1 moves into the general vicinity of MH_8 and MH_7 and away from the others (especially MH_2). The new internal forwarding tables at MH_4 might then appear as shown in Table 3.3.

Table 3.2. MH_4 Advertised Route Table

Destination	Metric	Sequence Number
MH_1	2	S406_MH_1
MH_2	1	S128_MH_2
MH_3	2	S564_MH_3
MH_4	0	S710_MH_4
MH_5	2	S392_MH_5
MH_6	1	S076_MH_6
MH_7	2	S128_MH_7
MH_8	3	S050_MH_8

Table 3.3. MH_4 Forwarding Table (Updated)

Destination	Next Hop	Metric	Sequence Number	Install	Stable_Data
MH_1	**MH_6**	**3**	**S516**_MH_1	**T810**_MH_4	Ptr1_MH_1
MH_2	MH_2	1	S238_MH_2	T001_MH_4	Ptr1_MH_2
MH_3	MH_2	2	S674_MH_3	T001_MH_4	Ptr1_MH_3
MH_4	MH_4	0	S820_MH_4	T001_MH_4	Ptr1_MH_4
MH_5	MH_6	2	S502_MH_5	T002_MH_4	Ptr1_MH_5
MH_6	MH_6	1	S186_MH_6	T001_MH_4	Ptr1_MH_6
MH_7	MH_6	2	S238_MH_7	T002_MH_4	Ptr1_MH_7
MH_8	MH_6	3	S160_MH_8	T002_MH_4	Ptr1_MH_8

Only the entry for MH_1 shows a new metric, but in the intervening time many new sequence number entries have been received. The first entry thus must be advertised in subsequent incremental routing information updates until the next full dump occurs. When MH_1 moves into the vicinity of MH_8 and MH_7, it triggers an immediate incremental routing information update, which is then broadcast to MH_6. MH_6, having determined that significant new routing information has been received, also triggers an immediate update, which carries along the new routing information for MH_1. MH_4, upon receiving this information, then broadcasts it at every interval until the next full routing information dump. At MH_4, the incremental advertised routing update has the form shown in Table 3.4.

In this advertisement, the information for MH_4 comes first, since it is doing the advertisement. The information for MH_1 comes next, not because it has a lower address but because MH_1 is the only one that has any significant route changes affecting it. As a general rule, routes with changed

Table 3.4. MH_4 Advertised Table (Updated)

Destination	Metric	Sequence Number
MH_4	0	S820_MH_4
MH_1	3	S516_MH_1
MH_2	1	S238_MH_2
MH_3	2	S674_MH_3
MH_5	2	S502_MH_5
MH_6	1	S186_MH_6
MH_7	2	S238_MH_7
MH_8	3	S160_MH_8

metrics are first included in each incremental packet. The remaining space is used to include those routes whose sequence numbers have changed.

In this example, one node has changed its routing information, as it is in a new location. All nodes have recently transmitted new sequence numbers. If there were too many updated sequence numbers to fit in a single packet, only the ones that fit would be transmitted, selected with a view to fairly transmitting them in their turn over several incremental update intervals. There is no such required format for the transmission of full routing information packets. As many packets as needed are used, and all available information is transmitted. The frequency of transmitting full updates is reduced if the volume of data begins to consume a significant fraction of the available capacity of the medium.

3.4.1 Damping Fluctuations

This section describes how the settling time table is used to prevent fluctuations of route table entry advertisements. The general problem arises because route updates are selected according to one of the following criteria:

- Routes are always preferred if the sequence numbers are newer.
- Routes are preferred if the sequence numbers are the same and yet the metric is better.

To see the problem, suppose that two routes with identical sequence numbers are received by a mobile node but in the wrong order. In other words, suppose that MH_4 receives the higher-metric next hop first and soon after gets another next hop with a lower metric but the same sequence number. This can happen when there are many mobile nodes, all transmitting their updates irregularly. Alternatively, if the mobile hosts are acting

