Journal of Southeast University (English Edition) Vol. 24, No. 4, pp. 414 – 419 Dec. 2008 ISSN 1003—7985

(α, β)-constraints connected dominating set algorithm in wireless sensor network

Sun Yanjing[1] Qian Jiansheng[1] Gu Xiangping[1] Chen Guangzhu[2]

([1] School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou 221008, China)

([2] School of Mechanical and Electrical Engineering, China University of Mining and Technology, Xuzhou 221008, China)

Abstract: To cope with the constraint problem of power consumption and transmission delay in the virtual backbone of wireless sensor network, a distributed connected dominating set (CDS) algorithm with (α, β)-constraints is proposed. Based on the (α, β)-tree concept, a new connected dominating tree with bounded transmission delay problem (CDTT) is defined and a corresponding algorithm is designed to construct a CDT-tree which can trade off limited total power and bounded transmission delay from source to destination nodes. The CDT algorithm consists of two phases: The first phase constructs a maximum independent set (MIS) in a unit disk graph model. The second phase estimates the distance and calculates the transmission power to construct a spanning tree in an undirected graph with different weights for MST and SPT, respectively. The theoretical analysis and simulation results show that the CDT algorithm gives a correct solution to the CDTT problem and forms a virtual backbone with (α, β)-constraints balancing the requirements of power consumption and transmission delay.

Key words: wireless sensor network; connected dominating set; transmission delay; maximal independent set; power consumption

A wireless sensor network inherits the characteristics of ad hoc wireless networking. Although a wireless ad hoc sensor network has no physical backbone infrastructure, a virtual backbone can be formed by nodes in a connected dominating set (CDS) of a corresponding unit-disk graph (UDG)[1].

The connected dominating set of a graph representing a network has a significant impact on the efficient design of topology control and routing protocols in wireless sensor networks. Recent studies[2-6] in this area have focused on finding a minimal CDS (MCDS) for higher efficiency and trying to maintain a certain degree of redundancy in the virtual backbone for fault tolerance and routing flexibility. In fact, as energy conservation is an important concern, transmission is also very significant for crucial information transport in wireless sensor networks. In the survey[7], many proposed algorithms for constructing a spanning tree to guarantee connectivity and coverage do not pay enough attention to energy and transmission delay simultaneously. Sensor nodes in a sensor network have very limited power, so power must be used efficiently. The total power consumption of all the nodes needs to be bounded for energy conservation[8]. Transmission delay refers to the time for messages to travel across the network. It is highly dependent on the length of the path between two communicating nodes. Therefore, to obtain a small transmission delay between a pair of nodes, it is expected that the path from the source to the destination should be as short as possible. However, little research work considers bounding the total energy consumption and transmission delays simultaneously for the construction of CDS. Ref. [9] introduced a (α, β)-tree into a broadcast tree to trade off energy and transmission delay. We construct a connected dominating tree with these two concerns and try to balance them so that the tree can serve the network better. We give a formal definition of this kind of problem as (α, β)-constraints. Then we extend it into the connected dominating tree (CDT) with the bounded transmission delay (CDTT) problem and give out the solution by constructing a CDT limited total power while the transmission delays between the source and all the other nodes are also bounded. With such a tree, the requirements for energy conservation and transmission delay are both satisfied. We also propose a distributed CDT algorithm (dCDT) which consists of two phases. In realistic application, we may integrate related factors into α or β respectively to tune the constraint relationships of power and delay for a virtual backbone in wireless sensor networks.

1 CDT Problem Formulation

1.1 (α, β)-constraint problem

For CDS, to the best of our knowledge, no other research work considers bounding the transmission delay and limiting the total energy consumption simultaneously. We propose a novel method to solve this problem with the help of an (α, β)-tree which has been introduced into wired networks[10]. As known, it is almost impossible to consider minimizing the total weight and minimizing the distance from the root to each node simultaneously using just a minimum spanning tree (MST) or just a shortest path tree (SPT), since there is a trade-off between these two minimization goals. In Ref. [10], the LAST algorithm is proposed to construct a spanning tree that can simultaneously approximate a shortest path tree and a minimum spanning tree. The motivation is to balance the minimization purposes of the total link cost and the cost of a message to be sent from each host to the root of the tree. In wired networks, energy conservation is not a great concern. However, in wireless sensor networks, energy is a cherished resource that all the algorithms and protocols must take into account. Therefore, the cost in LAST is defined based on the distance of each neighboring host, while the cost in our algorithm should be the energy spent to transmit

Received 2008-01-21.

