Unidirectional Links Prove Costly in Wireless Ad Hoc Networks

Ravi Prakash *
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75083-0688.
email: ravip@utdallas.edu

Abstract

Most of the routing algorithms for ad hoc networks assume that all wireless links are bidirectional. In reality, some links may be unidirectional. The presence of such links can jeopardize the performance of the existing distance vector routing algorithms. In this paper we show that distance vector based routing protocols that account for unidirectional links will require nodes to exchange $O(n^2)$ information with each other, where n is the number of nodes in the network. We also present modifications to distance vector based routing algorithms to make them work in ad hoc networks with unidirectional links.

1 Introduction

The mobility pattern of the nodes in an ad hoc network is often non-deterministic. Hence, the network topology is always in a flux. There has been a significant amount of effort towards developing routing algorithms for such networks. These algorithms can be classified into (a) *cluster-based* algorithms, and (b) *flat* algorithms. In cluster-based algorithms [1, 2, 5, 6], at regular intervals, a subset of nodes is elected as *cluster-heads.* A node is either a cluster-head or one wireless hop away from a cluster-head. Nodes that are not cluster-heads will, henceforth, be referred to as *ordinary* nodes. When an ordinary node has to send a packet, the node can send the packet to the cluster-head which routes that packet towards the destination. In flat routing algorithms [7, 9, 12, 14, 15] each node maintains routing information.

*The author work is supported in part by the National Science Foundation grants CCR-9796331 and ANI-9805133.

These routing algorithms have contributed significantly towards the understanding of the problem and the feasible solution approaches. However, to successfully deploy ad hoc networks we need to understand the various ways in which RF-propagation characteristics can impact the routing problem. Models based on the IEEE 802.11 physical and medium-access control layer protocol [8] consider the propagation issues. We will not go into these issues in detail. Instead, we will concentrate on a manifestation of the realistic propagation models, namely presence of some *unidirectional links* in the network.

Some links may be unidirectional due to the *hidden terminal problem* [17] or due to disparity between the transmission power levels of the nodes at either ends of the link. Node A may be able to receive messages from node B as there may very little interference in A's vicinity. However, B may be in the vicinity of an interfering node and, therefore, be unable to receive A's messages. So, the link between A and B is directed from B to A. Link unidirectionality may be a *persistent* phenomenon, especially if some nodes experience a significant depletion of their energy supply or a persistent and strong interferer. Alternatively, unidirectionality may be a *transient* phenomenon where a link quickly transitions from unidirectional to bidirectional state. The frequency of such transitions, and the duration of stay in each state would be a function of offered traffic, terrain, mobility pattern, and energy availability.

Almost all existing routing algorithms tend to assume that all links are bidirectional. In this paper we intend to evaluate the impact of unidirectional links on some of the existing distance vector routing algorithms for ad hoc networks. Based on the understanding of the impact of such links, we propose a strategy to modify existing algorithms so that they can work correctly in an ad hoc network that has a combination of unidirectional and bidirectional links. Evaluation of the impact of unidirectional links on hierarchical cluster-based routing algorithms and link-state routing algorithms is slated for future research.

Section 2 presents a brief description of some of the existing flat routing algorithms. As the focus of this paper is on such algorithms, we do not describe the hierarchical algorithms. In Section 3 we discuss the impact of unidirectional links on some of the existing algorithms for ad hoc networks. In Section 4 we prove that $O(n^2)$ size messages need to be exchanged between nodes to account for unidirectional links if distance vector based routing is employed. This is significantly greater than the $O(n)$ size messages exchanged in existing routing algorithms that assume all links to be bidirectional. We also propose an extension to distance-vector based routing algorithms. Finally, we present the conclusions in Section 5.

