Kinetic Spanning Trees for Minimum-Power Routing in MANETS

Camillo Gentile and Robert E. Van Dyck
National Institute of Standards and Technology
Wireless Communications Technologies Group
Gaithersburg, MD 20899
{cgentile,vandyck}@antd.nist.gov

Abstract–**A distributed kinetic spanning tree algorithm is proposed for routing in wireless mobile ad hoc networks. Assuming a piecewise linear motion model for the nodes, the sequence of shortest-path spanning trees is determined, valid until the time of the next node trajectory change. By computing the sequence of trees using one execution of the distributed routing algorithm, in contrast to computing the tree for a single time instant, the number of routing messages is substantially reduced. Moreover, the total power required to route through the trees as a function of time is also lower.**

Keywords–**Kinetic spanning trees, Wireless ad hoc networks, MANET**

I. Introduction

In a mobile ad hoc network (MANET), it is often necessary to route data in such a way as to minimize power consumption. For routes to a specific sink node, one can construct the shortest-path spanning tree [1], where the cost of each link is based on the power required. In such a tree, each node maintains in its forwarding database the next node in the tree. Because the nodes are moving, there are discrete times at which the present spanning tree is no longer optimal, and a new shortest-path spanning tree should be used. This tree is typically updated using a distributed algorithm *cf.* [2], [3], [4], and it is important that the nodes be able to determine when to change their forwarding databases. To do so, messages must be exchanged among neighboring nodes.

We propose a distributed algorithm that adapts techniques from the theory of kinetic spanning trees [5], [6] to maintain the correct sequence of shortest-path spanning trees. Our method minimizes the number of necessary routing messages to provide more throughput for the data, at the price of increased computation, exploiting the fact that the energy cost of computation is much less than the cost of message transmission [7]. Metrics include the power used to transmit over the tree, the number of routing messages, and the iterations required to achieve convergence. Here, the power cost for transmission between two nodes varies as a square of the distance, although the proposed algorithm also works for other cost functions.

[0]This work was supported by the Advanced Research and Development Activity (ARDA) under contract number 706400.

II. Problem Statement

Consider the nodes in a mobile ad hoc network. Over a relatively short period of time[1], one can assume that each such node follows a linear trajectory. Its position as a function of time is described by

$$\mathbf{x}_i(t) = \begin{bmatrix} x_{i,0} + \dot{x}_i t \\ y_{i,0} + \dot{y}_i t \end{bmatrix}, \tag{1}$$

where the vector $(x_{i,0}, y_{i,0})^T$ gives the initial position of node i, and the vector $(\dot{x}_i, \dot{y}_i)^T$ gives the velocity.

The *squared distance* between two nodes i and j is given simply by

$$\begin{aligned} D_{ij}^2(t) &= \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|_2^2 \\ &= a_{ij}t^2 + b_{ij}t + c_{ij}, \end{aligned} \tag{2}$$

where $a \geq 0$, $c \geq 0$, and $\min_t\{D_{ij}^2(t)\} = c - \frac{b^2}{4a} \geq 0$ is the minimum squared distance in time.

Definition

The *power* as a function of time, required to transmit between nodes i and j, is defined as $P_{ij}(t) = P_{ji}(t) = \kappa D_{ij}^2(t)$, for some constant κ; without loss of generality, we presently assume $\kappa = 1$. We choose power as our cost, since by minimizing this quantity through multi-hop paths, one can preserve battery life.

III. Kinetic Spanning Trees

The proposed distributed algorithm bears resemblance to the asynchronous distributed Bellman-Ford (BF) algorithm [1] for computing shortest-path spanning trees. With each execution cycle, the BF algorithm reduces the cost of the minimum multi-hop route from node i to the sink node S through other nodes that comprise this route. The proposed algorithm likewise reduces the costs of the minimum multi-hop routes, however for all time for which the fixed trajectories are valid rather than for a single time instant. Whenever any node changes trajectory, it simply informs its neighbors, thereby starting a new execution cycle of the algorithm.

We assume that all nodes operate with a synchronous clock, whose unit of time is that required to transmit a data packet from a node to a neighboring node. Please note that the actual message exchanges are asynchronous in the sense that any node

[1]The time required to transmit a data packet is orders of magnitude shorter than the time the node is moving along a fixed trajectory.