Biographies: Sun Yanjing (1977—), male, doctor; Qian Jiansheng (corresponding author), male, doctor, professor, qianjsh@cumt.edu.cn.

Foundation items: Major Program of the National Natural Science Foundation of China (No. 70533050), High Technology Research Program of Jiangsu Province (No. BG2007012), China Postdoctoral Science Foundation (No. 20070411065), Science Foundation of China University of Mining and Technology (No. OC080303).

Citation: Sun Yanjing, Qian Jiansheng, Gu Xiangping, et al. (α, β)-constraints connected dominating set algorithm in wireless sensor network[J]. Journal of Southeast University (English Edition), 2008, 24(4): 414 – 419.

and the path length from the sender to the receiver. Therefore, we have a double weights spanning tree for the CDS.

Thus, α directly correlates with the requirement of total path length which corresponds to transmission delay and β represents the requirement of power or energy consumption. Our purpose is to moderate two objectives such as α and β for a spanning tree of CDS, so-called (α, β)-constraints.

1.2 CDT problem formulation

We model a virtual backbone in a wireless sensor network as a general undirected and weighted graph $G = (V, E)$. We hope to find a tree C spanning G with (α, β)-constraints. The following is a list of notations we use:

$d(u, v)$ or $d(e)$ is the distance or length of an edge $E(u, v)$ from u to v according to some norm (we use the Euclidean norm).

P_{uv} is the power needed to transmit from u to v, which is equal to the weight from u to v, $w(e)$ or $w(u, v)$,

$$p_{uv} = w(e) = w(u, v) = C_v d_{uv}^{\gamma} \tag{1}$$

C_v is the constant determined by the signal detection threshold which is related to environments. To transmit a signal over a distance d, the required radiation energy is proportional to d^{γ} [9], where typically γ is 2 and can range up to 6 in environments with multiple-path interferences or local noise.

$p(u)$ is the maximum transmission power for node u in the spanning tree for a CDS, $p(u) = \max\{p_{uv}: (u, v) \in G\}$.

$w(T)$ is the total weight of a spanning tree T,

$$w(T) = \sum_{e \in T} w(e) \tag{2}$$

$p(T)$ is the total power consumption of a spanning tree T,

$$p(T) = \sum_{u \in V} p(u) \tag{3}$$

$d(G)$ is the total length of a graph G,

$$d(G) = \sum_{e \in E} d(e) \tag{4}$$

We now formally define the connected dominating tree with a bounded transmission delay problem as follows:

Definition 1 Connected dominating tree with bounded transmission delay (CDTT) problem in wireless sensor network. Given a subset of nodes in a wireless sensor network, construct a connected dominating tree (CDT) so that d(CDT) is bounded by a user-specified threshold and p(CDT) is limited.

In this paper, we introduce (α, β)-tree into wireless sensor networks to solve the CDTT problem in accord with (α, β)-constraints. It approximately satisfies the power constraint and the transmission delay constraint by constructing a connected CDT. Based on the definition of the (α, β)-tree in wired networks[10], we give the definition of a CDT tree in wireless sensor networks as follows:

Definition 2 For $\alpha \geqslant 1$ and $\beta \geqslant 1$, a spanning tree C of G meeting the following two requirements is called an (α, β)-tree rooted at r.

1) Distance: For every vertex u, the distance between r and u in C is at most α times the shortest distance from r to u in G.

2) Power: The power of C is at most β times the power of an optimal solution to the CDTT problem.

The weight or cost is defined as the energy needed to transmit between a pair of nodes in wireless sensor networks for MST of T and as the distance between a pair of nodes for SPT of T.