2 Previous Work

The *Destination Sequenced Distance Vector (DSDV)* [15] approach is a modification of the distance vector routing algorithm used earlier in ARPANET. In DSDV, each node maintains a distance vector that contains entries for each destination. The entry indicates the distance estimate and the next hop to be taken by a packet to reach a destination. Each entry has a sequence number associated with it, indicating its *freshness*. If a destination is unreachable, the distance metric is set to infinity. Periodically a node's distance estimates are diffused to neighbors. When a node p loses a link that it was using to forward packets meant for destination q, p sets its distance metric for q to infinity and propagates this information with a higher sequence number. Such updates are diffused immediately, without waiting for the next update time. Similarly, when a path is found to a hitherto unreachable node the finite distance metric to that destination is propagated immediately through the network.

Dynamic Source Routing (DSR) [9] uses a diffusion based mechanism to find a route to the destination. Instead of periodically exchanging routing information between nodes, route(s) are discovered when a node has to send packets to some destination node. During this process intermediate nodes can use the discovered routes to update their own routing information. Caching of recently discovered routing information is employed to speed up the routing process. The route maintenance mechanism (i) sends a *route error* packet to the source if it detects that the route to the destination is broken, and (ii) either tries to use any other cached route to the destination or invokes route discovery once again. In order to route packets, the source completely specifies the path the packet should take.

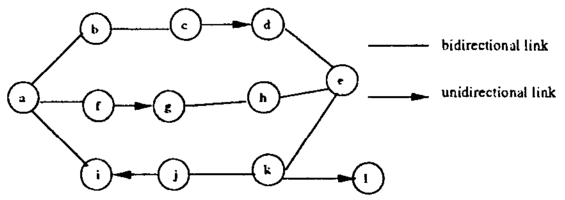

Figure 1: ad hoc network with unidirectional and bidirectional links.

In the *ad hoc On-Demand Distance Vector (AODV)* scheme [14], route discovery and maintenance are performed on demand, as in DSR, along with hop-based routing as in DSDV. In order to reduce communication overheads, as compared to DSDV, updates are propagated only along *active* routes, *i.e.*, routes that have seen some traffic in the recent past.

The *Temporally Ordered Routing Algorithm (TORA)* [12] is based on the notion of edge-reversal [7]. One instance of the algorithm is executed for each destination and a directed graph is maintained with respect to each destination. Only bidirectional links are considered, and a direction is associate with each link. Directed paths between every pair of nodes are initially determined through a sequence of edge reversals. When any node detects that it has lost the path to a destination (all edges incident on the node are directed towards it, in the graph for that destination) it performs full edge reversal so that it has only outgoing links to all its neighbors, and initiates route rediscovery for that destination. If a network partition is detected, the source is informed about the same.

3 Problem Description

Several flat routing protocols [12, 14, 15] and hierarchical routing protocols [1, 2, 5, 6] assume that all wireless links are bidirectional.[1] In the presence of unidirectional links several problems arise for distance vector based algorithms. For the purpose of illustration, let us consider DSDV [15]. AODV [14] has similar behavior. Other routing protocols may also exhibit similar problems.

Let us consider three interesting phenomena, illustrated with the help of the network configuration shown in Figure 1.

1. Knowledge Asymmetry: There is a two-hop path from j to a: jia. However, due to link $\overrightarrow{ji}$ being unidirectional, i cannot directly inform j about the path. *Just because i knows that j is*

[1] DSR [9] does not explicitly assume the presence of only bidirectional links.

its neighbor, i cannot assume that j also knows that i is its neighbor. Simple diffusion strategy may not be sufficient to propagate information about network topology.

