

# Local Leader Election, Signal Strength Aware Flooding, and Routeless Routing

Gilbert Chen, Joel W. Branch, and Boleslaw K. Szymanski  
Center for Pervasive Computing and Networking  
Rensselaer Polytechnic Institute  
110 8<sup>th</sup> Street, Troy, NY 12180, U.S.A.

## Abstract:

We have identified a fundamental operator in wireless networks that we named the local leader election problem in which the goal is to select a leader node in a spatially close neighborhood. We present a simple and elegant solution to the local leader election problem by making use of both: (i) implicit synchronization points, commonly observable by all nodes in the same neighborhood, and (ii) the prioritized backoff delay, dependent on the desired probability of each node becoming a leader. We then show that both flooding and routing are instances of the local leader election problem, so our general solution naturally applies. By relating the backoff delay to the signal strength of the received packet, we can design a variant of flooding named Signal Strength Aware Flooding (SSAF) that can improve its efficiency. By using a different metric to derive the backoff delay, we have designed a new generation wireless routing protocol, that we named the Routeless Routing protocol that possesses several interesting properties. It is free of route maintenance overhead, it allows nodes to conserve energy by independently switching their radios on or off, it is resilient to node or link failures, and it can also send packets via the shortest paths or un-congested paths.

## 1. Introduction

A distributed system is a collection of autonomous processes that communicate with each other, either synchronously or asynchronously. From time to time, these processes may need to select a common leader that can perform certain tasks on their behalf, such as coordination. Such a problem of electing a leader, called the *leader election* problem [1, 2], has become a fundamental problem in distributed systems, and it often serves as the basic building block for many other algorithms.

The leader election problem, in spite of simplicity of its definition, is by no means a simple problem. A correct solution must guarantee that there is eventually one and only one leader. Moreover, a correct solution may only exist when each process has a unique identifier, since it has been proven that it is impossible to elect a leader if all the processes are identical.

The leader election problem has been extensively studied in distributed systems. However, in large scale wireless networks, such as a futuristic but not unrealistic wireless

sensor network consisting of millions of tiny sensors [3], there would be almost no use of selecting a global leader, since algorithms designed for these wireless networks are usually localized in order to achieve good scalability [4]. What is more frequently encountered in this practice is the situation in which only a small group of nodes that can directly or indirectly communicate with each other need to select a local leader among them. We will refer to such a variant of this problem as the *local leader election* problem. In any geographical clustering based algorithm, for instance, a cluster head must be chosen among a set of nodes geographically close to each other. In other cases, the necessity of electing a local leader may be less obvious but existent nonetheless. For example, in some variants of flooding, after the source broadcasts a packet, one node (among all nodes that have received the packet) must be selected to forward the packet, and this node can be regarded as a local leader. Like the classic leader election problem, the local leader election problem also appears in other algorithms for wireless networks. For instance, in the token-based distributed mutual exclusion algorithm [5, 6], when the current token holder leaves the critical section, the token must be passed to a successor, and this successor is indeed a local leader among all other nodes that are competing for the token.

Unlike the traditional leader election problem, however, the local leader election problem in wireless networks may not require one and only one leader. How many leaders can or should be elected is determined by the particular nature of the requirements of wireless networks. Multiple local leaders may not affect the correctness, since redundancy sometimes is particularly sought. Failures to decide upon a local leader are not a problem either, for it is possible to repeat the leader election procedure until one or more leaders are established. Generally speaking, in wireless networks where individual nodes or communication links are not assumed to be completely reliable, it is not always appropriate to impose strict requirements on the outcome of any algorithm designed for them. Extra post-algorithm actions may be needed in order to ensure the correct termination of an algorithm.

Because of the broadcast-based nature of wireless communication, the local leader election problem has a simple yet effective solution, which forms the base of this paper. Such a solution may have a great influence on the design of many wireless network protocols. The next section gives a brief overview of related work, most of

which only pertains to the leader election problem in the classic sense. Section 3 introduces our solution in detail and provides a mathematical analysis of the efficiency of the simple solution. Section 4 applies this idea to improve the efficiency of flooding. Section 5 presents a new routing protocol for wireless networks based on this solution. This routing protocol, referred to as the Routeless Routing protocol, possesses an interesting property that the next hop is determined after the packet leaves the current hop, and therefore the route information is not stored anywhere in the network. Consequently, it responds to broken links or malfunctioning wireless nodes much quicker than other routing protocols, and more importantly, without any route maintenance overhead. The last section summarizes the contributions made by this paper and proposes future works that will be pursued.

## 2. Related Work

The classic leader election problem has been extensively studied in the context of radio networks [7, 8], where the communication channel is shared by all nodes, the time is slotted so that each packet transmission always starts at the beginning of a time slot, and the network is fully connected in the sense that any node can send packets directly to any other node. Such a network requires the synchronous version of the local leader election problem discussed here. However, since it is difficult, if not impossible at all, to synchronize clocks across wireless, the synchronous clock assumption cannot be satisfied in general.

Several methods for electing a global leader in wireless ad hoc networks have been proposed [9, 10]. The term “local leader election” has also been referred to as the problem of finding leaders in partitioned systems [11], where broken links cause the network to split into a number of disconnected sub-networks. One leader has to be elected for each connected sub-network. This problem is in fact a special case of the classic leader election problem, so its solution does not apply to the local leader election problem discussed in this paper.