2 CDT Algorithm

We present a CDT algorithm for constructing a CDT tree C. Given a weighted and connected graph $G = (V, E)$, a root (sink) node r and an $\alpha > 1$, the CDT construction algorithm constructs a CDT tree satisfying the constraints that the total power of the CDT and path lengths between the root r and the other nodes are bounded. The transmission power between pairs of nodes is used as weight to calculate the minimum spanning tree, and the path length between pairs of nodes is used as weight to calculate the shortest path tree for the CDT tree. The algorithm is given an $\alpha > 1$, a minimum spanning tree, and a shortest path tree rooted at a vertex r. It returns a CDT tree rooted at r.

The basic idea of the algorithm is to traverse the minimum spanning tree, maintaining a current tree, and checking each vertex when it is encountered to ensure that the distance requirement for that vertex is met in the current tree. If it is not met, the edges of the shortest path between the vertex and the root are added into the current tree. Other edges are discarded so that a tree structure is maintained.

After all vertices have been checked and all paths added as necessary, the remaining tree is the desired CDT tree. The final tree is not too heavy because a shortest path is only added if the path that it replaces is heavier by a factor of $\alpha > 1$. This allows a charging argument bounding the net weight of the added paths.

The general procedure of the CDT algorithm is as follows: First, it constructs an MST T. T is rooted at r where there exists a path from r to every other node. Then, based on the connectivity information of the network, the CDT algorithm constructs an SPT H. H includes all the shortest paths from r to every other node. Next, T will be traversed in a depth-first manner. When visiting a node u, if the length of the path from r to u in T is larger than a user-defined threshold α, then this path is replaced by the path from r to u in H. After the CDT algorithm ends, the CDTT problem is solved.

We use Dijkstra's algorithm to construct an SPT H. H includes all the shortest paths from the root r to all the other nodes. To construct and maintain the CDT C, each node u needs to have a parent pointer $p[u]$ and an upper bound $d[u]$ on the distance from node u to the root r. Let $D_G(u, v)$ denote the length of the path from u to v. We use the INITIALIZE and RELAX algorithms in Ref. [10] to initialize and maintain both of these attributes.

The CDT algorithm then traverses the MST T in a depth-first manner beginning from the root r along the paths from r to all the other nodes. This is possible since, when constructing C, each node u's parent and children have already been recorded. When node u is reached for the first time, if $d[u]$ is greater than $\alpha D_{\mathrm{SPT}}(r, u)$, then the shortest P_{ru} in H is

added to C and $d[u]$ and $p[u]$ are updated. After this, node u's parent v needs to be checked if the updated path from r to u will result in shortening the path from r to v. If so, then v's parent will be checked and so on until the root r is reached. The CDT algorithm is given in the following:

Algorithm CDT

```
INITIALIZE(G, r) {
    for each vertex v ∈ V do
    d[v] ←∞
    p[u] ← nil
    end for
d[r] ← 0}
RELAX(u, v) {
    if d[v] > d[u] + w_SPT(u, v) then
    d[v] = d[u] + w_SPT(u, v)
    p[v] ←u
    end if}
ADD-PATH(v) {
    if d[v] > D_SPT(r, v) and parent_SPT(v) ! = nil then
    ADD-PATH(parent_SPT(v))
    RELAX(parent_SPT(v), v)
    end if}
DFS(u) {
        if d[u] > αD_SPT(r, u) then
        ADD-PATH(u)
        end if
        for each child v of u in MST do
        RELAX(u, v)
        DFS(v)
        RELAX(v, u)
        end for}
CDT(MST, SPT, r, α) {
        INITIALIZE(MST, r)
        DFS(r)
        return CDT C}
```

The CDT C is the desired tree satisfying the constraints that both $p(C)$ and $d(C)$ are bounded.