2. Routing Asymmetry: In AODV, during the path discovery phase, let an intermediate node, v_i, get to know that the shortest path from x to y is $xv_1v_2 \ldots v_{i-1}v_iv_{i+1} \ldots y$. Then, v_i concludes that the shortest path from itself to x is $v_iv_{i-1} \ldots v_1x$: the lexicographical reversal of the path prefix ending at v_i. However, if there exists a unidirectional link on the path from x to v_i, then v_i's conclusion would be wrong. In Figure 1, as the link $\vec{ji}$ is unidirectional, the shortest path from i to j consists of seven hops and the path from j to i consists of one hop: *a routing asymmetry.*

3. Sink Unreachability: In DSDV path updates are initiated by the destination node. In AODV a source node finds a route to the destination only when a sequence of *route replies* flows back on the path from the destination to the source. In Figure 1, there exists a path to node l. So, it could be the destination of packets. However, there is no way node l can inform k that the latter can reach the former in one hop. So, reachability information about l cannot propagate to other nodes. Node l is a *sink* node as all its incident links are directed towards it. *The network topology may indicate that a sink is reachable from other nodes. But due to the limitations of the routing algorithm no node knows of the existence of the sink, making it effectively unreachable.*

In fact, the problem with DSDV and AODV in the scenario shown in Figure 1 is quite serious. As they can only use bidirectional links for routing purposes, they will ignore links $\vec{cd}$, $\vec{fg}$, $\vec{ji}$, and $\vec{kl}$. As a result, even though nodes a and e are reachable from each other, DSDV and AODV will perceive a and c to be in different network partitions.

In DSR, let i receive a path discovery message from j along $\vec{ji}$. When i has to send an acknowledgment to j it may need to initiate a new path discovery to find a route to j. The acknowledgment should then be sent along this route. Thus, while DSR does not ignore the possibility of unidirectional links, it makes an implicit assumption that routes in both directions always exist between a pair of nodes. Such an assumption *may not always be valid* in a network with a combination of bidirectional and unidirectional links.

4 Solution Approach

Each node needs to maintain enough information to distinguish between bidirectional and unidirectional links to its neighbors. A node may not be able to directly send information to a neighbor if there is no link from the node to the neighbor. Once knowledge of link orientations is available, appropriate routing decisions can be made.

First, let us determine the minimum amount of information participating nodes need to maintain to ensure correctness of the routing protocol. We will concentrate on modifications to protocols like DSDV and AODV to cope with the presence of unidirectional links.

4.1 Assumptions

We model the network as a graph G = (V, E), where V is the set of vertices and E is the set of edges. Some of the edges are assumed to be directed. Every vertex (also referred to as a node) is reachable from every other vertex. Thus, every node in the network can send packets to every other node in the network.

A node, on receiving a packet from some other node, can determine the length of the path taken by that packet. Let each packet start from the source x with its *Time_To_Live (TTL)* field initialized to TTL_max. All nodes have agreed *a priori* on the value of TTL_max. Each intermediate node z, and the destination y on receiving the packet decrements the TTL field by one. Let us refer to the resultant value as TTL_receive. When the packet arrives at the destination node the length of the path traversed by the packet thus far is equal to TTL_max - TTL_recv.

Definitions:

- *path*(ab): the shortest path from node a to node b. As some links are unidirectional, $path(ab)$ may be different from $path(ba)$.
- $path(av_1v_2 \ldots v_kb)$: the shortest path from a to b that passes through vertices $v_i : 1 \leq i \leq k$ such that v_i precedes v_j if $i < j$.
- $length(path(x))$: number of wireless links in $path(x)$, where x is a sequence of vertices.
- *directed path(ab)*: $path(ab)$ is said to be a directed path if it has at least one directed link.

Lemma 1 *$O(n)$ size distance vector exchange, as performed in protocols like DSDV, is not sufficient to determine routing information for distance vector based algorithms in the presence of unidirectional links.*

Proof: The lemma is proved by contradiction. Let us consider the graph G shown in Figure 2. In the figure:

1. $\vec{de}$ is a directed edge.
2. $length(path(cd)) \geq 0$.
3. $length(path(ef)) \geq 0$.