## 3. Local Leader Election

Distributed algorithms for synchronous networks are simpler than those for asynchronous networks. The task of electing a local leader would have been much easier if it had been possible to precisely synchronize the clocks on individual wireless nodes. Unfortunately, such synchronization turns out to be quite difficult, in spite of many attempts [12-15]. Yet, there are many situations in which a common signal can be observed by many nearby nodes, such as the transmission of a packet, or the occurrence of a commonly observable event. In these situations, if each node records the moment at which such a signal is detected, then all the nodes that received the same signal are implicitly synchronized, and these moments are called *implicit synchronization points*. Here, we assume

that the propagation delay of the signal to each node can be ignored. These implicit synchronization points are valuable because all participating nodes are aware of them at the same moment. Hence, the nodes are synchronized (within the precision of the signal propagation delay) without the use of any explicit and costly synchronization protocols.

To take advantage of implicit synchronization points, every node must be instructed to wait a different amount of time before taking further actions. This delay, referred to as the *backoff delay*, has been widely used in CSMA protocols to avoid collisions resulting from multiple simultaneous transmissions. For example, when a node detects that the medium is free and wants to send a packet, it should not immediately start the transmission. Instead, it must wait until a randomly chosen amount of time elapses. The backoff delay is introduced to reduce the chance of a node transmitting the packet simultaneously with others. The origin of using the backoff delay for this purpose can be traced back to Ethernet [16]. It is also used in the IEEE 802.11 protocol [17].

The backoff delay is also utilized in network layer protocols, such as flooding, in which multiple receivers may relay the same packet. In flooding, after a packet is broadcast, it may be received by more than one neighbor. If every neighbor that has received the packet attempts to retransmit the packet immediately, a collision is bound to happen. To avoid this situation, every node must incur a small amount of delay, also randomly chosen, before retransmission. The backoff delay in such a case is also called the jitter time.

The most fundamental observation with an important consequence is that *backoff delays not only avoid collision, but also provide a precious opportunity to prioritize different nodes, enabling a simple solution to the local leader election problem*. A common implicit synchronization point followed by different backoff delays assigns different chances of becoming the leader to all participating nodes. Each node, after observing the implicit synchronization point, calculates a backoff delay based on a certain criterion and then sets a backoff timer with duration equal to the backoff delay. When the backoff timer expires, the node can start transmitting an *announcement* packet. However, if the node receives an announcement packet from another node before its own backoff time expires, it cancels the backoff timer. Thus, in most cases, only the node with the smallest backoff delay will succeed in transmitting the announcement packet, and naturally will become the local leader. Upon receiving the announcement packet, other nodes know that a new local leader has been elected.

This simple solution is not guaranteed to produce at least one local leader. Multiple nodes may choose almost identical backoff delays, leading to a collision. It may not guarantee only one local leader either, since the announcement packet sent by a node may be out of receiving range of some nodes, and these nodes will continue to broadcast new announcement packets. However, both cases may not present a problem. If there is no local leader elected at the first time, some upper layer protocol such as in the transport layer, may invoke the procedure repeatedly until there is a local leader. Multiple local leaders, as mentioned earlier, may be deemed correct for redundancy.

If the reliability of the outcome is desired, then an *arbiter* node can be chosen charged with sending an *acknowledgement* packet when it hears an announcement packet. The arbiter node may or may not be the same node whose packet transmission triggered the implicit synchronization point. However, it must be so chosen that every node involved in the local leader election is within its transmission range. If the arbiter node does not pick up any announcement packet within a predefined interval, it will trigger the implicit synchronization point again by sending the same packet. If it does, it will immediately broadcast the acknowledgement packet, upon the receipt of which other nodes will cancel their backoff timers, even if they have not received any announcement packet. Eventually there will be at least one local leader elected.

The use of an arbiter node, however, cannot completely avoid multiple local leaders, though it does reduce probability of their appearing. It may still happen that a successfully transmitted announcement packet may not be correctly received by the arbiter node.

The heart of the solution is how to derive the backoff delay based on a metric or a combination of several metrics so that the most desirable node would have the greatest chance of being the elected leader. As mentioned earlier, in CSMA protocols the backoff delay is usually randomly generated. Since the backoff delay actually represents the priority assigned to each node, a fully random choice unfortunately wastes the precious opportunity to prioritize different nodes as they compete for the local leadership. As we will see in the next two sections, a wide variety of metrics can be used to derive the backoff delay, each of which may lead to interesting solutions to the problems that seem to be unrelated to the local leader election problem at the first glance.

The use of the backoff delay as a priority value is not completely new. For example, they have been used to give nodes with more connectivity and more energy higher priority to become the coordinators in the Span protocol [18]. However, the identification of the implicit

synchronization points, and the use of the arbiter node, as well as the generalization of the local leader election problem described here, to the best of our knowledge, have not been attempted before.

#### 4. Signal Strength Aware Flooding

In any kind of network, whenever there is a need for two nodes to communicate with each other, yet there is no direct connection between them, routing becomes a necessity. The simplest routing protocol is flooding, primarily used when there is no a priori knowledge about the network. In the most basic form of flooding, every incoming packet is forwarded to the receiver's every neighbor, except the one from which the packet came.