To better understand the CDT algorithm, a sample execution is illustrated in Fig. 1. Fig. 1(a) is the undirected graph. Fig. 1(b) is the connectivity information of a network where the number on each edge denotes the physical distance between a pair of nodes. Fig. 1(c) is the weighted graph after considering the transmission power of each node. The shortest path tree H is given in Fig. 1(d) and the MST T is illustrated in Fig. 1(e). Suppose that 1 is the root and $\alpha = 2$. T is first traversed. When visiting 4, the length of the path $(1, 2, 3, 4)$ in T is $d_T(1, 4) = 23$, and the length of the path $(1, 4)$ in SPT H is $d_H(1, 4) = 11$, therefore, $d_T(1, 4) = 23 > \alpha d_H(1, 4) = 11$. Thus, edge$(1, 4)$ is added and $P[u]$ is changed from 3 to 1. Note that edge $(3, 4)$ is not deleted (represented as dotted line) since we need to go back along edge$(3, 4)$ to retrieve 4's original parent 3 to check if the new added path (edge$(1, 4)$) also shortens the path from 1 to 3. In this example, the path from 1 to 3 will not be replaced. This backward check of a node's original parent happens whenever the path to this node is replaced. Finally, a CDT tree C is obtained after traversing T, as shown in Fig. 1(f). The number in the brackets represents the physical distance of a pair of nodes; otherwise, the numbers represent the weight of this edge.

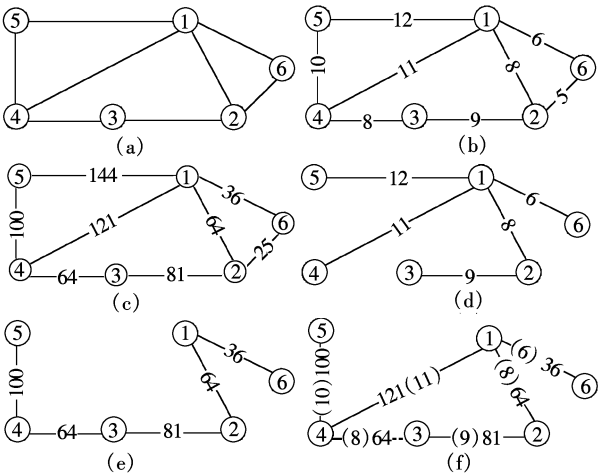

Fig. 1 A sample execution of the CDT algorithm

Next, let us check the trade-off ability of the CDT algorithm. We calculate the total power and the total length for MST, CDT and SPT, respectively, as shown in Fig. 1.

$$p(\mathrm{MST}) = 345, \quad p(\mathrm{CDT}) = 402, \quad p(\mathrm{SPT}) = 446$$
$$d(\mathrm{MST}) = 91, \quad d(\mathrm{CDT}) = 63, \quad d(\mathrm{SPT}) = 54$$

Obviously, we can conclude the above as

$$p(\mathrm{MST}) < p(\mathrm{CDT}) < p(\mathrm{SPT})$$
$$d(\mathrm{MST}) > d(\mathrm{CDT}) > d(\mathrm{SPT})$$

It shows that the CDT algorithm does well and complies with our expectations.

3 Distributed CDT Algorithm

Our distributed algorithm for construction of CDS with (α, β)-constraints consists of two phases. These two phases construct a maximal independent set (MIS) based on a unit disk graph, and a connected dominating tree based on a general undirected graph[11], respectively.

3.1 Distributed construction of MIS

By definition, any pair of nodes in an MIS are separated by at least two hops. However, a subset of nodes in an MIS may be three hops away from the subset of the remaining nodes in this MIS. The MIS constructed in this section guarantees that the distance between any pair of its complementary subsets is exactly two hops. Our construction uses a carefully chosen rank definition. The ranking is induced by an arbitrary rooted spanning tree T, which can be constructed by the distributed leader-election algorithm in Ref. [12] with $O(n)$ time complexity and $O(n\log n)$ message complexity. Given a rooted spanning tree T, the (tree) level of a node is the number of hops in T between itself and the root of T. (Thus the level of the root is 0.) The rank of a node is then given by the ordered pair of its level and its ID. Such ranking gives rise to a total ordering of the nodes in a lexicographic order. The following distributed process[1] enables each node to calculate its own rank and the number of lower-ranked neighbors.

