This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2005 proceedings.

LPT for Data Aggregation in Wireless Sensor Networks

Marc Lee and Vincent W.S. Wong
Department of Electrical and Computer Engineering,
University of British Columbia, Canada
email: wnmlee, vincentw@ece.ubc.ca

***Abstract*— In wireless sensor networks (WSNs), when a stimulus or event is detected within a particular region, data reports from the neighboring sensor nodes (sources) are sent to the sink or destination. Data from these sources are usually aggregated along their way to the sink. The data aggregation via in-network processing reduces communication cost and improves energy efficiency. In this paper, we propose an overlay structure in which the sources within the event region form a tree to facilitate data aggregation. We call this tree a Lifetime-Preserving Tree (LPT). LPT aims to prolong the lifetime of the sources which are transmitting data reports periodically. In LPT, nodes which have higher residual energy are chosen as the aggregating parents. LPT also includes a self-healing feature by which the tree will be re-constructed again whenever a node is no longer functional or a broken link is detected. By choosing the Directed Diffusion [1] as the underlying routing platform, simulation results show that in a WSN with 250 sensor nodes, the lifetime of sources can be extended significantly when data are aggregated using the LPT algorithm.**

I. Introduction

The rapid advances in wireless communication and Micro Electro Mechanical System (MEMS) have made WSNs possible. Such environments are typically comprised of a large number of sensors being randomly deployed for detecting and monitoring tasks. These sensors, developed at a low cost and in small size (*mm*-scale for smart dust motes [2]), are responsible for sensing, data processing, and routing activities. Applications of such networks range from battlefield communication systems (e.g. intrusion detections and target surveillance) to environmental monitoring networks such as habitat monitoring, chemical sensing, infrastructure security, inventory and traffic control. For example, sensors may be distributed across a forest in order to report the origin of a fire event where there is a significant increase in the average monitoring temperature. Unlike conventional ad hoc communication networks, energy resources in WSNs are usually scarce due to the cost and size constraints of sensor nodes. Conserving energy is thus the key to the design of an efficient WSN.

Perhaps the most significant difference between an Internet-based system and a WSN is the collaborative efforts provided by sensors. Each node in an Internet-based system competes with other nodes for a fair share of resources in order to run tasks and applications of its own. Per-hop fairness is therefore the primary concern. WSNs, on the other hand, rely on the collective information provided by the sensors but not on any individual sensing report. Most sensor nodes are task-specific in that they are all programmed for one common application. A node at one specific time may be granted more resources than other nodes if the program objective is still satisfied. For this reason, network resources are shared but it is not necessary that they be equally distributed as long as the application performance is not degraded.

Since sensors are densely-deployed in WSNs, the detection of a particular *stimulus* or *event* can trigger the response from many nearby sensor nodes. Data in WSNs are usually not directly transmitted to the interested users upon event detection. Instead, nearby sensor nodes (*sources*) would aggregate the data locally to remove any redundancy. Reference [3] suggests that transmitting a data packet of size 1 *Kb* to a distance 100 meters away is similar to executing 3 million instructions on a general-purposed computer. Thus, it is beneficial to perform data aggregation via in-network processing to reduce the communication cost and improve the energy efficiency.

In this paper, we consider a network with M randomly-deployed sensors in which each node m has an identical transmission range and initial residual energy e_m. An event, triggering N sensors around it, occurs at a random place in the network. Data reports from these sources are clock-driven upon event detection. We define the *node lifetime* to be the time that a source node runs out of its energy. To this end, we propose an overlay structure in which the sources within the event region form a tree to facilitate data aggregation. We call this tree a *Lifetime-Preserving Tree* (LPT). LPT aims to prolong the lifetime of the sources which transmit data reports periodically. In LPT, nodes which have higher residual energy are chosen as the aggregating parents. LPT also includes a self-healing feature by which the tree will be re-constructed again whenever a node is no longer functional or a broken link is detected. By choosing the Directed Diffusion [1] as the underlying routing platform, simulation results show that in a WSN with 250 sensor nodes, the lifetime of sources can be extended significantly when data are aggregated using the LPT algorithm.