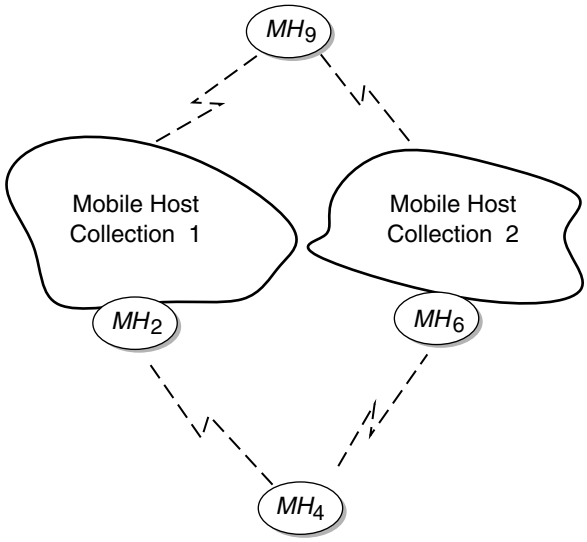

Figure 3.2. Receiving Fluctuating Routes

independently and with markedly different transmission intervals, the situation can occur with correspondingly fewer hosts. Suppose that, in any event, Figure 3.2 has enough mobile nodes to cause the problem, in two separate collections both connected to a common destination MH_9 but with no other mobile nodes in common. Suppose further that all mobile nodes are transmitting updates approximately every 15 seconds, that mobile node MH_2 has a route to MH_9 with 12 hops, and that mobile node MH_6 has a route to MH_9 with 11 hops. Moreover, suppose that the routing information update from MH_2 arrives at MH_4 approximately 10 seconds before the routing information update from MH_6. This might occur every time a new sequence number is issued from mobile node MH_9. In fact, the time differential can be significant if any mobile node in collection 2 begins to issue its sequence number updates in multiple incremental update intervals, as happens, for instance, when there are too many hosts with new sequence number updates for all of them to fit within a single incremental packet update. In general, the larger the number of hops, the larger the differentials between delivery of the updates that can be expected in Figure 3.2.

The settling time data is stored in a table with the following fields, keyed by the first field:

- Destination address
- Last settling time
- Average settling time

The settling time is calculated by maintaining, for each destination, a running, weighted average over the most recent updates of the routes.

Suppose that a new routing information update arrives at MH_4, and the sequence number in the new entry is newer than the sequence number in the currently used entry but has a worse (i.e., higher) metric. Then MH_4 must use the new entry in making subsequent forwarding decisions. However, MH_4 does not have to advertise the new route immediately and can consult its route settling time table to decide how long to wait before advertising. The average settling time is used for this determination. For instance, MH_4 may decide to delay (Average Settling_Time $\times$ 2) before advertising a route.

This can be quite beneficial because if the possibly unstable route were advertised immediately, the effects would ripple through the network; this bad effect would probably be repeated every time mobile node MH_9's sequence number updates rippled through the ad hoc network. On the other hand, if a link via mobile node MH_6 actually does break, the advertisement of a route via MH_2 should proceed immediately. To achieve this when there is a history of fluctuations at mobile node MH_4, the link breakage should be detected fast enough so that an intermediate host in collection 2 discovers the problem and begins a triggered incremental update showing an ∞ metric for the path to mobile node MH_9. Routes with an ∞ metric are required to be advertised by this protocol without delay.

To bias the damping mechanism in favor of recent events, the most recent measurement of the settling time of a particular route must be counted with a higher weighting factor than that for less recent measurements. And, importantly, a parameter must be selected that indicates how long a route has to remain stable before it is counted as actually stable. This amounts to specifying a maximum value for the settling time for the destination in the settling time table. Any route more stable than this maximum value will cause a triggered update if it is ever replaced by another route with a different next hop or metric.

When a new routing update is received from a neighbor, at the same time that the updates are applied to the table, processing also occurs to delete stale entries. Stale entries are defined as those for which no update has been applied within the last few update periods. Each neighbor is expected to send regular updates; when no updates are received for a while, the receiver may make the determination that the corresponding computer is no longer a neighbor. When that occurs, any route using that computer as a next hop should be deleted, including the route indicating that computer as the actual (formerly neighboring) destination. Increasing the number of update periods before entries are determined would result in more stale routing entries but would also allow for more transmission errors. Transmission errors are likely to occur when a CSMA-type broadcast medium is

used, as may well be the case for many wireless implementations. When the link breaks, an ∞ metric route should be advertised for it as well as for the routes that depend on it.