Each node maintains two local metering variables, *neib* and *child*. The variable *neib* counts the number of neighbors whose levels have not yet been identified and is thus initialized to the number of neighbors. The variable *child* counts the number of children who have not yet reported the completion and is thus initialized to the number of children. Each node also maintains a *levellist* that records the levels of its neighbors and is initially empty, and a local variable *lneib* stores the number of lower-ranked neighbors. After the rooted spanning tree T is constructed, the root announces its level 0 by broadcasting a LEVEL message.

Upon receiving a LEVEL message, a node appends an entry consisting of the sender's ID and level to *levellist* and then decreases *neib* by 1. If the sender is its parent in T, it sets its own level to one plus the sender's level, and then announces this level by broadcasting a LEVEL message. If $neib = 0$, it sets *lneib* to the number of lower-ranked neighbors which can be calculated from *levellist*. If it is a leaf in T (i. e. $child = 0$ initially) and its own level has been determined, it transmits an L_Complete message to its parent. Upon receiving an L_Complete message towards itself, a node decreases *child* by 1; if $child = 0$ after the update and it is not the root, a node transmits an L_Complete message to its parent and then resets *child* to the number of children. When the local variable $child = 0$ at the root, the root simply resets *child* to the number of children. By this time, all nodes know their own ranks and all its neighbors and thus the root moves onto the construction of the MIS by a color-marking process.

All nodes are marked with white color initially and are marked with either gray or black eventually. Each node also maintains a *blacklist* which records the IDs of its black neighbors. Note that the *blacklist* can contain at most five black nodes. The root first marks itself black and broadcasts a BLACK message. Upon receiving a BLACK message, a node adds the sender's ID to the *blacklist*, and if its color is still white, it marks itself gray and broadcasts a GRAY message which contains its level. Upon receiving a GRAY message, if the rank of the sender is lower than its own, a white node decreases *lneib* by 1; if $lneib = 0$ after the update, it marks itself black and broadcasts a BLACK message. When a leaf node is marked with either gray or black, it transmits an M_Complete message to its parent. Upon receiving an M_Complete message towards itself, a node decreases *child* by 1; if $child = 0$ after the update and it is not the root, a node transmits an M_Complete message to its parent. By the time the local variable $child = 0$ at the root, all nodes will have been marked with either gray or black and thus the root will move onto the construction of the CDS.

3.2 Distributed CDT algorithm

We introduce the distributed CDT algorithm which is executed at every node in a general undirected graph. Three existing distributed algorithms dMST[13], dSPT[14], and dDFS[15] are executed to obtain an SPT, an MST, and a DFS traversal order. All of these algorithms are initiated and terminated by a Manager. The Manager can be the root node. Then, a CDS with(α, β)-constraints is constructed in a distributed manner. During the construction of a CDS, we include message ADD_FINISH and a field initiator for some messages for the purpose of synchronization. Nodes communicate with each other through exchanging the following messages[8]:

WAKEUP: If node u sends this message to node v, this indicates that u has been visited in the DFS order and v becomes the current being visited node.

ADD: If node u sends this message to node v, this indicates that the path from r to u in the SPT will be added and u is telling its parent v in the SPT this fact. This message should record u as the initiator of this ADD-PATH operation.

ADD_FINISH: This message is initiated by the root r after it receives an ADD-PATH request and this message is transferred by all the nodes on the being added path. Note that since each ADD message has an initiator, the ADD_FINISH message also needs to have an initiator which is the same as the one in the corresponding ADD message.

UPDATE: If node u sends this message to node v, this indicates that edge(u, v) needs to be RELAXed.

To construct a CDS, the dSPT[14], dMST[13], and dDFS[15] are executed at each node first. Since the initiation and termination of each distributed algorithm is conducted by the root r, all the nodes in the network can be synchronized and these algorithms can be employed in a serialized distributed fashion. A DFS traverse is first conducted on the MST T according to the order obtained in the dDFS procedure. Each node u uses $d[u]$ to record the current distance between the root r and itself and uses $P[u]$ to record its parent in the desired CDS. Initially, $d[u]$ is set to $d_T(r, u)$ for each node u. The root r initiates this traverse by sending out a WAKE-UP message to its successor in the order obtained from the dDFS.