In wireless networks, a packet broadcast by one node can be received by many neighboring node. To reduce the number of packet transmissions, one variant of flooding does not forward every packet that has been received. Only those that have not been received before need to be forwarded to the other nodes. For this purpose, every packet must contain a unique sequence number, in order to be distinguishable from other packets. Every node must also keep a list of sequence numbers of received packets, so that whenever a packet is received, its content is checked against this list. Only when the sequence number is new, will the packet be rebroadcast. This variant of flooding can be called counter-1 flooding [19].

When the node finds that a packet needs to be rebroadcast, it cannot do it immediately, for the reason discussed in the previous section. The transmission must start after a backoff delay to avoid collision. Traditionally, this backoff delay is randomly chosen.

In the light of the local leader election problem, the node that is chosen to forward the packet being broadcast is viewed as a local leader, so the solution to the local leader election problem naturally applies here. The end of the current packet transmission is an implicit synchronization point commonly known to all nodes that received the transmission. With properly selected backoff delays, nodes that are more appropriate to forward the packet can be given higher chances of becoming the local leader.

But which nodes are the most appropriate to forward the packet? The simple observation is that nodes furthest from the previous sender of the packet should be given higher priorities. This is the main idea of location-based flooding [19, 20]. However, location information is not generally available in wireless networks.

The main idea of Signal Strength Aware Flooding (SSAF) is to relate the backoff delay with the strength of the received signal. In general, the further the receiving node is from the sending node, the weaker the signal is. This is

true for large scale propagation models [21], such as the free space model or the two ray model. In small scale propagation models such as the Rayleigh model [21] and in practice [22], it might be the case that at some points the signal strength may vary dramatically due to multiple path interference. Yet even in these cases the weakening of the signal as the distance increases still holds at large scales. SSAF does not intend to precisely select the furthest node every time, but to choose nodes that are more likely to be further away from the sender.

A series of simulation-based experiments have been conducted to compare SSAF with counter-1 flooding. The wireless network being simulated is a sensor network consisting of 100 nodes distributed randomly in a 1000-meter by 1000-meter terrain. A new wireless network simulator called SENSE [23] was used to implement the simulation. 50 connections were selected between randomly chosen sources and destinations. Figure 1 compares SSAF to counter-1 flooding with respect to average delay, average number of hops, and delivery ratio, respectively. The average delay is the average time interval from the departure of a packet from the source to its arrival at the destination. The number of hops of a packet counts nodes traversed until the packets reaches its destination. The delivery ratio is calculated by dividing the number of packets sent by all sources by the number of packets received by all destinations.

In all the simulations, the free space propagation model was used, in which the signal strength is an inverse quadratic function of the distance. Interestingly, if the backoff delay is inversely proportional to the signal strength, it can be deduced that backoff delays obtained in this way are uniformly distributed. Therefore, in counter-1 flooding, we also chose a uniform distribution for the backoff delay with the same mean value as that used in SSAF. These choices were made merely to guarantee the fairness of the comparison.

As shown by the simulation results, SSAF consistently outperforms counter-1 flooding with respect to all three criteria. In most cases, SSAF obtains a slightly shorter end-to-end delay, but for smaller packet generation intervals, the gap becomes much more significant. This is because with more intense traffic, the queue between the network layer and the MAC layer tends to be more crowded. A priority queue favors those packets with a shorter backoff delay. Therefore, the prioritization takes effect not only among packets in different nodes, but also among packets in the same node. The priority queue has no effect on the counter-1 flooding.

However, the more interesting property of SSAF is its decreased average hop counts and increased delivery ratios. The average hop count is smaller in SSAF than in counter-1 flooding, since nodes further from the current hop are more likely to forward the packet. The delivery ratio is also higher, since the rebroadcast of a packet tends to cover a larger area that has not seen the packet before. Consequently, in SSAF there is a greater likelihood to establish shorter routes, when flooding is deployed to find paths between sources and destinations, which is the primary use of flooding in other, more complicated routing protocols.

## 5. Routeless Routing

Flooding, in spite of its simplicity, is not a practical solution for large-scale networks, since even in optimized flooding the number of packets transmitted is still in the same order as the total number of nodes in the network. Granted, at the beginning there is usually little knowledge about the network, so flooding is the only possible solution to routing. However, after a number of packets have been exchanged, these packets often carry some important information about the topology of the underlying network, which is completely ignored by flooding.