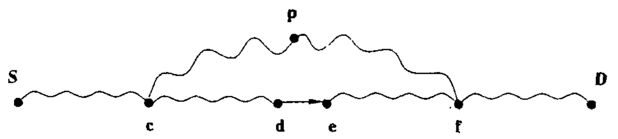

Figure 2: Representation of directed and undirected paths.

Let each node i maintain a vector V_i of length n such that V_i[j].dist is node i's knowledge of its path-length to node j. Let the shortest path from c to D be the directed path $path(cdefD)$ and let $path(fD)$ be an undirected path. Also, let $path(fpc)$ be a path of length greater than zero between f and c. There are two possibilities regarding this path:

1. It is a directed path from f to c,
2. It is an undirected path, or directed from c to f.

Possibility 1: As the distance vectors are exchanged between neighboring nodes, the reachability information about D reaches c along $path(Dfpc)$. Therefore, node c's estimate of the distance to D is: $length(path(Dfpc))$, which may be different from $length(path(cdefD))$.

Possibility 2: If $path(fpc)$ is directed from c to f, node c cannot learn about its distance to D as no path exists from D to c This is a violation of the assumption that every pair of nodes can communicate along a path.

Also, if $path(fpc)$ is undirected or directed from c to f, $length(path(cpf)) \geq length(path(cdef))$. Otherwise, the shortest path from c to D would have been $path(cpfD)$.

If $length(path(fpc)) > length(path(cdef))$, $V_c[D].dist = length(path(fpc)) + length(path(fD))$, which is greater than $length(path(cdefD))$. Hence, maintaining only a distance vector will lead to erroneous calculation of path-lengths. ∎

Lemma 2 *It is necessary to exchange $O(n^2)$ size matrices of pair-wise distance estimates to correctly construct path-length estimates for distance vector based algorithms.*

Proof: Let us once again refer to Figure 2. We assume that $path(cdef)$ is the shortest path from c to f. Let:

$X = \{x\colon x \text{ is a node on } path(cd)\}$, and
$Y = \{y\colon y \text{ is a node on } path(ef)\}$

As $path(cdef)$ is the shortest path from c to f, for all x and y, $path(xy)$ goes through vertices d and e. As edge $\vec{de}$ is directed, information about $length(path(xy))$ cannot propagate from y to x along the path that goes along $\overline{ed}$. Therefore, every node p on $path(yfcx)$ has to propagate $length(path(xy))$, $\forall x \in X$, $y \in Y$. As sets X and Y can be as large as V, $|X| = O(n)$ and $|Y| = O(n)$, where $n = |V|$.

Therefore, node p needs to store and forward $O(n^2)$ units of length information. ∎

4.2 Data Structures and Algorithm

It is assumed that each node emits a beacon at regular intervals. A node can hear beacons transmitted by a neighboring node provided the link between them is bidirectional, or directed from the neighbor to itself. The transmission of beacons by different nodes is not synchronized as there is no global clock in the system.

4.2.1 Data Structures

Each node p maintains the following data structures:

- $Nodesheard_p$: set of nodes whose beacons have been heard by node p within the last t time units. If $q \in Nodesheard_p$ and $p \in Nodesheard_q$, then there exists a bidirectional link between p and q. However, if $q \in Nodesheard_p$ and $p \notin Nodesheard_q$, then there is a unidirectional link from q to p. This data structure is modeled after the one by the same name used in the Linked Cluster Algorithm [1, 2].
- *D:* an $n \times n$ matrix of 2-tuples, where n is the number of nodes in the network. $D[i,j] = (seq,\ dist)$ means node p knows that the path from node i to node j if of length $dist$, and the sequence number associated with this information, pertaining to node j, is seq. Due to the possibility of unidirectional links, $D[i,j].dist$ may not be equal to $D[j,i].dist$. The sequence number associated with a destination is monotonically increasing. Each time a node sends updates to its neighbors, the node increases its sequence number by a constant value. As in AODV and DSDV, routing information with a higher sequence number overrides the corresponding information with a smaller sequence

number. As a result, stale routing information cannot suppress new routing information. Consequently, knowledge about link disruptions propagates quickly and the *count to infinity* problem (associated with distance vector algorithms) is avoided.