The algorithm terminates after the root r receives an UPDATE message with initiator u, which is marked as the last node in the order obtained from the dDFS.

4 Analysis and Simulation

4.1 Analysis

First, we evaluate the correctness of the CDT algorithm by examining if the two constraints in definition 2 are satisfied. α is a user-defined input and β is related to α. We derive a relationship between α and β so that the relationship between the powers of the constructed CDT and the optimal solution to the CDTT problem can be obtained.

We employ the CDT algorithm in a graph $G = (V, E)$ to construct a CDT. For Eq. (1), we set γ to 2 and C_v to 1 for every node v. We define $p(\text{opt})$ to be the total power consumption of an optimal solution opt to the CDTT problem in G. Let T_{opt} be the tree rooted at root r of opt.

Lemma 1 The distance between r and v in H is at most α times the shortest path distance from r to v in G.

Proof During the construction of T, for each node v, if $d[v]$ exceeds α times the distance in the SPT H, then the shortest path between r and v is added to replace the old one by calling ADD-PATH. In any case, after v is visited, $d[v]$ is at most α times the shortest path distance and subsequently never increases. This added shortest path will never be replaced by other paths. On termination it bounds the distance in H.

Lemma 2 Given an SPT H and an MST T, the transmission power of H is at most $\left(\frac{2}{\alpha-1}\right)^2$ times the MST weight.

Proof As proven in Ref. [10], $d(H) \leqslant \left(\frac{2}{\alpha-1}\right) d(T)$

$$p(H) = w(H) = d^2(H) \leqslant \left(\frac{2}{\alpha-1}\right)^2 d^2(T) = \left(\frac{2}{\alpha-1}\right)^2 w(T) \quad (5)$$

Lemma 3 $w(T_{\text{opt}}) \leqslant \Delta \cdot p(\text{opt})$, where Δ is the maximum node degree in G.

Proof For each node u on the tree rooted at the root, $\sum_{(u,v)\in E} w(u,v) \leqslant \Delta \cdot p_u$, thus $w(T_{\text{opt}}) \leqslant \Delta \cdot p(\text{opt})$.

Theorem 1 Let C be the CDT tree constructed by the CDT algorithm, then $p(C) \leqslant \left(1+\left(\frac{2}{\alpha-1}\right)^2\right)\Delta \cdot p(\text{opt})$.

Proof Let H be an SPT and T be an MST rooted at the root. From lemma 2, lemma 3 and the definition of CDT, we have

$$p(C) \leqslant w(C) \leqslant w(T) + w(H) \leqslant \left(1+\left(\frac{2}{\alpha-1}\right)^2\right) w(T) \leqslant \left(1+\left(\frac{2}{\alpha-1}\right)^2\right) w(T_{\text{opt}}) \leqslant \left(1+\left(\frac{2}{\alpha-1}\right)^2\right)\Delta \cdot p(\text{opt}) \quad (6)$$

In the following, we analyze the message complexity and time complexity of the distributed algorithm.

Theorem 2 The time complexity of the dCDT algorithm is $O(n^2)$ and the message complexity of the dCDT algorithm is $O(n^2)$.

Proof The dCDT is executed at each node simultaneously. The execution time is thus dominated by the ADD-PATH procedure. The ADD-PATH function executes RELAX at most once and the execution time for the function RELAX is $O(1)$. Therefore, the execution time of the function ADD-PATH is proportional to the number of relaxed edges. The worst case is that the longest path of length $O(n)$ in a network needs to be added. Thus, the execution time of the function ADD-PATH is $O(n)$. Since the time complexities of dSPT, dMST, and dDFS are all $O(n^2)$, the time complexity of dCDT is $O(n^2)$.