Additional data fields, other than those stated before, might be transmitted as part of each entry in the route tables that are broadcast by each participating computer (mobile or base station). These fields may depend, for instance, on higher-level protocols or other protocols according to the operation of layer 2. For instance, to enable correct ARP operation, each route table entry must also contain an association between the Internet Protocol (IP) address and the destination's MAC address. This also enables an intermediate computer, when serving a routing function for its neighbors, to issue proxy ARP replies instead of routing ARP broadcasts around. However, if packet forwarding is based on MAC addresses, such techniques ought to be unnecessary. And, if forwarding is based on IP addresses, no ARP is strictly necessary as long as neighboring nodes keep track of associations gleaned from route table broadcasts. Note also that layer-3 operation violates the normal subnet model of operation, as even if two mobile nodes share the same subnet address there is no guarantee that they will be directly connected—in other words, within range of each other. Even so, this is compatible with the model of operation offered by the Mobile IP Working Group of the IETF [Perkins 1994, Perkins 1996].[3]

The new routing algorithm was developed to enable the creation of ad hoc networks, which are specifically targeted to the operation of mobile computers. However, both the routing algorithm and the ad hoc network can be beneficially used in situations that do not include mobile computers. For instance, the routing algorithm can be applied in any situation in which reduced memory requirements are desired (compared to link-state routing algorithms), and an ad hoc network can be applied to wired as well as wireless mobile computers. In general, then, we provide a new destination-sequenced routing algorithm, and this algorithm is supplemented by a technique for damping fluctuations.

3.5 PROPERTIES OF THE DSDV PROTOCOL

At all instances, the DSDV protocol guarantees loop-free paths to each destination. To see why this property holds, consider a collection of N mobile hosts forming an instance of an ad hoc-style network. Further, assume that the system is in steady state; that is, the routing tables of all nodes have already converged to the actual shortest paths. At this instant, the next

[3]Note that this paragraph was originally written well before Mobile IP was promoted as a proposed standard.

node indicators to each destination induce a tree rooted at that destination. Thus, the route tables of all nodes in the network can be collectively visualized as forming N trees, one rooted at each destination. In the following discussion, we will focus our attention on one specific destination x and follow the changes occurring on the directed graph $G(x)$ defined by nodes i and arcs $(i, p_i(x))$, where $p_i(x)$ denotes the next hop for destination x at node i. Operation of the DSDV algorithm ensures that at every instant $G(x)$ is loop-free or, equivalently, is a set of disjoint directed trees. Each such tree is rooted either at x or at a node whose next hop is nil. Because this property holds with respect to each destination x, all paths induced by the route tables of the DSDV algorithm are indeed loop-free at all instants.

Starting from a loop-free state, a loop may potentially form each time node i changes its next hop. Two cases should be considered. In the first case, when node i detects that the link to its next hop is broken, the node resets $p_i(x)$ to nil. Clearly, this action cannot form a loop involving i. In the second case, node i receives, from one of its neighbors, k, a route to x, with sequence number $s_k(x)$ and metric m, which is selected to replace the current route it has through $p_i(x)$. Let $s_i(x)$ denote the value of the sequence number stored at node i and let $d_i(x)$ denote the distance estimate from i to x just prior to receiving a route from k. Node i will change its next hop from $p_i(x)$ to k only if either of the following two situations occurs.

1. The new route contains a newer sequence number; that is, $s_k(x) > s_i(x)$.
2. The sequence number $s_k(x)$ is the same as $s_i(x)$, but the new route offers a shorter path to x; that is, $m < d_i(x)$.

In the first case, by choosing k as its new next hop, node i cannot close a loop. This can be easily deduced from the following observation. Node i propagates sequence number $s_i(x)$ to its neighbors only after receiving it from its current next hop. Therefore, the sequence number value stored at the next hop is always greater than or equal to the value stored at i. Starting from node i, if we follow the chain of next-hop pointers, the sequence number values stored at visited nodes form a nondecreasing sequence. Now suppose that node i forms a loop by choosing k as its next hop. This implies that $s_k(x) \leq s_i(x)$. But this contradicts our initial assumption that $s_k(x) > s_i(x)$. Hence, loop formation cannot occur if nodes use newer sequence numbers to pick routes.

The loop-free property holds in the second scenario because of a theorem proved by Jaffe and Moss [Jaffe+ 1982], which states that in the presence of static or decreasing link weights distance-vector algorithms always maintain loop-free paths.

3.6 COMPARISON WITH OTHER METHODS

Table 3.5 presents a quick summary of some of the main features of a few chosen routing protocols. The chosen set, although small, is representative of the routing techniques most commonly employed in operational data networks. Except for the link-state approach, all routing methods shown in the table are a variant of the basic distance-vector approach. The comparison criteria reflect some of the most desirable features that a routing algorithm should possess for it to be useful in a dynamic ad hoc environment. In wireless media, communication bandwidth is the most precious and scarce resource. The formation of any kind of routing loops is therefore, highly undesirable. In the case of infrared LANs that employ a pure CSMA protocol, looping packets not only consume the communication bandwidth but can further degrade performance by causing more collisions in the medium. A common technique employed for loop prevention is what we call *internodal coordination,* whereby strong constraints on the ordering of the updates among nodes is imposed. The resulting internode protocols tend to be complex. Furthermore, their update coordination may restrict a node's ability to obtain alternate paths quickly in an environment in which topology changes are relatively frequent. The last criterion used for comparison is the space requirement of the routing method. Nodes in an ad hoc network may be battery powered laptops, or even handheld notebooks, which do not have the kind of memory that backbone routers are expected to have. Therefore, economy of space is important.