The remainder of this paper is organized as follows. The related work on data aggregation is summarized in Section II. The problem formulation is given in Section III. Our proposed LPT algorithms, including both centralized and distributed versions, are described in Section IV. The performance improvement of data aggregation is presented in Section V. Conclusions are given in Section VI.

II. Related Work

An energy-aware data aggregation tree (EADAT) algorithm is proposed in [4]. The base station (root) sends a broadcast control message periodically. Each node upon receiving this message for the first time starts a timer. The expiration time is inversely proportional to the node's residual energy. The timer is refreshed when a node receives this message during the timer count down.

A dynamic convoy tree-based collaboration (DCTC) framework for tracking a mobile target is proposed in [5]. Heuristics are used to predict the object's moving direction. A dynamic tree is then created by adding or pruning the sensors near the moving target. The root of the tree can dynamically refine the readings gathered from various tree nodes.

Since the coverage area of individual sensor nodes usually overlaps, the work in [6] attempts to periodically search the smallest subset of nodes that covers the monitoring area. This group of nodes is referred to as the area-dominating set. A distributed spanning tree, induced by the initial interest flood over the area-dominating set, is used to aggregate the reply messages from various event sources.

The issue of convergecast (many-to-one) for data aggregation is addressed in [7]. A tree that is rooted at the base station is constructed such that the link cost from each node to base station is minimized. Further improvement includes enhancing the chance of simultaneous aggregation and reducing the latency for convergecast. However, the algorithm is centralized and the knowledge of global connectivity is required.

Some recent work proposed the partition of network into small adjacent grids or clusters on which data aggregation is performed. Specifically, a cluster head is associated with each cluster so that all the nodes belonging to the same cluster can aggregate their readings through this node. The work in [8] periodically selects cluster heads according to a hybrid of the residual energy and node degree, resulting in a set of energy-rich cluster heads being uniformly distributed across the network. However, their work assumes that nodes have variable transmission power to maintain a certain degree of connectivity between the clusters.

An energy-aware spanning tree construction algorithm (E-Span) is proposed in [9][10]. E-Span is a distributed protocol. It creates a tree structure which includes all source nodes and contains no cycles. The node with the highest energy is selected as the root. Each other node selects the highest-energy neighbor for which the shortest-path configuration message comes from as its parent node.

III. Problem Formulation

Not all tree structures are ideal for aggregation inside the event area. Consider a multicast tree connecting 5 different sources (Fig 1 left). The fact that the lowest-energy node B has a dependent child of node C can indeed deplete its valuable energy quicker than if node C is attached to D instead (Fig 1 right). Node D will have a higher energy dissipation rate than what it had before. However, by balancing the lifetime of each individual node, the frequency of tree reconstruction, which repairs any broken tree link and incurs an additional energy cost to each source, can now be reduced. We thus conclude that by not considering the residual energy in tree construction can affect the collective functional lifetime of each source.

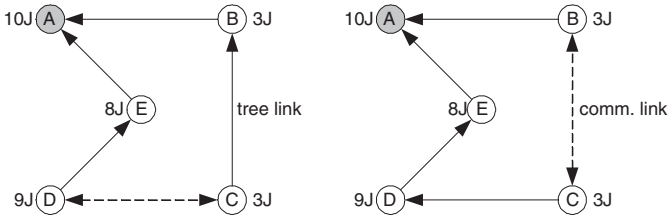

Fig. 1. An example to show that not considering the residual energy in the tree construction can affect the functional node lifetime.