In each of the above operations, each node sends a certain message to just one node. The WAKEUP messages are sent twice at each node. The ADD, ADD_FINISH, and UPDATE messages may be sent up to $O(n)$ times. Thus, the message complexity of dCDT is $O(n^2)$.

4.2 Simulation

We evaluate its performance by conducting simulations in NS2 to measure the total power and the transmission delays of the constructed CDT, SPT and MST. We conduct the simulations for the networks sized from 20 to 300 nodes. All these networks are randomly generated in a fixed 1 km × 1 km region. For each edge $e=(u,v)$, we set its weight $w_{uv} = C_v d_{uv}^{\gamma}$, where d_{uv} is the Euclidean distance between u and v, and γ is fixed to 2, which is a typical value for an unobstructed environment, and C_v is a random constant. For all the simulations, we compare the results of the dMST algorithm, the dCDT algorithm, and the dSPT algorithm.

In order to evaluate β that reflects the total power consumption, we compare the total powers of a CDT, an MST, and an SPT. Thus, the total power is the sum of each node's maximum transmission power. In this simulation, α is set to 2. The results are shown in Fig. 2.

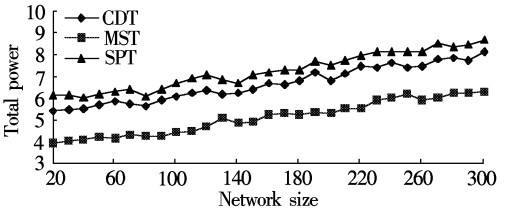

Fig. 2 Total power

From this simulation, it can be seen that the CDT has less total power consumption than the SPT does. The reason is that, for a node u, the CDT only adds the path from u to r in the SPT under the condition that the weight of the path from u to r in the MST is greater than two times the weight of the one in the SPT. This limits the total power consumption and complies with our expectations. Another observation is that, as the size of the network increases, the total power consumption does not increase much. This is because, as more and more nodes are employed into the fixed area, two neighbouring nodes are getting closer. Therefore, the total power consumption does not increase much.

Since transmission delay is highly related to the length of each path that messages traverse, the total length of the paths in the constructed tree is calculated to evaluate it. In this simulation, α is set to 2. Fig. 3 compares the total lengths of all the paths from the root to every other node and also the paths from each node to the root in each tree constructed by the MST algorithm, the CDT algorithm, and the SPT algorithm. It is shown that, for all the networks, the tree constructed only by the SPT has the smallest total length. This result coordinates with our example in Fig. 1.

From Fig. 2 and Fig. 3, the CDT has shown the ability to balance the advantages and shortcomings of MST and SPT. This satisfies the requirements of the CDTT problem to bound both the total power consumption and the transmission delays.

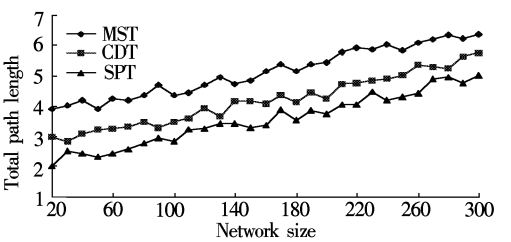

Fig. 3 Total path length(transmission delay)

5 Conclusion

In order to construct a CDS with bounded total energy consumption and transmission delay, we propose (α,β)-constraints derived from an (α,β)-tree, to solve the CDTT problem. At the same time, the distributed version of the CDT algorithm is also presented. Both the theoretical analy-

sis and simulations show that our method can satisfy the constraints of bounded total power and transmission delay. The CDT problem studies how to construct a CDT tree for one source node. We will investigate more deeply the CDT tree construction when multiple sources are presented and further extend the constraint problem from undirected graphs to directed and weighted graphs.

References

[1] Wan Pengjun, Alzoubi Khaled M, Frieder Ophir. Distributed construction of connected dominating set in wireless ad hoc networks[J]. *Mobile Networks and Applications*, 2004, **9**(2): 141 - 149.

[2] Dai Fei, Wu Jie. On constructing *k*-connected *k*-dominating set in wireless ad hoc and sensor networks[J]. *Journal of Parallel and Distributed Computing*, 2006, **66**(7): 947 - 958.