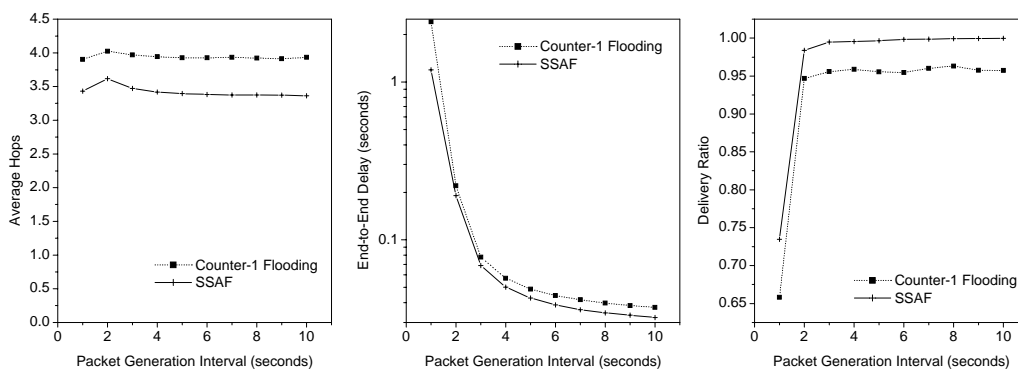


Figure 1. SSAF and Counter-1 Flooding

More efficient routing protocols must be developed in order to overcome the deficiency of flooding. Almost all existing routing protocols attempt to maintain one or multiple routes from the source and the destination. In wired networks, the routes can be constructed from link-state information or distance-vector information [24, 25]. In wireless networks, however, due to limited bandwidth and shared medium, wired routing protocols cannot be readily applied, and hence many new wireless routing protocols have been proposed. These wireless routing protocols can be classified as either *proactive*, such as DSDV [26], or *reactive*, (or on-demand) such as AODV and DSR [27, 28]. Moreover, a number of routing protocols have been developed specifically for wireless sensor networks, such as SPIN [29], LEACH [30], and Directed Diffusion [31].

For wireless networks with dynamic topological changes, however, the maintenance of routes could be very costly. A node on a route must be constantly aware of the status of the next hop on the route. Unlike wired links, failures on wireless links are difficult to detect quickly, and the links can also be temporally down due to burst interference. It is often the case that a considerable amount of time elapses before a node notices the unavailability of the next hop. When this happens, the node has to either report the route error, or actively repair it by discovering a new route or selecting an alternative route through other available candidates. The necessity of maintaining explicit routes also implies that nodes on the route cannot go to sleep at will to conserve energy; it is their responsibility to notify other nodes if they want to do so.

Most of these wireless routing protocols, in fact, essentially do not differ from wired routing protocols. They are merely equipped with facilities that handle better the dynamic nature of wireless links, and they do not take into consideration the fact that wireless communication differs from wired one in its shared media. The routing problem can be reduced to the local leader problem, as deciding how the next hop will be directed is equivalent to selecting a suitable local leader. To apply local leader election requires just deciding how to obtain the backoff delay. We have seen that SSAF derives the backoff delay from the signal strength. So the subsequent question is, what other metric can be used to derive the proper backoff delay?

Naturally, the backoff delay should be assigned in such a way that the closer a node is to the target node, the more likely it will be the next hop to forward the packet. This is similar to the basic idea behind SSAF that a node further away from the current hop should be given a higher chance to forward the packet. Therefore, the distance to the target node, measured in the number of hops, becomes an appropriate metric to calculate the backoff delay. This metric naturally gives rise to a new wireless routing

protocol, referred to as the *Routeless Routing* protocol, which is distinguished from many others by the fact that it never keeps explicit routes. Rather, the actual route is constructed in such a way that the next hop is always determined *after* the packet leaves the current hop.

### 5.1 Assumptions

We first list several assumptions that are required for Routeless Routing to obtain satisfactory performance. We assume that wireless links between neighboring nodes at most times are bidirectional. The existence of unidirectional links may negatively affect the efficiency, but not the correctness of the protocol. Furthermore, each node must have a hardware clock with a sufficiently small resolution. These hardware clocks neither do need to be very precise and accurate nor do they need to be synchronized.

Nodes may be stationary or mobile. As in AODV and DSR, we assume that, if nodes are mobile, their movements should not be so rapid that any routing protocol except flooding would be useless. However, Routeless Routing allows mobile nodes to move at higher speeds than permitted by AODV and DSR or any other routing protocols, for reasons explained later.

### 5.2 The Protocol

The data structure used by the Routeless Routing protocol is fairly simple. Each node maintains an *active node table*, each entry of which consists of (1) the identity of a target node (which is either a source or a destination) and (2) the number of hops from this target node to the node owning the table.

Similar to AODV and DSR, the Routeless Routing protocol starts with the *path discovery* process. Whenever a *source* node wants to send data packets to a *destination* node for the first time, it initiates a path discovery packet. This path discovery packet could be transmitted using flooding, or SSAF described in the last section. Each path discovery packet contains the identity of the source node, a sequence number to distinguish the packet from other path discovery packets originating from the same source, and the identity of the destination node. In addition, the packet discovery packet must have an *actual hop count* field that records the number of hops traveled from the source to the receiving node.

Once an intermediate node receives a path discovery packet, it will check if the same packet has been received before, by checking its source id and sequence number. If the packet is new, it will update its own active node table by either adding the source node found in the path discovery packet, or updating the number of the hops in the corresponding entry if the source exists in the table and the packet carries a smaller hop count from the source.

The node then attempts to forward the packet. It will first set a backoff timer upon the expiration of which the packet will be forwarded. This is to avoid collision with other nodes that received the same packet. Similar to counter-1 flooding, if the node receives another packet with the same source and sequence number before the backoff timer expires, it simply cancels the timer and does not relay the packet. The backoff delay is randomly calculated, although the use of SSAF, as discussed in the previous section, would definitely improve the efficiency of this part of the algorithm.

The destination node, upon receiving a new path discovery packet, will reply with a *path reply packet*, whose header contains the same fields as the path discovery packet had, as well as an *expected hop count* field indicating the expected number of hops needed for the packet to travel to reach the target node (in this case, the source). Unlike the path discovery packet, the path reply packet does not rely on flooding to find its return path back to the source. Neither does it use an existing path determined during the traversal of the path discovery packet, as AODV and DSR do.