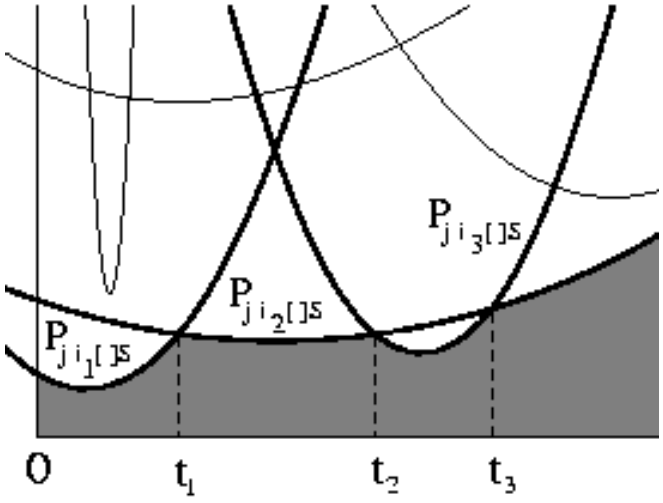

Fig. 1. New and current power costs in the pruning step at node j.

can transmit at any time (in the larger time scale). Moreover, each node need not transmit at any given time step.

A. Distributed Kinetic Spanning Tree Algorithm

Initialization

1. Each node i in the network computes the cost as a function of time, $P_{iS}(t)$, to the sink node, S, and retains this *current minimum cost* in its forwarding database.
   We assume that each node can transmit to and receive from all the other nodes that are in its range as determined by the RF transmitter power of a transmitting node and the sensitivity of the receiver at the receiving node. If S is not in range, then $P_{iS}(t) = \infty$.
2. Node i computes and distributes the *new* first time cost $P_{jiS}(t) = P_{ji}(t) + P_{iS}(t)$, from node j routing through i to the sink S, $j \neq i, S$ (*i.e.* to its neighboring nodes in the network).

Iteration Step

1. At a given time step, node j receives new costs $P_{ji_1[\,]S}(t), P_{ji_2[\,]S}(t), \cdots, P_{ji_n[\,]S}(t)$, from nodes $i_1, i_2, \cdots, i_n$, which computed them at the previous iteration; [] denotes the ordered nodes on a multihop route between i_l and S.
2. Pruning step: The new costs and the current costs (*i.e.* those in the forwarding database of j from the previous steps) are compared amongst each other. Only those (minimum) costs that contribute to the minimum routes of the node in time are retained in the forwarding database.
   Figure 1 visualizes both the new and current competing power costs in the pruning step at node j. The union of new and current costs appears as the six parabolic functions in time. However, only three of them, namely $P_{ji_1[\,]S}(t)$, $P_{ji_2[\,]S}(t)$, and $P_{ji_3[\,]S}(t)$, contribute to the minimum cost in time of node j. This quantity is indicated by the shaded area. $P_{ji_1[\,]S}(t)$, $P_{ji_2[\,]S}(t)$, and $P_{ji_3[\,]S}(t)$ form the forwarding database for node j at this iteration as follows: node j forwards to node i_1 for $0 \leq t < t_1$, to node i_2 for $t_1 \leq t < t_2$, to node i_3 for $t_2 \leq t < t_3$, and to node i_2 for $t_3 \leq t$.
3. For only the new minimum costs, we compute and distribute the costs, $P_{kj[\,]S}(t) = P_{kj}(t) + P_{j[\,]S}(t)$, to node k, $k \neq j, [\,], S$. Note that the costs from previous steps were already transmitted in those steps, and need not be retransmitted.

The distributed algorithm ceases when no new minimum power costs arise at a given iteration, and so no packets need be further transmitted. At this stage, the forwarding database of each node indicates the minimum cost next-hop for times for which the fixed trajectories are valid.

We note that the number of routing packets decreases exponentially with each iteration step, both because a node can not retransmit to nodes already on its route to the sink, and because the majority of costs will not become minimum costs after the pruning step. In fact, the proposed distributed algorithm carries the same complexity and number of transmissions as the BF algorithm for shortest-path spanning trees; however, the proposed algorithm requires more computation at each node per iteration.