[3] Du Wei, Wu Li, Du Hongwei, et al. Maximal independent set and minimum connected dominating set in unit disk graphs [R]. Department of Computer Science and Engineering of University of Minnesota, 2004.

[4] Blum Jeremy, Ding Min, Thaeler Andrew, et al. *Connected dominating set in sensor networks and MANETs*[M]. Hingham, MA: Kluwer Academic Publishers, 2004: 329 - 369.

[5] Wu Yiwei, Wang Feng, Thai My T, et al. Constructing *k*-connected *m*-dominating sets in wireless sensor network [C]//*Military Communications Conference*. Orlando, FL, 2007: 1 - 7.

[6] Wu Weili, Du hongwei, Jia Xiaohua, et al. Minimum connected dominating sets and maximal independent sets in unit disk graphs[J]. *Theoretical Computer Science*, 2006, **352**: 1 - 7.

[7] Basagni S, Mastrogiovanni M, Panconesi A, et al. Localized protocols for ad hoc clustering and backbone formation: a performance comparison[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2006, **17**(4): 292 - 306.

[8] Khan Maleq, Pandurangan Gopal. Energy-efficient distributed constructions of minimum spanning tree for wireless ad-hoc networks[R]. West Lafayette: Department of Computer Science Purdue University, 2006.

[9] Li Yingshu, Thai My T. On the construction of a strongly connected broadcast arborescence with bounded transmission delay[J]. *IEEE Transactions on Mobile Computing*, 2006, **15**(10): 1460 - 1470.

[10] Khuller Samir, Raghavachari B, Young N. Balancing minimum spanning trees and shortest path trees[J]. *Algorithm*, 1994, **120**(4): 305 - 321.

[11] Thai My T, Feng Wang D L. Connected dominating sets in wireless networks with different transmission ranges[J]. *IEEE Transactions on Mobile Computing*, 2007, **16**(7): 1 - 10.

[12] Cidon Israel, Mokryn O. Propagation and leader election in a multihop broadcast environment [C]//*Proceedings of the* 12*th International Symposium on Distributed Computing*. Andros, Greece, 1998: 104 - 119.

[13] Gallager R G, Humblet P A. A distributed algorithm for minimum-weight spanning trees[J]. *ACM Transactions on Programming Languages and Systems*, 1983, **5**(1): 66 - 77.

[14] Brim L, Cerna I, Krcal P, et al. Distributed shortest paths for directed graphs with negative edge lengths, technical report FIMU-RS-2001-01 [R]. Faculty of Informatics of Masaryk University, 2001.

[15] Sharma M B, Iyengar S S, Mandyam N K. An optimal distributed depth-first-search algorithm [C]//*Proceedings of* 17*th Ann ACM Conf Computer Science*: *Computing Trends in the* 1990s. Louisville, Kentucky, USA, 1989: 287 - 294.

时延和功耗约束无线传感器网络连通支配集算法

孙彦景[1] 钱建生[1] 顾相平[1] 陈光柱[2]

([1] 中国矿业大学信息与电气工程学院,徐州 221008)

([2] 中国矿业大学机械与电气工程学院,徐州 221008)

摘要:针对无线传感器网络虚拟骨干时延和功耗的约束问题,提出(α,β)-约束的连通支配集算法. 根据(α,β)-约束定义了时延约束的连通支配树问题(CDTT),并给出构建同时符合时延约束和有限总功率消耗的连通支配树(CDT)算法. 算法分为 2 个阶段执行:首先在单位圆图上构建网络的极大独立集,然后在无向图上基于不同权值的最小生成树和最短路径树,构造满足(α,β)-约束要求的支撑树. 理论分析和仿真结果表明提出的算法能够正确地解决 CDTT 问题,并能够构建平衡功率消耗和传输时延要求的无线传感器网络虚拟骨干.

关键词:无线传感器网络;连通支配集;传输时延;极大独立集;功耗

中图分类号:TP393. 01