The destination node simply broadcasts the path reply packet, without specifying which node the next hop is. Instead, it obtains the hop count to the source from the active node table, subtracts 1 and then puts the result into the *expected hop count* field in the path reply packet. Every node that detects the arrival of a path reply packet will first look at the *expected hop count* field of the path reply packet. Now, deciding the next hop becomes a local leader election problem, and the solution presented in Section 3 can be readily applied here. The central idea of the Routeless Routing protocol is to derive the backoff delay based on the known distance, measured by the number of intermediate hops, from the target node to the current hop. This idea is based on the rationale that the node closer to the target node should be given the higher priority to forward the packet. However, an intermediate node only knows the distance from the target node to itself, not the opposite, by passively listening to all packets and looking into the *actual hop count* field. This is why the assumption of bidirectional links is needed. A link in the forward path that is unidirectional may result in a longer return path, but cannot prevent the path reply packet from reaching the source.

After broadcasting the path reply packet, the destination node will continue to listen on the medium. If it captures the rebroadcast of the same packet by another node, it will immediately follow by transmitting an acknowledgement packet that contains the source id and the sequence number of the path reply packet. The acknowledgement packet serves the purpose of notifying those nodes not detecting

the rebroadcast that the packet has been relayed. If the rebroadcast is not overheard within a certain time, the destination node will retransmit the same packet. Here, the transmission of the path reply packet is the implicit synchronization point, while the node that transmits the packet (the destination node) is the arbiter node.

Having received a new path reply packet, a node determines the backoff delay, denoted by  $d_{backoff}$ , according to the following equation:

$$d_{backoff} = \begin{cases} \lambda \cdot (h_{table} - h_{expected}) \cdot U(0,1), & \text{if } h_{table} > h_{expected} \\ \frac{\lambda}{h_{expected} - h_{table} + 1} \cdot U(0,1), & \text{if } h_{table} \leq h_{expected} \end{cases}$$

Inside the equation,  $h_{table}$  is the number of hops to the target node known to this node which is looked up from the active node table,  $h_{expected}$  is the number of expected hops carried by the path reply packet, and  $U()$  is the generator of uniformly distributed random numbers.  $\lambda$  is a parameter that must be carefully chosen. If  $\lambda$  is too small, the difference between backoff delays calculated by different nodes will be too small to avoid collisions. A large  $\lambda$  would increase the end-to-end delay of packet delivery. The equation assigns a backoff delay larger than  $\lambda$  to nodes with a larger hop count than expected. The smaller  $h_{table}$  is, the smaller the backoff delay will be, and the more likely the node will succeed in transmitting the packet.

After obtaining the backoff delay, the node will set the backoff timer accordingly, which will be cancelled in two cases. In the first case, the node receives the same path reply packet again. In the second case, the node receives an acknowledge packet from the node from which it received the path discovery packet. In both cases, there is another node that has already relayed the packet, so the backoff timer needs to be aborted. When the backoff timer goes off without being cancelled, the node immediately starts transmitting the packet. Similar to the destination node, it will serve as the arbiter now to make sure that at least one subsequent node will relay the packet.

The path discovery process ends when the source node receives the path reply packet. The source now can add the destination into its active node table, since the path reply packet contains the number of hops it has traveled from the destination. It then must send another acknowledgement packet to indicate that the packet has reached the target, since otherwise other nodes would keep trying to retransmit the packet.

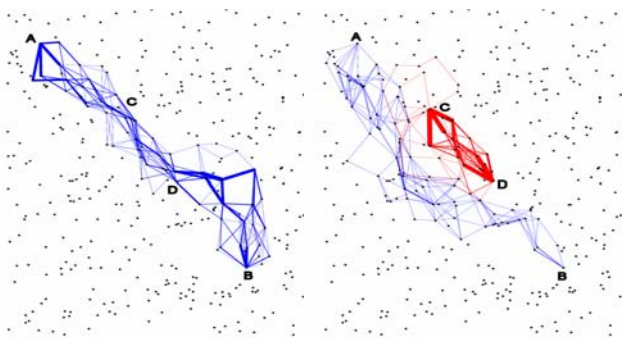
The source can now start transmitting data packets towards the destination. Data packets are transmitted and treated in the same way as path reply packets. Both data packets and path reply packets always keep track of the number of hops

that they have traveled, using the *actual hop count* field. Therefore, upon the receipt of either a data packet or a path reply packet, the receiving node can update the entry in its active node table corresponding to the node from which the packet originated.

### 5.3 Properties

The most salient feature of Routeless Routing is that it does not keep an explicit route, so that there is no need to constantly watch the route connectivity. As a result, *Routeless Routing can handle node or link failures without incurring any overhead of control packets*. In traditional routing protocols such as AODV and DSR, routes are explicitly maintained, which requires nodes to frequently check their connectivity. If a node on the route wants to go to sleep, it must inform others and pass to them the task of relaying packets. It is even more troublesome when a node or a link suddenly goes down, for it is difficult to quickly distinguish a temporarily breakdown from a permanent one. A substantial amount of time may elapse before it is known for sure what went wrong. In contrast, in Routeless Routing, when an established route encounters a node failure, other nodes will immediately take over and a new route will be quickly formed. The transition is seamless and no extra actions are needed. As a result, any node, even if it is on the route, can freely switch to the sleep or standby mode in order to save energy, making Routeless Routing well suited for energy limited sensor networks.