B. Convergence Issues

As explained above, the proposed algorithm is quite similar to the distributed Bellman-Ford algorithm, except that a, b, and c of $P_{ji[]S}(t)$ are exchanged from node i to node j at time t^*; in the BF algorithm, rather $P_{ji[]S}(t^*)$ is exchanged. The proof of convergence is essentially identical in both cases and depends on the stability of the link costs. In this paper, we assume the costs are changing according to the linear trajectories. That is, the costs correspond to the required transmitter powers, which are quadratic. As $t \to \infty$, each link cost is dominated by at^2; dividing by t^2 gives fixed costs as in the BF algorithm, and so our algorithm follows the same convergence proof [1][pp. 404-409]. In Section V, we compare the average number of iterations required to achieve convergence for one execution cycle of each algorithm.

C. Computational Complexity

The kinetic spanning tree algorithm reduces the number of routing messages exchanged, at the cost of increased computation at each node. Since computation is generally much cheaper than communication and throughput costs [7], this proves a good trade-off. To further reduce the computation at each node, one can use ideas from computational geometry as applied to kinetic spanning trees [5], [6]. In most of this literature, the costs, $W_{ij}(t)$, are linear, but it is relatively straight-forward to extend them to our quadratic costs, $P_{ij}(t)$.

IV. Example Network

Figure 2 shows the shortest-path spanning trees routed at node S for a simple five-node network at two distinct times; the solid arrows indicated $t = 0$, and the dashed arrows indicate $t = 8.378$. Table I shows the trajectories for each node.

TABLE I
NODE TRAJECTORIES.

Node	Trajectory
S	$(0, 0)$
A	$(-1 - 0.1t, 1)$
B	$(-0.4t, 1.5 - 0.2t)$
C	$(2 - 0.5t, 1 + 0.2t)$
D	$(0, 2 + 0.3t)$

TABLE II
CHANGES IN THE SHORTEST-PATH SPANNING TREE.

t	Change
1.252	A $\rightarrow$ B
2.193	D $\rightarrow$ C
3.754	A $\rightarrow$ S
4.085	B $\rightarrow$ A
5.692	C $\rightarrow$ A
8.378	D $\rightarrow$ A
8.452	D $\rightarrow$ S

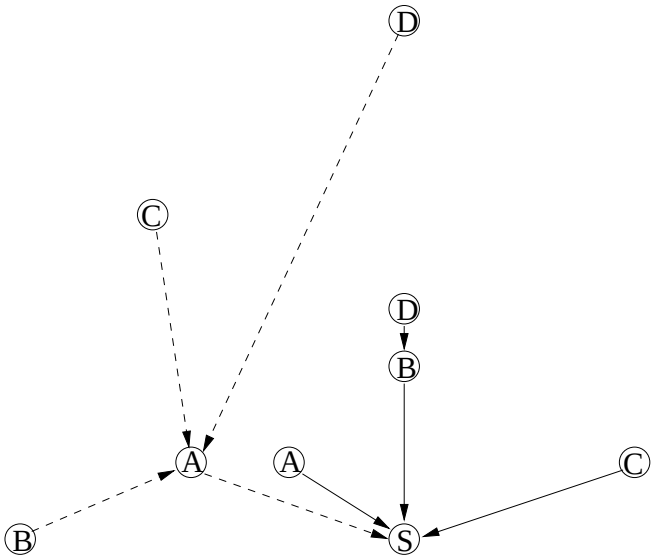

Fig. 2. Example network at $t = 0$, and $t = 8.37$.

We assume that these trajectories are valid for ten seconds. At time $t = 0$, node A transmits directly to the source. Node A is moving slowly to the left, while node B is moving more quickly down and to the left. So at time $t = 1.252$, it is more efficient for node A to route through node B, as shown in Table II. Similarly, at time $t = 2.193$, node D begins to route through node C.

Only two iterations are require for our distributed algorithm to converge, thereby providing all the nodes with the a, b, and c coefficients required to calculate the sequence of eight shortest-path spanning trees for the time interval $0 \leq t \leq 10$ seconds. While the distributed Bellman-Ford algorithm also converges in two iterations, it gives a shortest-path spanning tree for a single time instant. One can run multiple cycles of the BF algorithm in order to obtain the sequence of shortest-path spanning trees; yet, since the times where the shortest-path spanning tree changes are not known *a priori*, the BF algorithm would have to be run at a very high frequency to closely maintain the sequence. Specifically, the BF algorithm would need to be run at each time unit that corresponds to the greatest common divisor of the transition times.