In our scheme, we intend to extend the time to refresh a tree so that the cost of maintenance is reduced. We accomplish this by assigning nodes with higher energy to be the aggregating parents for the lower-energy nodes whenever possible. We define a *branch* to be the route from a root to a leaf node in a given tree. The terms *branch* and *tree energy* are introduced to reflect the time until the first link breaks in a given branch and tree, respectively. Let I_x denote the set of nodes along a given branch rooted at node x, and J_y be the set of nodes in a given tree rooted at node y. We have:

$$branch\ energy = min[e_i \mid i \in I_x,\ i \neq leaf\ node] \quad (1)$$
$$tree\ energy = min[e_j \mid j \in J_y,\ j \neq leaf\ node] \quad (2)$$

Our objective is to find a tree spanning all sources and an appropriate tree root for data collection so that the functional lifetime of each source node is prolonged as much as possible. Since the time till the first link breaks in a tree determines the lifetime of each source, and the term *tree energy* directly reflects this time, we tackle this problem by searching a tree that comprises the highest tree energy. Our LPT construction problem is thus formulated as follows:

Construct a tree rooted at node r such that
$$treeE_r \geq treeE_n \quad \forall\, n \in N,\ n \neq r \quad (3)$$
subject to the condition that
$$brE_{s,r,b} \geq brE_{s,r,p} \quad \forall\, s \in N,\ \forall\, p \in P_{s,r},\ p \neq b \quad (4)$$

where $P_{x,y}$ is the set of possible routes, with each labelling p, from nodes x to y, $treeE_z$ is the energy of a tree rooted at node z, and $brE_{f,g,h}$ is the energy of a branch h with node g as the root and node f as the leaf ($h \in P_{f,g}$).

IV. The LPT Construction Algorithm

In this section, we describe the centralized and distributed implementations of the LPT algorithms.

A. Centralized Approach

Recall that our scheme requires a root (initially unknown) to collect data from each other node via routes with the highest branch energy. Furthermore, the tree has energy that directly depends on the minimum energy of any non-leaf node on one

of these routes. If there exists a way to identify this minimum-energy node (which represents the bottleneck to the network), the highest tree energy can then be determined. The question lies on how to identify this node and coordinate the given set of network connection such that a tree is obtained with this node being configured as the minimum-energy non-leaf node.

To address this issue, we begin by arranging nodes in ascending energy levels. Starting from the node with the least energy, we test whether the removal of all the network links to this node except from its highest-energy neighbor will disconnect the existing graph. If so, the bottleneck node is found and there are no better ways than to collect data via this node. The removed links are therefore restored, and any tree rooted at one of the nodes in the remaining set shall have the energy as that of this chosen node. If not, the removed links do not contribute to the construction of the LPT and we shall continue with the next node with higher energy. Finally, when we come to the last node (i.e. the one with the highest energy), we conclude that there is no bottleneck node for this particular topology and any tree rooted at this last node, on the existing graph, can have the highest tree energy. The *centralizedLPT* function below summarizes our descriptions.

centralizedLPT (connectivity and energy matrices)
1 *sorts nodes in ascending energy level*
2 *for n = 1 to N, n++*
3 *get* $node_{n,max}$
4 *if* $node_{n,max}$ *exists,*
5 *remove* $link_{n,i}$ *and save i to I* $\forall\, i \in N, i \neq node_{n,max}$
6 *if the graph is not connected,*
7 *restore* $link_{n,j}$ $\forall\, j \in I$ *and empty I*
8 *set* $node_k$ *to be the root and run Dijkstra's algorithm on* $node_k$ *where k is any one number from n to N*
9 *set* $treeE_k$ *to be e(*$node_n$*) and return*
10 *set* $node_N$ *to be the root and run Dijkstra's alg. on* $node_N$
11 *compute* $treeE_N$ *by eqn (2) for the tree rooted at* $node_N$

where $node_x$ is the node with x^{th} least energy; $e(node_y)$ is the energy of $node_y$; $node_{z,max}$ is the highest-energy neighbor of $node_z$ subject to the condition that $e(node_{z,max}) \geq e(node_z)$; $link_{a,b}$ is the communication link, if it exists, between nodes a and b; I is a set, initially empty.