- *To* and *From:* vectors of length n, where each entry is a 3-tuple of the form (seq, $dist$, $next$) and (seq, $dist$, $prev$), respectively. The *To* vector is similar to the distance vector of DSDV as it maintains information about the path length from a node to all other nodes, and the next hop on the path to those nodes. $From_p$ vector contains information about paths from other nodes to p. Due to the presence of unidirectional links in the network, and the resultant routing asymmetry, the corresponding *dist* values in the *To* and *From* vectors may be different from each other. When routing information stabilizes, To_p should have the same *dist* and *seq* values as the corresponding entries in the p^{th} row of D_p. There should be a similar match between $From_p$ and the p^{th} column of D_p.

Determination of Link Orientation: We employ the Nodesheard set, in a manner similar to [1], to determine network adjacency. Each node periodically transmits its Nodesheard set with its beacon. It also continuously listens for similar transmissions from other nodes. If node p hears that $p \in Nodesheard_q$, node p knows that there exists a bidirectional link between p and q. The next time p broadcasts its beacon it includes q in its Nodesheard set. When q hears this beacon, it, too, knows of the presence of the bidirectional link.

If node p finds that $p \notin Nodesheard_q$, p concludes that there exists a unidirectional link from q to p. However, how does q get to know of the presence of this link? For this purpose we employ the matrix D.

4.2.2 Routing Algorithm

Let V denote the set of nodes in the network. Initially, the D matrix at each node p only contains its adjacency information. Each node periodically transmits its D matrix. The time between successive transmissions of D is a multiple of the time between successive transmissions of the $Nodesheard$ set. This is so for two reasons:

1. Transmission of D consumes much more bandwidth than the transmission of $Nodesheard$.
2. Transient noise that may interfere with the reception of a few successive $Nodesheard$ messages from a neighbor does not lead a node into erroneously concluding that its path to/from that neighbor is broken.

On link discovery: If p discovers a bidirectional link between p and q, then $D_p[p,q].dist = D_p[q,p].dist = 1$. If p discovers that there exists a unidirectional link from q to p, then $D_p[q,p].dist = 1$. The sequence number associated with each entry is analogous to the sequence number associated with routing table entries in DSDV and AODV. The sequence numbers are initialized to zero.

On receiving D matrix from neighbor: Let node p receive matrix D_{recv} from node q. If $\overline{pq}$ is a bidirectional link or a unidirectional link from q to p, p modifies its D matrix in the following manner on receiving the matrix:

- For all nodes $r \in V$, different from p and q:
  - If $D_{recv}[r,q].seq < D[r,p].seq$ then perform no action.
  - If $((D_{recv}[r,q].seq == D[r,p].seq)$ OR $((D_{recv}[r,q].seq > D[r,p].seq)$ AND $(From_p[r] != q)))$:
    - $D[r,p].dist = min(D_{recv}[r,q].dist + 1, D[r,p].dist)$
    - if $D[r,p].dist$ has decreased as a result then $From_p[r].prev = q$
  - If $((D_{recv}[r,q].seq > D[r,p].seq)$ AND $(From_p[r] == q))$:
    - $D[r,p].dist = D_{recv}[r,q].dist + 1$
  - If $D[r,p].dist$ has changed as a result, increment $D[r,p].seq$.
  - If $D_{recv}[r,q].seq == D[r,q].seq$
    - $D[r,q].dist = \min(D_{recv}[r,q].dist, D[r,q].dist)$
  - If $D_{recv}[r,q].seq > D[r,q].seq$
    - $D[r,q] = D_{recv}[r,q]$

  These operations enable node p to determine its distance from other nodes.