The primary concern with a DBF algorithm [Bertsekas+ 1987] in an ad hoc environment is its susceptibility to forming routing loops and the counting-to-infinity problem. RIP [Malkin 1993], which is very similar to

Table 3.5. Comparison of Various Routing Methods

Routing Method	Looping	Internodal Coordination	Space Complexity
Bellman-Ford	s/l	–	$O(nd)$
Link-State	s	–	$O(n+e)$
Loop-Free BF*	s	–	$O(nd)$
RIP	s/l	–	$O(n)$
Merlin Segall	Loop-free	Required	$O(nd)$
Jaffe Moss	Loop-free	Required	$O(nd)$
DSDV	Loop-free	–	$O(n)$

s—short-term loop, l—long-term loop, n—number of nodes, d—maximum degree of a node.

*See [Cheng+ 1989].

DBF, also suffers from this problem. Unlike DBF, RIP keeps track of only the best route to each destination, which results in some space saving at no extra performance hit. It also employs techniques known as *split-horizon* and *poisoned-reverse* to avoid a ping-pong style of looping, but these techniques are not powerful enough to avoid loops involving more than two hops. The primary cause of loop formation in DBF algorithms is that nodes make uncoordinated modifications to their route tables on the basis of information that could be incorrect. This problem is alleviated by employing an internodal coordination mechanism as proposed by Merlin and Segall [Merlin+ 1979]. A similar technique, but with better convergence results, is developed by Jaffe and Moss [Jaffe+ 1982]. However, we do not know of any operational routing protocols that employ these complex coordination methods to achieve loop freedom, which leads us to the conclusion that the usefulness of such complex methods, from a practical point of view, is diminished.

Link-state algorithms [McQuillan+ 1980] are also free of the counting-to-infinity problem. However, they need to maintain the up-to-date version of the entire network topology at every node, which may constitute excessive storage and communication overhead in a highly dynamic network. Besides, link-state algorithms proposed or implemented to date do not eliminate the creation of temporary routing loops.

It is evident that within an ad hoc environment the design tradeoffs and the constraints under which a routing method has to operate are quite different. The proposed DSDV approach offers a very attractive combination of desirable features. Its memory requirement is a very moderate $O(n)$. It guarantees loop-free paths at all instants, and it does so without requiring nodes to participate in any complex update coordination protocol. The worst-case convergence behavior of the DSDV protocol is not optimal, but in the average case it is expected that convergence will be quite rapid.

3.7 FUTURE WORK

Many parameters of interest control the behavior of DSDV—for instance, the frequency of broadcast, the frequency of full route table dumps versus incremental notifications, and the percentage change in the routing metric that triggers an immediate broadcast of new routing information. By performing simulations, we hope to discover optimal values for many of these parameters for large populations of mobile computers.

Our original goals did not include making any changes to the idea of using the number of hops as the sole metric for making route table selections. We ended up designing a method to combat fluctuations in route tables at the mobile nodes, which can be caused by information arriving faster over a path that has more hops. However, it may well be the case

that such paths are preferable just because they are faster, even if more mobile computers are involved in the creation of the path. We would like to consider how to improve the routing metric by taking into account a more sophisticated cost function that includes the effects of time and cost as well as the number of hops.

The DSDV approach relies on periodic exchange of routing information among all participating nodes. An alternative is to design a system that performs route discovery on a need-to-know basis. For devices operating on limited battery power, this may be an important design consideration.[4]

A pure on-demand system operates in two phases: *route discovery* and *route maintenance*. A source starts the first phase by broadcasting a route discovery (RD) packet in the network. These packets are relayed by all participating nodes to their respective neighbors. As an RD packet travels from a source to various destinations, it automatically causes formation of *reverse paths* from visited nodes to the source. To set up a reverse path, a node is only required to record the address of the neighbor from which it receives the first copy of the RD packet; any duplicates received thereafter are discarded. When the RD packet arrives at the destination, a reply is generated and forwarded along the reverse path. By a mechanism similar to reverse-path setup, *forward route* entries are initialized as the reply packet travels toward the source. Nodes not lying on the path between source/destination pairs eventually time out their reverse path routing entries. Once the path setup is complete, the route maintenance phase takes over. The second phase is responsible for maintaining paths between active source/destination pairs in the face of topological changes.