In Routeless Routing, data packets and path reply packets always carry the most up-to-date information about the distance from the originating node. Hence, *Routeless Routing can often choose the shortest paths to the destination*. In other routing protocols, such information could be made available to intermediate nodes. However, finding and adapting to shortest paths requires constant route changes, and the overhead of excessive routing maintenance may offset the benefit. It is Routeless Routing's ability to handle topology changes effortlessly that makes it capable of always looking for the shortest paths as well.

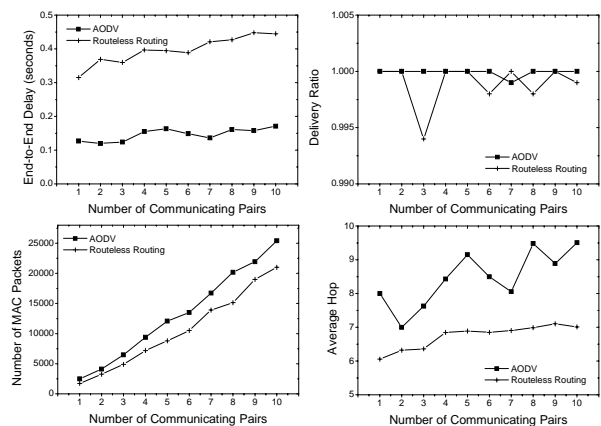


**Figure 2.** Automatic Congestion Avoidance in Routeless Routing

Another less obvious feature is that *Routeless Routing can automatically avoid congestion*. When an area is too crowded, the nodes within may contain too many packets waiting in the MAC queue to be transmitted. Even if such a node may be assigned a small backoff when relaying a packet, it is very likely that it will not be able to actually transmit it as quickly as nodes in less congested areas may. Figure 2 visualizes the actual paths taken by different packets in two simulations. On the left is the case in which there is one communicating pair with packets sent from node A to node B. On the right is the same visualization after another communicating pair with traffic flowing from node C to node D is introduced. Apparently, packets from A to B are smart enough to go around the congested area caused by the intensive traffic between C and D. Although it seems that they are now taking longer path to the destination, the end-to-end delay may drop. This is because the point-to-point delay incurred at each hop on the new, longer path is smaller than on the old, shorter path (otherwise packets would take the old path).

### 5.4 Simulation Results

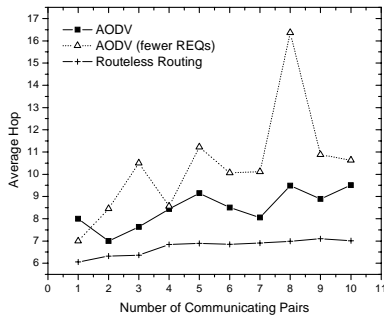
We have implemented Routeless Routing using SENSE [23]. Two sets of simulations have been conducted to compare Routeless Routing with AODV. The first set of simulations focused on the ideal conditions with no node failures, while the second one is more concerned with the performance of the protocols in presence of node failures. In all these simulations, the wireless network consists of 500 nodes distributed among a 2000 by 2000 meters terrain, with a transmission range of roughly 250 meters. The constant-bit-rate (CBR) model is used for the traffic pattern, with the traffic being bidirectional (i.e., in each pair of two communicating nodes, both nodes send packets to each other periodically).



**Figure 3.** Routeless Routing versus AODV in Absence of Node Failures

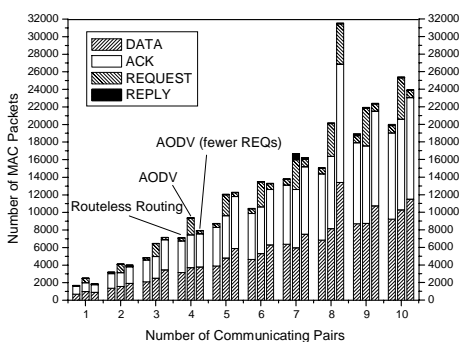
The number of sources varies from 1 to 10 in the first set of simulations. Figure 3 shows that Routeless Routing achieves roughly the same delivery ratio as AODV does,

while incurring larger end-to-end delays. This is understandable, as at each hop Routeless Routing takes more time in order to make the routing decision. Surprisingly, however, Routeless Routing requires fewer packet transmissions in the MAC layer, for two reasons. First, as shown in Figure 3, packets in Routeless Routing take on average fewer hops, because of the ability of Routeless Routing to find the shortest paths. Second, in this particular implementation of AODV, the route discovery procedure is based on original flooding, while in Routeless Routing counter-1 flooding is used, resulting in much fewer route request packets.

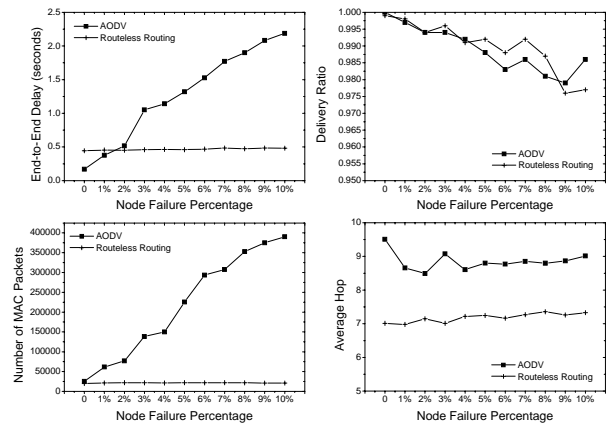


**Figure 4.** Fewer Route Request Packets Means Longer Routes in AODV

One may think that by optimizing the route discovery procedure, the total number of packet transmissions can be reduced in AODV. However, AODV is so much dependent on the quality of routes established at the first place that the reduction of the number of route request packets only increases the average length of routes (Figure 4), and, as a result, increases the total number of packet transmissions. Figure 5 shows the detailed breakdown of the number of different packets. If we look only at data packets that are much larger in size than other types of packets and thus more energy consuming, it can be seen that Routeless Routing generates fewer data packets than AODV in all but one case where there are 7 communicating pairs.