- For any arbitrary pair of nodes r and s in V, different from p and q:
  If $((D_{recv}[r,s].seq > D[r,s].seq)$ OR $((D_{recv}[r,s].seq == D[r,s].seq)$ AND $(D_{recv}[r,s].dist < D[r,s].dist)))$
  - $D[r,s] = D_{recv}[r,s]$

If link $\bar{pq}$ is a bidirectional link, node p also performs the following operations for all $r \in V$:

I. If $D_{recv}[q,r].seq < D[p,r].seq$, then do not perform any action using $D_{recv}[q,r]$.

II. If $D_{recv}[q,r].seq == D[p,r].seq$:
   - if $D_{recv}[q,r].dist + 1 < D[p,r].dist$:
     * $To_p[r].dist = D[p,r].dist = D_{recv}[q,r].dist + 1$,
     * $To_p[r].seq = D[p,r].seq = D_{recv}[q,r].seq$,
     * $To_p[r].next = q$
   - $D[q,r].dist = \min(D_{recv}[q,r].dist, D[q,r].dist)$.

III. If $D_{recv}[q,r].seq > D[p,r].seq$ then:
   - $To_p[r].dist = D[p,r].dist = D_{recv}[q,r].dist+1$,
   - $To_p[r].seq = D[p,r].seq = D_{recv}[q,r].seq$,
   - $D[q,r].dist = D_{recv}[q,r].dist$, and
   - $To_p[r].next = q$.

The preceding operations are similar to the updates performed by DSDV and AODV. They enable node p to determine its distance to other nodes.

If the received D matrix from node q is such that $D_{recv}[p,s].dist = 1$ and $s \notin Nodesheard_p$, node p concludes that there exists a unidirectional link from p to s. Therefore:

- $To_p[s].dist = D[p,s].dist = 1$
- $To_p[s].seq = D[p,s].seq = D_{recv}[p,s].seq$
- $To_p[s].next = s$

Also, for every arbitrary node r that is different from p and s, p updates its D matrix as follows:

- if $D[p,r].seq$ is equal to $D_{recv}[s,r].seq$ then updates are performed similar to case II described above, substituting q with s.
- if $D_{recv}[s,r].seq$ is greater than $D[p,r].seq$ then updates are performed similar to case III described above, once again substituting q with s.

Thus, each node updates its reachability information and propagates this information to other nodes.

On detecting link break: Let p's data structures indicate the existence of a bidirectional link between p and q, or a unidirectional link from q to p. If p does not hear a certain predetermined number of successive beacons from q, then p concludes that direct communication from q to p has been disrupted. Hence, p performs the following operations:

- increment $From_p[q].seq$ and $D[q,p].seq$
- $From_p[q].dist = D[q,p].dist = \infty$
- $From_p[q].prev = NULL$
- $\forall r$: $From_p[r].prev == q$:
  - $D[r,p].dist = \infty$
  - increment $D[r,p].seq$
  - increment $D[r,p].seq$
- Node p immediately broadcasts its updated D matrix to all its neighbors. The idea is to *propagate bad news fast.*

Example: Let us refer back to Figure 1. Node i knows that there is a path of length one from j to i. This information is forwarded by i, through a to the rest of the network. Later, when node j receives D_{recv} matrix from k, j finds that $D_{recv}[j,i] = 1$. It is at this point that j realizes that it has an outgoing link to i. Using this information, along with distance estimates from i to other nodes, j can revise its estimate of its distance to other nodes.

Also, when node b sends its D matrix to node c, c realizes that b is two hops away from i. Therefore, c concludes that it must be three hops away from i.