Consider Fig 2 as an example. Starting with the network shown on the left, several links attached to nodes 1, 7 and 4 are removed according to the procedures in the *centralizedLPT* function. Then, the graph on the right will become disconnected when the link between nodes 6 and 2 is removed. Thus, node 6 has to be a parent for some nodes in this network and is the bottleneck node. Any tree rooted at one of the nodes in the remaining set (i.e. nodes 2, 3, 5, 6, or 8) will have the tree energy of 7 J.

B. Distributed Approach

In this section, we describe a distributed approach to construct an LPT. The scheme examines the highest-energy branch from each source to a root node, by using a method similar

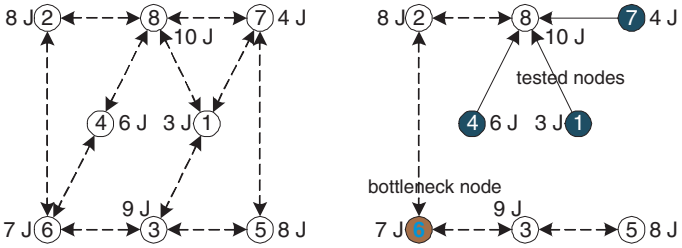

Fig. 2. An example of *centralizedLPT* function: Left - connectivity diagram. Right - bottleneck node found.

to Reverse-Path Forwarding (RPF) [11]. This approach will generate a total of N unique trees with each tree being rooted at a distinct source node. It then continues by comparing the energy of these trees and only uses the one with the highest tree energy for data aggregation.

1) Exploring the Highest-Energy Branch from every Source to any Root: In order to obtain the highest-energy branch between any pair of source and root such that equation (4) is satisfied, the protocol requires each source node *s* to initiate a message in a format that contains its energy information. When another source receives this message, it appends its energy information and broadcasts the updated message only if it has not received this message before or it has forwarded the message containing a lower branch energy. Otherwise, the message is being dropped. Eventually, various copies of the initiated message will traverse through various different routes p and only the ones with higher branch energy will arrive at the root r through route b.

We define eid_n to be the pair of energy and ID information of a node n and $brList_{i,j,k}$ to be a list containing the *eid*s for the message initiating node i up to the last receiving node j via route k with branch energy $brE_{i,j,k}$. Notice that $brE_{i,j,k}$ can be readily calculated by using equation (1) since all the required *eid*s are available. Therefore, $brList_{i,j,k}$ shall have the format of a list as follows:

$$brList_{i,j,k}\text{: } eid_i \rightarrow eid_x \rightarrow \ldots \ldots \ldots \rightarrow eid_y \rightarrow eid_j \qquad (5)$$

where node x and node y are some intermediate receiving nodes. Note that when node j receives $brList_{i,y,p}$ from node y via a route p, it is as if node j is a root and node i is a leaf for the branch sitting between nodes j and i. Our descriptions can hence be summarized in the following function:

exploreBranch (node ID n, node energy e_n*)*
1 *create and single-hop broadcast* $brList_{n,n,-}$
2 *while receiving* $brList_{i,j,k}$ *from j* $(k \in P_{i,j}, i \in N, i \neq n)$*,*
3 *if n has not seen the message initiating node i,*
4 *append* eid_n *to head of* $brList_{i,j,k}$ *and update* $brE_{i,n,p}$
5 *store and single-hop broadcast* $brList_{i,n,p}$ $(p \in P_{i,n})$
6 *else if min*$\{e_n, brE_{i,j,k}\} >$ *the stored* $brE_{i,n,q}$
7 *remove* $brList_{i,n,q}$ *and* $brE_{i,n,q}$ $(q \in P_{i,n})$
8 *append* eid_n *to head of* $brList_{i,j,k}$ *and update* $brE_{i,n,p}$
9 *store and single-hop broadcast* $brList_{i,n,p}$ $(p \in P_{i,n})$

2) Constructing a Tree Spanning all Event Nodes for every Source: We now proceed to construct N trees with each