**Figure 5.** Routeless Routing versus AODV in Absence of Node Failures



**Figure 6.** Routeless Routing versus AODV When Nodes May Fail

In the second set of simulations, node failures are artificially introduced to turn off transceivers in all nodes but those that generate and receive CBR traffic. For instance, a node failure of 10% means that randomly selected 10% of the time the transceiver of a node is turned off and not able to transmit or receive any packets. It is apparent from Figure 6 that in order to guarantee the same delivery ratio, AODV has to use an order of magnitude more packets, while incurring much larger delays when there are frequent node failures. The end-to-end delays and the number of packet transmissions rise linearly with the probability of node failures, while in Routeless Routing they remain constant. These results confirm that Routeless Routing is completely resilient to node failures.

### 5.5 Similar Work

The previous work that is most similar to Routeless Routing is Gradient Routing [32]. However, in Gradient Routing only nodes with a smaller hop count to the destination are allowed to forward packets, which may cause a large portion of packets to be dropped. Moreover, every node with a smaller hop count may retransmit the same packet, resulting in a significant increase in the number of packet transmissions. In fact, the main drawback of Gradient Routing is that it makes the network more congested [32], which is not a problem for Routeless Routing.

Another similar protocol is Gradient Broadcast [33], which achieves reliability by building a band of interleaved paths. Again, such an approach congests the medium since multiple copies of the same packet exist along different paths. In contrast, Routeless Routing is more energy efficient, in terms of packet transmission, than AODV even when there are no node failures.

In [34] a routing protocol is briefly mentioned which selects the next hop “through the use of a deterministic

relay timer whose value is inversely proportional to the distance between sender and receiver". No more details are presented though.

## 6. Conclusion and Future Work

The discovery of the local leader election problem and its solution may have a significant impact on the protocol design for wireless networks. The local leader election solution can be applied to general protocol design in the following way. First, the protocol to be developed must be carefully analyzed to see if there is any instance of the local leader election problem, as such instances may not be apparent at the first glance. Next, implicit synchronization points must be identified, since they are valuable as they synchronize wireless nodes at no cost. Finally, an

appropriately chosen metric for deriving the backoff delays must be found to complete the solution.

SSAF and Routeless Routing are two examples of the application of the local leader election solution. Of these two, SSAF has been shown to be capable of improving the efficiency of flooding. This implies that in wireless networks, especially sensor networks that are bound by many constraints on energy, memory, and computing power, every bit of information is valuable and should not be wasted. Routeless Routing, on the other hand, represents the second-generation wireless routing protocols that do not attempt to maintain routes explicitly. The benefit of doing so is that it makes networks more adaptive to dynamic changes and therefore more fault-tolerant.

## Bibliography

1. Barbosa, V.C., *An introduction to distributed algorithms*. 1996, Cambridge, Mass.: MIT Press. xiii, 365.
2. Lynch, N.A., *Distributed algorithms*. The Morgan Kaufmann series in data management systems. 1996, San Francisco, Calif.: Morgan Kaufmann Publishers. xxiii, 872.
3. Kahn, J.M., R.H. Katz, and K.S. Pister. *Next century challenges: mobile networking for "Smart Dust"*. in *Proceedings of 5th Annual Joint ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99), 15-20 Aug. 1999*. 1999. Seattle, WA, USA: ACM.
4. Estrin, D., et al. *Next century challenges: scalable coordination in sensor networks*. in *Proceedings of 5th Annual Joint ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99), 15-20 Aug. 1999*. 1999. Seattle, WA, USA: ACM.
5. Baldoni, R., A. Virgillito, and R. Petrassi. *A distributed mutual exclusion algorithm for mobile ad-hoc networks*. in *Proceedings ISCC 2002 Seventh International Symposium on Computers and Communications, 1-4 July 2002*. 2002. Taormina-Giardini Naxos, Italy: IEEE Comput. Soc.
6. Walter, J.E., J.L. Welch, and N.H. Vaidya, *A mutual exclusion algorithm for ad hoc mobile networks*. *Wireless Networks*, 2001. **7**(6): p. 585-600.
7. Willard, D.E., *Log-logarithmic selection resolution protocols in a multiple access channel*. *SIAM Journal on Computing*, 1986. **15**(2): p. 468-477.
8. Nakano, K. and S. Olariu, *Uniform leader election protocols for radio networks*. *IEEE Transactions on Parallel and Distributed Systems*, 2002. **13**(5): p. 516-26.
9. Vasudevan, S., et al. *Leader election algorithms for wireless ad hoc networks*. in *Proceedings DARPA Information Survivability Conference and Exposition, 22-24 April 2003*. 2003. Washington, DC, USA: IEEE Comput. Soc.
10. Malpani, N., J.L. Welch, and N. Vaidya. *Leader election algorithms for mobile ad hoc networks*. in *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Aug 11 2000*. 2000. Boston, MA, United States: Association for Computing Machinery.
11. Fetzer, C. and F. Cristian, *A highly available local leader election service*. *IEEE Transactions on Software Engineering*. **25**(5): p. 603-18.
12. Elson, J., L. Girod, and D. Estrin. *Fine-grained network time synchronization using reference broadcasts*. in *2002 Usenix Symposium on Operating Systems Design and Implementation (OSDI'02), 9-11 Dec. 2002*. 2002. Boston, MA, USA: USENIX Assoc.
13. Ganesan, D., et al., *Coping with irregular spatio-temporal sampling in sensor networks*. *ACM SIGCOMM Computer Communication Review*, 2004. **34**(1): p. 125-130.
14. Sichitiu, M.L. and C. Veerarittiphan. *Simple, accurate time synchronization for wireless sensor networks*. in *WCNC 2003 - IEEE Wireless Communications and Networking Conference, 16-20 March 2003*. 2003. New Orleans, LA, USA: IEEE.