Lemma 3 *The algorithm for updating the D matrix and the To vector resuls in loop-free routing.*

Proof Outline: The proof is by contradiction. Let us assume that prior to an update of the D matrix and the To vector there is no loop. Therefore, $To[r].next$ values form a directed acyclic graph representing acyclic paths from nodes in V to node r. Such a directed acyclic graph can be constructed for each *destination* node. Let the following operation result in the formation of a cycle in node r's graph: $To_p[r].next = q$, where nodes q is a neighbor of node p. There are two cases when this update to $To_p[r].next$ is performed:

1. Node p gets to know that the sequence number of node q's path to r is greater than the sequence number of its own path to r. By construction of the algorithm, node $To_q[r].next$ should have a sequence number that is greater than or equal to q's sequence number. Extending this argument, as the chain of $To[r]$ pointers is traversed, the sequence number must be nondecresing. As we now have a cycle, the chain should lead back from q to p. This means that the $To_p[r].seq$ cannot be less than $To_q[r].seq$: a contradiction.

2. The sequence numbers associated with paths from p and q to r are the same. $To_p[r].next$ is set to q because $D_{recv}[q,r].dist+1 < D[p,r].dist$. As this has resulted in a cycle, the path from q to r must lead through p. This would imply that $D[p,r].dist < D[q,r].dist$: a contradiction. ∎

4.3 Storage and Communication Overheads

The storage requirement at each node is $O(n^2)$, where n is the number of nodes in the system. This is significantly greater than distance vector based protocols like DSDV and AODV which only require $O(n)$ units of information to be stored by each mobile node. The increased storage complexity of the proposed scheme is due to the topology matrix D maintained by each node. Similarly, the largest message is of size $O(n^2)$ data items, once again greater than the communication overheads of DSDV and AODV. The increased communication overheads also result in greater energy consumption for route maintenance.

4.4 Impact of Alternative Strategy on Route Stability

A pertinent question to ask at this juncture is: *Is it possible to reduce the storage and communication cost incurred in route maintenance for a network with potentially unidirectional links?*

One possibility could be to ignore all unidirectional links and restrict all operations to bidirectional links. As described in Section 3, this can lead to longer routes, or may lead to the impression that the network is partitioned when in reality all node pairs are reachable from each other. Also, links that are bidirectional most of the time may briefly become unidirectional. This may temporarily invalidate some routes. If one were to assume that the link has entirely disappeared for the duration it is unidirectional then this may: (i) invalidate an even greater number of routes for that period, (ii) generate more route update messages.

This will result in reduced *stability* of routes, where stability of a route between a pair of nodes indicates the duration for which the route remains unchanged. *It is to be noted that protocols like AODV and DSR cache routing information to reduce the overhead of route discovery. Reduced route stability will result in reduced effectiveness of caching, and shorter cache invalidation time.*

If link unidirectionality is a rare phenomenon and its impact on route length and stability is small, one could ignore all unidirectional links and only incur $O(n)$ storage and communication overheads. The reduction in overheads from $O(n^2)$ to $O(n)$ throughout the lifetime of the network may be more desirable than occasional increase in path lengths and reduction in route stability. However, an implementer should make the decision as to whether link unidirectionality needs to be considered or ignored only after careful interference modeling and extensive simulation experiments.

The observation that adjacent nodes need to exchange $O(n^2)$ information raises an interesting question: *Is there any fundamental difference in performance between distance vector based routing and link-state routing algorithms when unidirectional links are present in the network?* Link-state algorithms require a total of $O(n^2)$ information, *i.e.*, entire network topology to be conveyed to each router. Distance vector algorithm, as described earlier, would require $O(n^2)$ information to be sent along each incident edge of a node. Thus, the actual communication overhead would depend on the density of the network and the extent of dynamism in the network. However, further study is required before making any assertion about the superiority of one routing algorithm over the other in the ad hoc network scenario.

The presence of unidirectional links may also affect hierarchical routing algorithms. Unidirectional links may result in routing assymetry between cluster-heads. So, the m cluster-head ($m \leq n$) may have to exchange $O(m^2)$ information to maintain routes if the algorithm described in this paper is employed. However, once again, further investigation is required before reaching a conclusion.