V. SIMULATION RESULTS

To show the utility of the kinetic spanning tree algorithm, consider a network of N nodes spread randomly over a 10×10 mile area at initialization. The number of nodes varies as 20, 50, 100, or 200 by simulation. We assign to each node a random velocity, whose speed does not exceed 60 miles/hour. We assume that the underlying communication links require 100 ms to transmit a packet across a single link and set the unit time step accordingly. In the first set of simulations each node maintains a fixed trajectory, while in the second set the probability that some node randomly changes trajectory is uniformly distributed between zero and ten seconds. Hence in the latter case there is a trajectory change on average every five seconds.

A. Fixed Trajectories

Since the proposed algorithm computes the sequence of spanning trees, the number of messages required to achieve convergence for one execution is slightly higher than for the original BF algorithm. Figure 3 shows the ratio of messages for the two algorithms as a function of the transmission radius, when each algorithm is executed once at time $t = 0$. It is clear from the figure that the ratio is close to one over a range of transmission radii and for the four different network sizes. The number of iterations required to achieve convergence for one execution cycle of the proposed algorithm is equal to the maximum number of iterations required by the BF algorithm to achieve convergence, where the maximum is over all spanning trees. Depending on the update period of the Bellman-Ford algorithm, the total number of messages required may be significantly higher, as shown below[2].

Figure 4 shows the the power ratio of the BF algorithm to the proposed algorithm as a function of the update period. The power ratio is the total power costs of all spanning tree links in the Bellman-Ford algorithm divided by the total power of all spanning tree links in the proposed algorithm. For each algorithm, the total power is determined by integrating the cost

[2]Every data point in this subsection is the average of 100 trials, where the initial position and velocity of each node are chosen randomly for each trial.

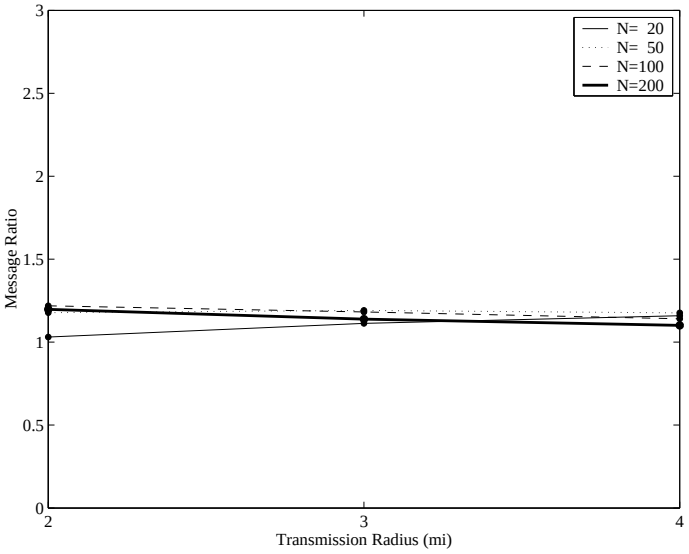

Fig. 3. Ratio of messages required for algorithm convergence vs. transmission radius.

per unit time over time[3]. The update period determines how often the BF algorithm is executed; the kinetic spanning tree algorithm requires a single execution at $t = 0$.

The power ratio for a transmission radius of two miles is shown in Figure 4(A). Consider a 200-node network for an update period of 5 seconds: the increase is no more than 3.5%. Yet when the update time is increased to 60 seconds, one can achieve a power ratio of 508%. For some applications, such as multicasting of video to a large number of nodes, this increase is quite substantial. Figure 4(B) shows the ratio for a four-mile transmission radius. Here, the number of hops across the entire network is greatly reduced, yet similarly large ratios are still achieved. For example, a 100-node network with an update time of 20 seconds yields a degradation of 26.6%.

It is interesting to examine the trade-off between power efficiency and routing message overhead. Figure 5 shows the power ratio vs. the message ratio, which is defined as the total number of routing messages used in all executions of the Bellman-Ford algorithm, over a period of one minute, divided by the total number of routing messages used by the kinetic spanning tree algorithm. By increasing the period between executions of the distributed Bellman-Ford algorithm, one reduces the total number of routing messages sent. However, the spanning trees depart from optimal, and more power is required. For the 20-node network, there is not much improvement over the BF algorithm since the network is essentially unconnected, allowing few, if any, routing choices, and so a small number of total shortest-path trees. For a larger network with more choices, using the true shortest-path trees gives a substantial improvement.