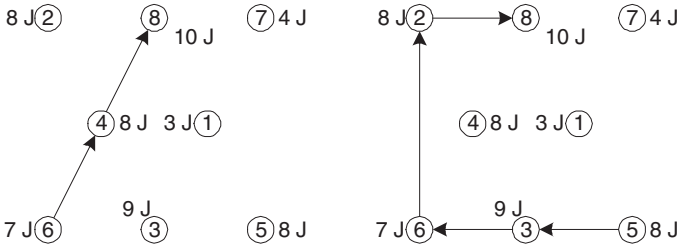

Fig. 3. An example of when a loop can be created: Left - stored branch for initiator node 6. Right - received branch for initiator node 5.

tree rooted at a distant source by incrementally attaching any branch explored in the last section. Each source has an initial tree structure that only comprises of itself. In order to construct a tree for each source that spans all other event nodes, each source node has to incrementally update its existing tree structure upon receiving any message with an unknown initiating node, or equivalently any new branch. Notice that the energy of the received branch directly determines the energy of the updated tree. To ensure that each tree carries the highest energy, the tree is also updated whenever the receiving node identifies a message with higher branch energy.

We define *initiator* to be the source that initiates the message and $tree_n$ to be a table containing the tree structure created by node *n*. In other words, when a message arrives, the information of *initiator*, $brList$, and brE in $tree_n$ are stored in $tree_n$. Notice that $treeE_n$ of $tree_n$ can be calculated as the minimum among all the stored branch energy.

Besides attaching new branches, each source is also responsible for preserving the loop-free property of the tree during the updates. In fact, some precautions are required in order to reject a branch that actually violates this property. To show when this can happen, consider Fig 3. Assuming that the branch for the initiating node 6 shown on the left has already been stored in $tree_8$, a loop will be created as soon as the newly-arrived branch shown on the right is being added to $tree_8$. In order to avoid creating loops, each node always has to reject a newly-arrived branch when the already-stored branch for each parent on this newly-arrived branch does not match the route where the message travels.

3) Searching a Lifetime-Preserving Tree for every Source: Since each source carries its unique tree structure, the protocol requires every source to broadcast its tree structure and select the one with the highest tree energy for aggregation among these nodes such that equation (3) can be satisfied.

We define lpt_n to be the LPT that a node *n* currently selects and $lptE_n$ to be the tree energy of lpt_n. Note that lpt_n initially equals to $tree_n$. By single-hop broadcasting lpt_n for all sources $n \in N$ and have each source node *n* to re-broadcast the received lpt_i if $lpt_i > lpt_n$, a tree with the highest tree energy will eventually be selected by all sources.

Our descriptions can thus be summarized in the *searchLPT* function. Notice that the *betterTree* function compares the lpt_i with lpt_n and returns *true* if $lptE_n \geq lptE_i$. Tree depth, root energy, and root ID are used to break ties whenever necessary.

searchLPT (node ID n)
1 single-hop broadcast lpt_n
2 while receiving lpt_i *from node i* ($i \neq n$),
3 if betterTree(lpt_i),
4 delete lpt_n *and copy* lpt_i *to* lpt_n
5 calculate $lptE_n$ *using (1)*
6 single-hop broadcast lpt_n

C. Discussion

The LPT shares the same objective as [4]-[7] in an attempt to construct a data aggregation tree and select a dedicated root for which data are gathered. Both LPT and EDAT [4] select the aggregating nodes which have higher energy. In addition, LPT, EDAT [4], and HEED [8] consider the residual energy, thereby enhancing the likelihood of distributing the loads over higher-energy nodes.

However, LPT differs from EDAT [4] in that extensive use of timers is not necessary. While the work in [4][6][7] requires the prior knowledge or support from a given tree root, our proposed LPT does not require the root to be any particular node. In terms of root selection, we consider the residual energy of nodes within the event area whereas [6][7] compare the link cost that associates with each of them.