5 Conclusion and Future Work

Most of the research in mobile computing tends to assume that all links are bidirectional. However, due to a variety of reasons, only unidirectional communication may be possible between some pairs of adjacent nodes. Existing distance vector based algorithms will fail in the presence of such links.

We described the adverse impact of unidirectional links on existing distance vector based routing algorithms. We also described simple data structures and proposed a strategy to propagate routing information in networks with a combination of unidirectional and bidirectional links. The proposed strategy is a modification of DSDV and AODV: well known routing algorithms proposed for wireless ad hoc networks. It incurs higher communication and storage overheads of $O(n^2)$. However, such overheads seem unavoidable for distance vector based routing approaches.

While it may not be possible to reduce the storage and communication complexity, we intend to work on efficient storage and information propagation strategies to reduce the absolute size of messages exchanged between neighboring nodes. This is of significance due to the low bandwidth of wireless links. Also, the $O(n^2)$ upper bound on route information maintenance points towards an evaluation of link-state routing strategies for networks with unidirectional links. In the future we intend to investigate the impact of unidirectional links on hierarchical routing algorithms. We will also try to gain a better understanding of the role of sink nodes in a network.

Acknowledgments

The author wishes to thank Mukesh Singhal for valuable comments.

References

[1] D. J. Baker and A. Ephremides. The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. *IEEE Transactions on Communications*, COM-29(11):1694–1701, November 1981.

[2] D.J. Baker, A. Ephremides, and J. A. Flynn. The Design and Simulation of a Mobile Radio Network with Distributed Control. *IEEE Journal on Selected Areas in Communications*, pages 226–237, 1984.

[3] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A Feedback Based Scheme for Improving TCP Performance in Ad-Hoc Wireless Networks. In *Proceedings of the* 18^{th} *International Conference on Distributed Computing Systems (ICDCS'98)*, pages 472–479. IEEE, May 1998.

[4] K. M. Chandy and J. Misra. The Drinking Philosophers Problem. *ACM Transactions on Programming Languages and Systems*, 6(4):632–646, October 1984.

[5] B. Das and V. Bharghavan. Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets. In *Proceedings of ICC*, 1997.

[6] B. Das, R. Sivakumar, and V. Bharghavan. Routing in Ad-Hoc Networks Using a Spine. In *Proceedings of IEEE IC3N*, 1997.

[7] E. Gafni and D. Bertsekas. Distributed Algorithms for Generating Loop-free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications*, pages 11–18, January 1981.

[8] IEEE. *P802.11. IEEE Draft Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, D2.0*, July 1995.

[9] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad-Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*. Kluwer Academic Publishers, 1996.

[10] T.V. Lakshman, U. Madhow, and B. Suter. Window-based error recovery and flow control with a slow acknowledgment channel: a study of TCP/IP performance. In *Proceedings of INFOCOM*. IEEE, 1997.

[11] S. Nesargi and R. Prakash. Distributed Wireless Channel Allocation in Networks with Mobile Base Stations. In *Proceedings of IEEE INFOCOM'99*, New York, N.Y., March 1999.

[12] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of IEEE INFOCOM*, April 1997.

[13] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.

[14] C. Perkins. Ad Hoc On Demand Distance Vector (AODV) Routing. Internet-Draft, draft-ietf-manet-aodv-02.txt, November 1998.

[15] C.E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of ACM SIGCOMM Conference on Communication Architectures, Protocols and Applications*, pages 234–244, August 1994.

[16] M. Singhal. A Dynamic Information-Structure Mutual Exclusion Algorithm for Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 3(1):121–125, January 1992.

[17] A. Tanenbaum. *Computer Networks(*3^{rd} *Edition)*. Prentice Hall, Upper Saddle River, N.J., 1996.

[18] J. Walter, J.L. Welch, and N. Vaidya. A Mutual Exclusion Algorithm for Ad Hoc Mobile Networks. In *Proceedings of the* 2^{nd} *International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL M for Mobility)*, Dallas, October 1998.