[3]The integral is only over the first update period: *i.e.* $0 \leq t < 5$s, or $0 \leq t < 10$s, *etc.*

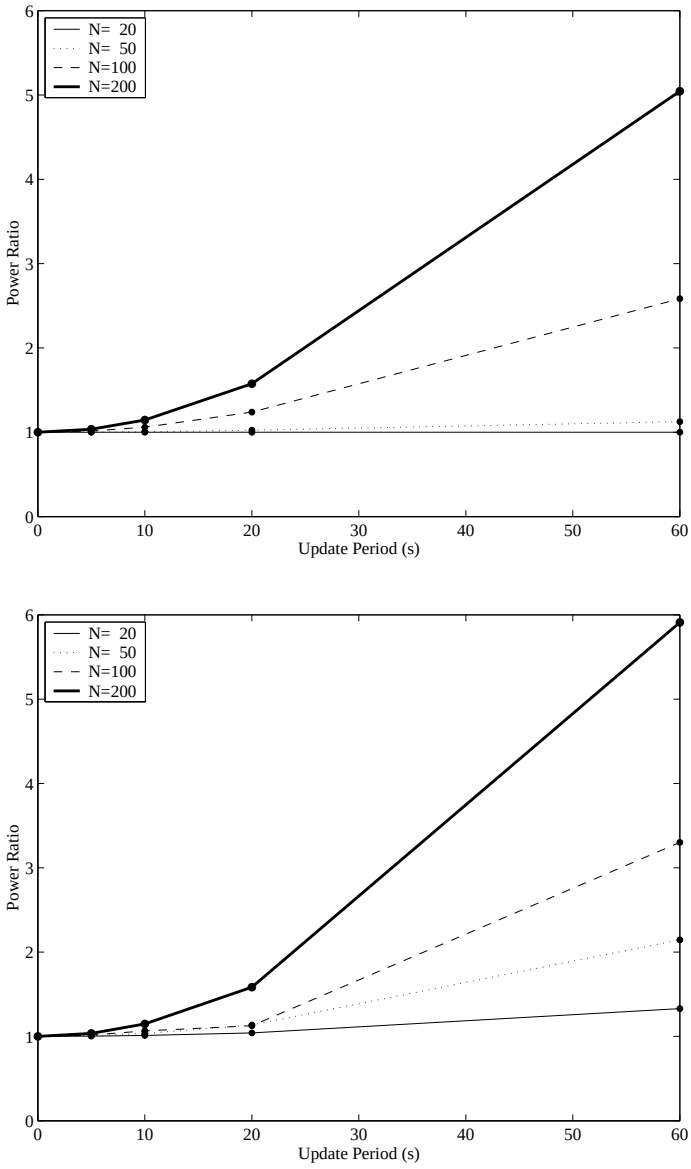

Fig. 4. $\boxed{\frac{(A)}{(B)}}$ Power ratio vs. time between executions of the distributed Bellman-Ford algorithm. (A) Transmission radius equals 2. (B) Transmission radius equals 4.

B. Randomly Changing Trajectories

In this subsection, the nodes are allowed to randomly change their trajectories (speed and/or direction). The time between the change in the trajectory of any node is uniformly distributed between zero and ten seconds. Again, the Bellman-Ford algorithm is executed periodically, while the kinetic spanning tree algorithm is executed only at the initial time and whenever there is a trajectory change. The simulation is run for 500 seconds.

Figure 6 shows the message ratio as a function of the update period of the BF algorithm. The message ratio is the total number of routing messages used by the BF algorithm divided by the total number used by the proposed algorithm (including trajectory changes). For all four network sizes, there is a sizeable

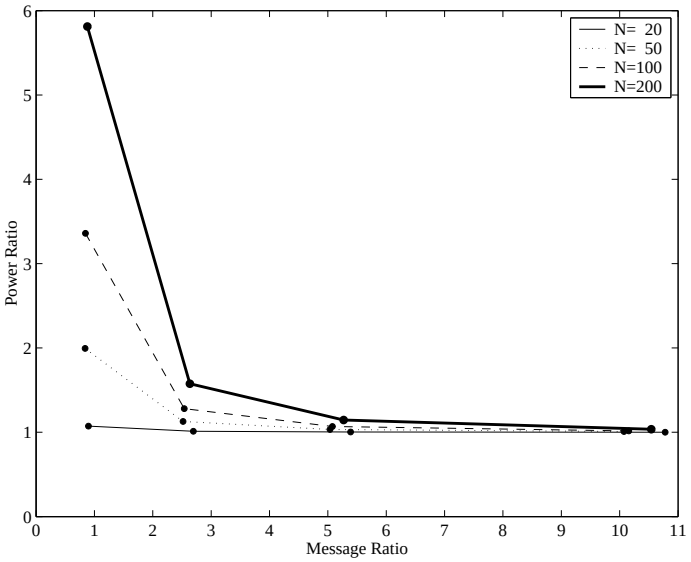

Fig. 5. Power ratio vs. message ratio.