V. Performance Evaluation

In this section, we first present the performance comparisons between the centralized and distributed LPT algorithms. We then compare the distributed LPT algorithm with the Directed Diffusion [1] and E-Span [9][10].

A. Simulation Model

We implemented our tree construction modules on top of Directed Diffusion [1] in the *ns*-2 network simulator [12]. A square field with each side measuring *L* meters is considered. A number of *M* identical nodes, ranging from 50 to 250 in increment of 50, are randomly deployed in this field such that the average node density is kept at $\lambda = 50/160^2 \ \ nodes/m^2$. Also, 5 sinks and N sources are randomly chosen among the nodes, subject to the conditions that $N = 0.1\,M$ and sources be interconnected to each other (to model a single stimulus). The IEEE 802.11 MAC is used as the link layer. The radio range is 40 meters. To model an event-driven data network, each source generates random data reports of size fixed at 136 bytes in constant intervals 1 of pkt/sec. An application that computes the average of reports generated by various sources is used to model the aggregation behaviors.

To reasonably limit the simulation time, we altered the *ns*-2 energy model such that the sources carry an initial energy that is randomly chosen between 10 and 15 *J*. All other nodes (i.e., pure relaying nodes) are given with much higher initial energy so that the performance of the lifetime of the sources can be studied. Idle time power, receive power, and transmit power dissipation are set at 35, 395, and 660 *mW*, respectively. We assume that the data processing and aggregation cost are negligible.

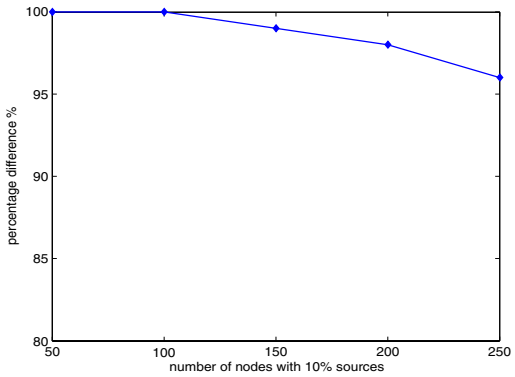

Fig. 4. Percentage difference on tree energy between distributed and centralized LPTs.

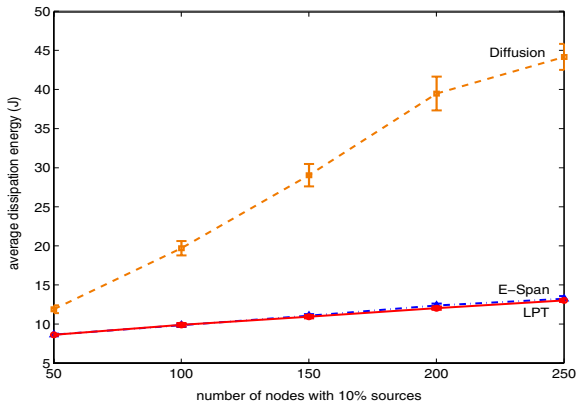

Fig. 5. Average dissipation energy as a function of network size.

B. Tree Energy: Distributed LPT vs. Centralized LPT

Fig 4 shows the percentage difference between the tree energy obtained by distributed and centralized LPT algorithms. The x-axis is the total number of sensor nodes M. Since we assume that the number of sources N is always equal to $0.1M$, an increase M also increases N. Results show that the difference is less than 5%. There is a slight increase in the percentage difference when N increases. This is due to the fact that the distributed LPT algorithm sends broadcast messages during the tree construction phase. When packet loss occurs occasionally, the distributed tree may not be the same as the centralized one.

C. Dissipated Energy

To study the impact of aggregation on energy reduction, we measure the *average dissipated energy*, which is defined as the average amount of energy consumed by a relaying sensor node throughout the entire simulation. The results are averaged over 20 different runs with a 95% confidence interval. Results in Fig 5 show a considerable amount of energy reduction by using either LPT or E-Span when compared with the Directed Diffusion. Such energy reduction is expected since both LPT and E-Span efficiently aggregate the data by combining several readings from various sources into a single summarized message.