15. Van Greunen, J. and J. Rabaey. *Lightweight Time Synchronization for Sensor Networks*. in *Proceedings of the Second ACM International Workshop on Wireless Sensor Networks and Applications, WSNA 2003, Sep 19 2003*. 2003. San Diego, CA, United States: Association for Computing Machinery.
16. Boggs, R.M.M.a.D.R., *Ethernet: distributed packet switching for local computer networks*. *Communications of the ACM*, 1976. **19**(7): p. 359--404.
17. Crow, B.P., et al., *IEEE 802.11 Wireless Local Area Networks*. *IEEE Communications Magazine*. **35**(9): p. 116-26.
18. Chen, B., et al., *Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks*. *Wireless Networks*, 2002. **8**(5): p. 481-94.
19. Tseng, Y.-C., et al., *The broadcast storm problem in a mobile ad hoc network*. *Proceedings of 5th Annual Joint ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99)*, 15-20 Aug. 1999, 2002. **8**(2-3): p. 153-67.
20. Williams, B. and T. Camp. *Comparison of broadcasting techniques for mobile ad hoc networks*. in *MOBIHOC 2002. Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing, 9-11 June 2002*. 2002. Lausanne, Switzerland: ACM.
21. Rappaport, T.S., *Wireless communications : principles and practice*. 2nd ed. 2001, Upper Saddle River, N.J. London: Prentice Hall PTR. xxiii, 707.
22. Govindan, J.Z.a.R. *Understanding packet delivery performance in dense wireless sensor networks*. in *Proceedings of the first international conference on Embedded networked sensor systems*. 2003. Los Angeles, California, USA: ACM Press.
23. Chen, G., et al., *SENSE: A Sensor Network Simulator*, in *Advances in Pervasive Computing and Networking*, B. Szymanski and B. Yener, Editors. 2004, Springer. p. 249-267.
24. Abolhasan, M., T. Wysocki, and E. Dutkiewicz, *A review of routing protocols for mobile ad hoc networks*. *Ad Hoc Networks*, 2004. **2**(1): p. 1-22.
25. Tanenbaum, A.S., *Computer networks*. 3rd ed. 1996, Upper Saddle River, N.J.: Prentice Hall PTR. xviii, 813.
26. Perkins, C.E. and P. Bhagwat, *Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers*. *Computer Communication Review, ACM SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, 31 Aug.-2 Sept. 1994, 1994. **24**(4): p. 234-44.
27. Maltz, D.B.J.a.D.A., *Dynamic source routing in ad hoc wireless networks*, in *Mobile Computing*, T.I.a.h. Korth, Editor. 1996, Kluwer Academic Publishers.
28. Perkins, C.E. and E.M. Royer. *Ad-hoc on-demand distance vector routing*. in *Proceedings of WMCSA99: 2nd IEEE Workshop on Mobile Computing Systems and Applications, 25-26 Feb. 1999*. 1999. New Orleans, LA, USA: IEEE Comput. Soc.
29. Heinzelman, W.R., J. Kulik, and H. Balakrishnan. *Adaptive protocols for information dissemination in wireless sensor networks*. in *Proceedings of 5th Annual Joint ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99)*, 15-20 Aug. 1999. 1999. Seattle, WA, USA: ACM.
30. Heinzelman, W.R., A. Chandrakasan, and H. Balakrishnan. *Energy-efficient communication protocol for wireless microsensor networks*. in *Proceedings of HICSS33: Hawaii International Conference on System Sciences, 4-7 Jan. 2000*. 2000. Maui, HI, USA: IEEE Comput. Soc.
31. Intanagonwiwat, C., R. Govindan, and D. Estrin. *Directed diffusion: a scalable and robust communication paradigm for sensor networks*. in *Proceedings of MobiCom 2000. Sixth Annual International Conference on Mobile Computing and Networking, 6-11 Aug. 2000*. 2000. Boston, MA, USA: ACM.
32. Poor, R.D., *Gradient Routing in Ad Hoc Networks*.
33. Ye, F., et al., *GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks*, <http://irl.cs.ucla.edu/papers/grab-winet.ps>.
34. Min, R. and A. Chandrakasan, *Top Five Myths about the Energy Consumption of Wireless Communication*. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2003. **7**(1): p. 65-67.