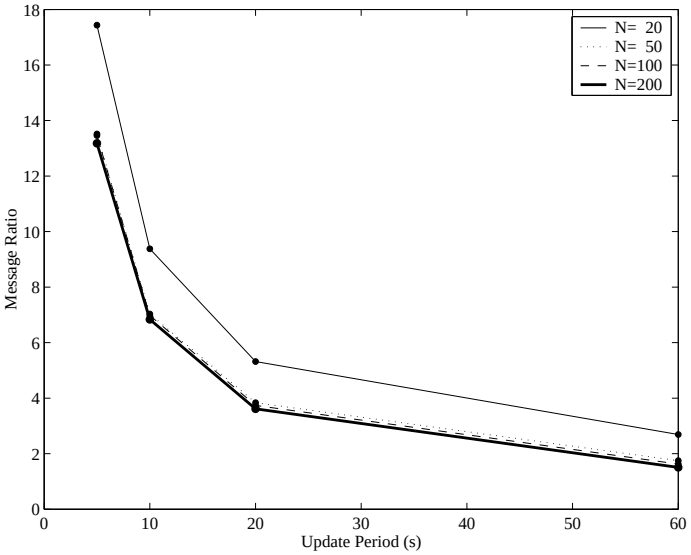

Fig. 6. Message ratio vs. the time between executions of the distributed Bellman-Ford algorithm. Transmission radius equals 2.

reduction in the number of messages required: while the periodic updates in the BF algorithm originate from the sink and must propagate across the whole network, the trajectory change of a node in the proposed algorithm originates from this node and affects only neighboring nodes; most of the kinetic spanning trees computed before the change are still valid after the change. As the update period of the BF algorithm is increased to 60 seconds, the message ratio drops to about two. However, the power ratio significantly increases (not shown).

VI. Conclusions and Future Work

While the amount of computation is increased at each node, the number of transmissions in one execution cycle of the proposed kinetic spanning tree algorithm is effectively the same as in the distributed Bellman-Ford algorithm. However, the proposed algorithm need not be updated continuously, so it substantially reduces the total number of routing messages; therefore, more bandwidth is left for data messages and the total power required to send the routing messages is reduced. Moreover, by using the sequence of minimum power spanning trees for data messages, the battery life is even further increased.

Future work is progressing in two directions. Firstly, we are studying the use of kinetic spanning tree algorithms in hierarchical networks. The problem of jointly clustering the network and determining the routes within and among clusters can be posed as an optimization problem. An approach using global competition finds an approximate solution. Secondly, we are investigating the performance degradation when multipath fading and other physical layer effects lead to imperfect knowledge of trajectory information.

References

[1] D. Bertsekas ad R. Gallager, *Data Networks*, Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[2] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Trans. on Prog. Lang. and Systems*, Vol. 5, No. 1, pp. 66-77, Jan. 1983.

[3] A. Fetterer and S. Shekhar, "A performance analysis of hierarchical shortest path algorithms," *Proc. IEEE Int. Conf. on Tools with Artificial Intelligence*, pp. 84-93, 1997.

[4] G. N. Frederickson, "Ambivalent data structures for dynamic 2-edge-connectivity and k smallest spanning trees," *SIAM J. Comput.*, Vol. 26, No. 2, pp. 484-538, April 1997.

[5] P. K Agarwal, D. Eppstein, L. J. Guibas, and M. R. Henzinger, "Parametric and kinetic minimum spanning trees," *Proc. 39th Annual Symp. on Foundations of Computer Science*, pp. 596-605

[6] L. J. Guibas, "Kinetic data structures: a state of the art report," *Proc. 3rd Workshop Algorithmic Foundations of Robotics*, 1998.

[7] G. J. Pottie, "Wireless sensor networks", *Proc. IEEE Information Theory Workshop*, Killarney, Ireland, pp. 139-140, June 1998.