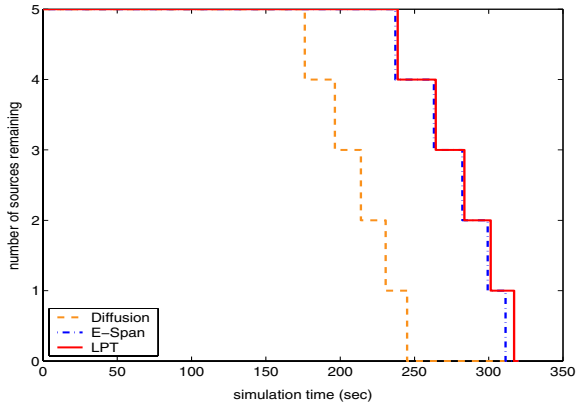

Fig. 6. Average node lifetime for each source with $M = 50$ nodes.

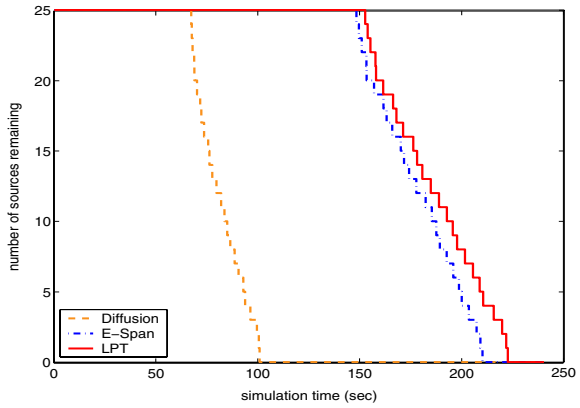

Fig. 7. Average node lifetime for each source with $M = 250$ nodes.

D. Node Lifetime

The *average node lifetime* measures the time at which a source runs out of its available energy. Figs 6 and 7 show the results when $M = 50$ and 250 nodes, respectively (similar results for other network sizes can be found in [9]). We make the following observations:

1) Both LPT and E-Span considerably extend the node lifetime of each source, especially when the network is large.
2) Both LPT and E-Span have similar performance when M is small.

The impact of data aggregation is again validated in 1). By combining reports from various sources, both LPT and E-Span suppress a considerable amount of traffic in the network. Since less energy is being consumed, there is a noticeable lifetime improvement when data are collected via the trees. For 2), we argue that the chance of obtaining an identical tree by using LPT and E-Span, respectively, is relatively high when there are fewer sources. In fact, when all the nodes are within the radio range of each other, both LPT and E-Span will create an identical tree with the highest-energy node selected as the root and all other nodes as leaf nodes. When this happens, the lifetime improvement will be similar. LPT has a better performance than E-Span when there are more sources within the event region.

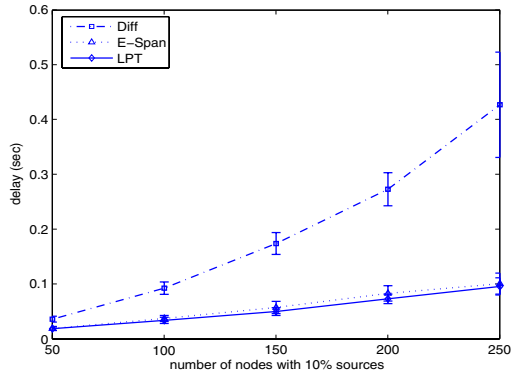

Fig. 8. Average data packet delay between a source and sink.

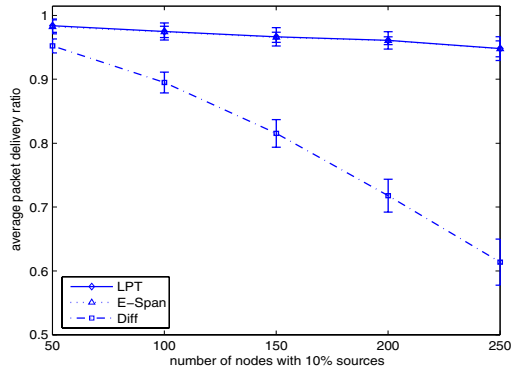

Fig. 9. Average packet delivery ratio between transmitting a data packet from each source and receiving it at each sink.