3.8 SUMMARY

Providing convenient connectivity for mobile computers in ad hoc networks is a challenge that is only now being met. DSDV models the mobile computers as routers cooperating to forward packets to each other as needed. We believe that this approach makes good use of the properties of the wireless broadcast medium. It can be utilized either at the network layer (layer 3) or below the network layer but still above the MAC layer software in layer 2. In the latter case certain additional information should be included along with the route tables for the most convenient and efficient operation. The information in the route tables is similar to that found in route tables with today's distance-vector (Bellman-Ford) algorithms, but it

[4]Such systems are described in several other chapters; specifically, AODV (Chapter 5) may be viewed as an on-demand modification of DSDV.

includes a sequence number as well as settling time data useful for damping out fluctuations in route table updates.

All sequence numbers are generated by the destination computer in each route table entry, except in cases when a link has been broken. Such a case is described by an ∞ metric; it is easily distinguishable, as no ∞ metric will ever be generated along the tree of intermediate nodes receiving updates originating from the destination. By the natural operation of the protocol, the metric chosen to represent broken links will be superseded as soon as possible by real routes propagated from the newly located destination. Any newly propagated routes will necessarily use a metric less than what was used to indicate the broken link. This allows real route data to quickly supersede temporary link outages when a mobile computer moves from one place to another.

We have borrowed the existing mechanism of triggered updates to make sure that pertinent route table changes can be propagated throughout the population of mobile hosts as quickly as possible whenever any topology changes are noticed. This includes movement from place to place as well as the disappearance of a mobile host from the interconnect topology (perhaps as a result of turning off its power).

To combat problems arising with large populations of mobile hosts, which can cause route updates to be received in an order delaying the best metrics until after poorer metric routes are received, we have separated the route tables into two distinct structures. The actual routing is according to information kept in the internal route table, but this information is not always advertised immediately upon receipt. We have defined a mechanism whereby routes are not advertised until it is likely, on the basis of history, that they are stable. This measurement of the settling time for each route is biased toward the most recent measurements for the purpose of computing an average.

We have found that mobile computers, modeled as routers, can effectively cooperate to build ad hoc networks. We hope to explore further the necessary application-level support needed to automatically enable use of the network-layer route capabilities to provide simple access to conferencing and workplace tools for collaboration and information sharing.

References

[Bertsekas+ 1987] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, Englewood Cliffs, N.J., 1987, 297–333.

[Cheng+ 1989] C. Cheng, R. Riley, S.P.R. Kumar, and J.J. Garcia-Luna-Aceves. A Loop-Free Bellman-Ford Routing Protocol without Bouncing Effect. In *Proceedings of ACM SIGCOMM '89,* September 1989, 224–237.

[Garcia-Luna-Aceves 1989] J.J. Garcia-Luna-Aceves. A Unified Approach to Loop-Free Routing Using Distance Vectors or Link States. In *Proceedings of ACM SIGCOMM '89,* September 1989, 212–223.

[IEEE 1997] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11-97.* The Institute of Electrical and Electronics Engineers, New York, 1997.

[Jaffe+ 1982] J.M. Jaffe and F. Moss. A Responsive Distributed Routing Algorithm for Computer Networks. *IEEE Transactions on Communications* COM-30:1758–1762, July 1982.

[Malkin 1993] G. Malkin. RIP Version 2—Carrying Additional Information. RFC 1388 (proposed standard), Internet Engineering Task Force, January 1993.

[McQuillan+ 1980] J.M. McQuillan, I. Richer, and E.C. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications* COM-28(5):711–719, May 1980.

[Merlin+ 1979] P.M. Merlin and A. Segall. A Failsafe Distributed Routing Protocol. *IEEE Transactions on Communications* COM-27:1280–1287, September 1979.

[Perkins 1994] C. Perkins. Mobile IP as Seen by the IETF. *Connexions,* March 1994, 2–20.

[Perkins 1996] C. Perkins. IP Mobility Support. RFC 2002 (proposed standard), Internet Engineering Task Force, October 1996.

[Schwartz+ 1980] M. Schwartz and T. Stern. Routing Techniques Used in Computer Communication Networks. *IEEE Transactions on Communications* COM-28: 539–552, April 1980.