Note that we choose a low initial residual energy and a small *simulation time* (less than 400 secs) in order to reduce the *actual time* to run all the simulations. Results reported in this section can be extrapolated to the scenarios where the sensor nodes initial residual energy are much higher.

E. Average Data Packet Transfer Delay

The next experiment compares the average data packet delay between each source and sink for Directed Diffusion (Diff), LPT, and E-Span. Results in Fig 8 show that the Directed Diffusion incurs a higher delay than both LPT and E-Span. Since LPT and E-Span combine data from various sources, it is as if only a single source is generating packets. This is also true in a network with a large number of data sources.

F. Average Data Packet Delivery Ratio

The last experiment measures the average packet delivery ratio for Directed Diffusion (Diff), E-Span, and LPT. Fig 9 shows that the Directed Diffusion experiences severe congestion when the number of sources N increases. However, both LPT and E-Span are able to maintain their packet delivery ratios even when N increases. The Directed Diffusion has its network overloaded with data traffic when more sources are sending packets. A considerable amount of data packets are therefore being dropped. LPT and E-Span, on the other hand, inject data to the sensor network as if there is only a single source. Thus, they are able to steadily maintain the packet delivery ratio even when the network is large.

VI. Conclusions

In this paper, we proposed an overlay structure in which the sources within the event region form a tree to facilitate data aggregation. We call this the Lifetime-Preserving Tree (LPT). Both the centralized and distributed LPT algorithms are described. Simulation results show that the trees created by both centralized and distributed LPT algorithms have similar tree energy. The average node lifetime for sensors using LPT are higher than that of the Directed Diffusion and E-Span. LPT also maintains a low average packet transfer delay and a high packet delivery ratio. Further work includes performance comparison between LPT with other data aggregation algorithms.

Acknowledgment

This work is supported by the Natural Sciences and Engineering Research Council of Canada under grant number 261604-03.

References

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. of ACM MobiCom'00*, Boston, MA, Aug. 2000, pp. 56–67.

[2] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile networking for smart dust," in *Proc. of ACM MobiCom'99*, Seattle, WA, Aug. 1999, pp. 271–278.

[3] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, May 2000.

[4] M. Ding, X. Cheng, and G. Xue, "Aggregation tree construction in sensor networks," in *Proc. of IEEE VTC'03*, Orlando, FL, Oct. 2003, pp. 2168–2172.

[5] W. Zhang and G. Cao, "DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1689–1701, Sept. 2004.

[6] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *IEEE Computer Magazine*, vol. 37, no. 2, pp. 40–46, Feb. 2004.

[7] S. Upadhyayula, V. Annamalai, and S. K. S. Gupta, "A low-latency and energy-efficient algorithm for convergecast," in *Proc. of IEEE GLOBECOM'03*, Dec. 2003, pp. 3525–3530.

[8] O. Younis and S. Fahmy, "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, Oct. 2004.

[9] M. Lee, "The construction of a lifetime-preserving tree for data aggregation in wireless sensor networks," M.A.Sc. Thesis, The University of British Columbia, Vancouver, Canada, Nov. 2004.

[10] M. Lee and V. Wong, "An energy-efficient spanning tree algorithm for data aggregation in wireless sensor networks," in *Proc. of IEEE PacRim'05*, Victoria, BC, Aug. 2005.

[11] Y. K. Dalal and R. M. Metcalfe, "Reverse-path forwarding of broadcast packets," *Communications of the ACM*, vol. 21, no. 12, pp. 1040–1048, Dec. 1978.

[12] VINT. (2001, Nov.) The network simulator ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns