INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Data Collection In Wireless Sensor Networks Using TDMA

O. Madhuri

B.Tech final year student, computer science engineering, G.Pullaiah college of engineering and technology, Kurnool

B. Vijaya Bhargavi

B.Tech final year student, computer science engineering, G.Pullaiah college of engineering and technology, Kurnool

Mr. R. Sandeep Kumar

Assistant professor, G.Pullaiah college of engineering and technology ,Kurnool

G. S. Shabbir Ahmmed

B.Tech final year student, computer science engineering, G.Pullaiah college of engineering and technology, Kurnool

Abstract:

The network topology and interferences causes significant effects on data collection and hence on sensors' energy usage. Various approaches using single channel, multichannel and convergecasting had already been proposed. In this chapter, we survey contention-free Time Division Multiple Access (TDMA) based scheduling protocols for such data collection applications over tree-based routing topologies. We classify the algorithms according to their common design objectives, identifying the follow-ing four as the most fundamental and most studied with respect to data collection in WSNs

1.Introduction

Data collection from a set of sensors to a common sink over a tree-based routing topology is a fundamental traffic pattern in wireless sensor networks. Data in such topology flows from sensor nodes (leaves) to the sink (root) of the tree. Collection of data from a set of sensors to an intermediate parent (sink) in a tree is known as converge-casting. The simplest approach forwards raw-data as-is, without performing intermediate processing or compression on transit traffic. However, saving energy is crucial for prolonging the lifetime of sensor nodes. Since the wireless communication module is usually accounted for most of the power consump-tion, an effective way of reducing power consumption is to reduce the amount of transmitted data. Hence, some functions might be applied to aggregated and compress data, so that the actual payload uses only a fraction of the maximum allow load Whereas applications like weather for-casting, under-water observations needs continuous and fast data delivery for analysis, for longer periods. Here in this paper our emphasis is on such applications focusing on fast data streaming from sensor to sink node.

2.System Modeling

A WSN is modeled as an undirected graph G = (V; E), where V contains all nodes and E contains all communication links between nodes. Let s is the sink node such that s Є V. The distance between two nodes i and j is denoted by d_{ij}. All the nodes other than s generate and transmit data packets through a network path to sink s. Let, T = (V, E_T) is a spanning tree on G where E_T E and represents the tree edges. We adopt a time-division model by dividing time into fixed-length slots. Each k consecutive slots are grouped together and called a frame. In each frame, each node v_i in T will be assigned a wake-up slot s_i 2 f0; 1; : : : ; k ¡1g. During slot s_i, v_i must wake up to announce a beacon to synchronize with its children and then collect sensory data from them. Excluding s_i, v_i may go to sleep. The value of k should be large enough to ensure each node to find a slot.

The assignment of wake-up slots should meet two goals simultaneously: (i) the communication must be interference-free and (ii) the overall reporting latency from leaves of T to the sink should be minimized. To address goal (i), one typical approach is to enforce a node not to use the same wake-up slot as any of its 1-hop and 2-hop neighbors. However, in our data collection scenario, since not all nodes are involved in

the communication and communication directions are always toward the sink, a node only needs to consider a tighter set of interference neighbors

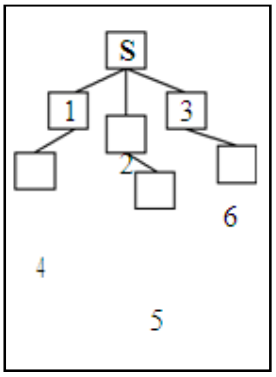

Figure 1: Tree Topology

3.Data-Gathering in Wireless Sensor Networks

Data-gathering is a common task in wireless sensor net-works. Nodes are deployed in order to collect and report sensor readings to a single sink, that, e.g., writes all data to a database. In particular, we consider using a flat as well as a hierarchical/clustering architecture to realize many-to-one communications. The capacity of the network under this many-to-one data-gathering scenario is reduced compared to random one-to-one communication due to the unavoidable creation of a point of traffic concentration at the data collector/receiver.

4.Query Processing

Sensor networks generate a large amount of data for extracting information, we need to collect and query the data from sensor networks. The primary focus is on aggregate summarized data. A query can be a request for information or orders to collect more data. When a user requests a query at the sink node to the sensor network the query is disseminated across the network. In response to the query system builds a routing tree at the sink. The probable queries for the sensor networks can be categorized into:

1) Simple Queries:

These are non aggregate queries :E.g. ”SELECT temperature FROM sensor WHERE node=z”.

These are generally mapped into broadcast or point to point queries

5.Complex Queries

They may contain sub queries: e.g."SELECT temperature FROM sensor WHERE room=(SELECT room WHERE floor='3')"

6.Event Driven Queries

These are the continuous queries that return values periodically at specified time intervals. E.g. SELECT AVG (temperature) FROM sensor where node=z". The query is similar to SQL, supporting the SQL clauses like SELECT,FROM,WHERE,GROUP BY,HAVING and aggregate clauses like MAX,AVG,MIN,COUNT and Sum. The difference is they support continuous monitoring queries by adding the clauses like DURATION and EVERY representing the lifetime of the query and the rate of answering the query.

7.Data Aggreagation

To support queries over sensor networks , the distributed data over sensor nodes need to be processed. This poses up an implicit requirement for aggregating the data. There are two approaches for data aggregation

- Centralized Approach
- In.-Network Aggregation

8.Centralized Approach

This is an address centric approach where each node sends data to central node via the shortest possible route using a multi-hop wireless protocol. The sensor nodes simply send the data packets to a leader, which is a powerful node. The leader aggregates the data which can be queried. The underlying routing protocols need minimal changes. Wireless protocol like AODV can be caused. Each intermediate node has to send the data packets addressed to leader from the child nodes. So a large number of messages have to be transmitted for a query in the best case equal to the sum of external path lengths for each node. The major drawback is the sensor networks are energy constrained and hence the approach is costly as requires a lot of messages to be exchanged for each query.

9.In-Network Aggregation

This is a data centric approach where the intermediate nodes can look at the content and perform aggregation on multiple packets it receives. The transmitting and receiving cost is greater than the computing cost. This has been the motivation for in-network aggregation. It implies the shifting of a part of the computation from clients to the sensor nodes aggregating the results or filtering the irrelavent data records with an aim of reducing the message transfer and efficient use of bandwidth there by increasing the life time of the system.

10.Main Design

To address the MDCD problem, we first analyze the lower and upper bounds on the delay for data collection, and give the procedure of proofs. Furthermore, we introduce a novel concept, the VGN to convert the MDCD problem into max-flow problem with special constraints. Based on VGN, we propose a MDCD algorithm to obtain the optimal collection paths with minimum delay in polynomial time.

10.1.The Lower and Upper bounds on the Data Collection Delay

- Lemma 1: If all the interfering links are limited, the delay for data collection is tightly lower bounded by S(sink)+(N−1)T,where T denotes cycle period of working schedule,

Node \ Slot	3	6	7	10	11
1	-	3	-	-	-
2	-	-	-	3	-
3	4	-	4(1)	-	4(2)

Figure 2: One of the best cases. (a) a network topology with given working schedule of each node. (b) Node ids from which packets are received by their corresponding parents in different timeslots.

S(sink) is the active start time of the sink in T , and N is the number of source nodes.

Proof: The best case of data collection paths is to keep the sink busy in receiving packets for all its active time. Since the sink can receive at most one packet at each active

timeslot, the number of sink's active timeslots should be at least equaled to the number of source nodes. Thus, S(sink)+(N − 1)T is a trivial lower bound to receive all packets.

Next, we prove that the lower bound is tight by shown one of the best case in Fig. 2. In Fig. 2(a), the symbols in braces indicate the working schedule of each node, and the numbers inside the circles represent the node ids. From the Fig. 2(b), we can see the packets delivery process and the schedule showing the received packets from the associated senders by each parent on different timeslots. The number in brace represents the source node which generated the current transmitted packet. According to the schedule shown in Fig. 2(a), we can see all packets are collected to the sink at timeslot 11. In this case, the S(sink) is timeslot 3, the number of source nodes is 3 and the cycle period is 4 timeslots. We can easily obtain that the lower bound is 11 timeslots. We can see that this case achieves the lower bound. Thus, the lower bound S(sink)+(N − 1)T is tight.

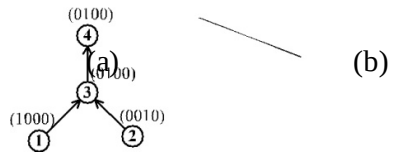

Figure 3: One of the worst cases. (a) a network topology with given working schedule of each node. (b) Node ids from which packets are received by their corresponding parents in different timeslots.

- Proof: Consider the worst scheduling principle: each packet begins to deliver after the other finishing collected to the sink one by one. T_{min}^{i} denotes the minimum time that source node i requires to deliver a packet to the sink, which can be easily computed by any one of Shortest Path Routing (SPR) algorithms, while the other packets are waiting until the current packet arrived at the sink. In this case, only one link in the network can be scheduled at each timeslot, which leads to the most time, i.e. the sum of minimum time that source node i requires to deliver a packet to the sink, to complete the data collection.

Next, we prove that the upper bound is tight by shown one of the worst case in Fig. 3. According to the schedule shown in Fig. 3(b), we can see all packets are collected to the sink at timeslot 18. In this case, T_{min}^{1}, T_{min}^{2} and T_{min}^{3} are 6, 10 and 2 timeslots respectively. The sum of them equals to 18, the same as the data collection delay in this case, which means

$\sum^{N}$

the upper bound achievable. Thus, the upper bound

$i=1$ is tight.

- Virtual Grid Network: To make the node states and packet transmission process more clearly, we define and detail a concept, VGN inspired by the time-expanded network proposed in [2]. The edge construction regulations of VGN are different from the time-expanded network. We first show how to construct the VGN based on the original directed communication graph G helping us better understanding the data collection method proposed later.

  For simple presentation, we first classify the nodes in G into three types by different roles.

- Leaf node: only acts as a source node, i.e., transmits its own sensor reading.
- Intermediate node: acts as double roles of a source node and a relay node, i.e., not only transmits its sensor reading, but also receives a packet and tries to forward it when its neighbors awake. The own generated packet has higher priority to transmit than the forwarded packets.

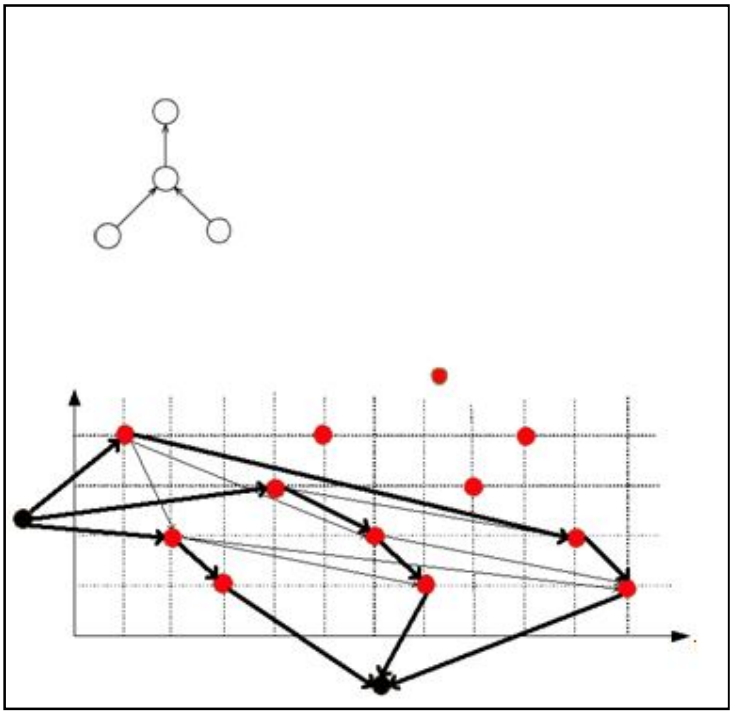

Figure 4: VGN transformation. (a) A simple network. (b)The Virtual Grid Network

- Sink node: responsible for receiving packets.

Given a communication graph G = (V; E) of deployed WSN where V with cardinality n denotes the node set and E is the edge set. Assume that the working schedule for each wireless node $n_i \in V$ is given, we build a VGN according to the following rules.

- For each node n_i is active at time $t \in [0; T]$, we build a virtual active node $N_{i;t}$ in VGN.
- When node n_i is a leaf node, and has a directed edge to the node n_j in G, if the First active time of n_i is t and the active time of node n_j is p after time t, we add a direct directed edge from $N_{i;t}$ to $N_{j;p}$ in VGN. Since the leaf node only transmits its own sensor reading, it is not possible for n_i to have new arrived packet at the other active time except the first one.
- When node n_i is a intermediate node and has a directed edge to the node n_j in G, if the active time of node n_i and n_j are t and p (p; $t \in [0; T]$) respectively and p > t, we add a directed edge from $N_{i;t}$ to $N_{j;p}$ in VGN.

For the raw data collection, the MDCD problem can be obtained an optimal result by reduced into the max-flow problem. In order to formulate a max-flow problem over the VGN, we introduce two virtual vertices, the super source and the super sink, symbolized by s and d respectively, to represent the source and destination of the total flow over the graph. We complement the rules to build the connection from s and d to the virtual active nodes.

- When node n_i is a sink, we connect all its corresponding virtual active nodes to the super sink d.
- We build the edges from the super source s to the corresponding First active virtual nodes of all source nodes.

Here, the first rule illustrates the regulations to establish all nodes of VGN. The remainder rules depict how to build edges to connect nodes in VGN according to G. We take the simple network in the best case mentioned in Fig.2 as an example to give a walk-through of VGN construction (shown in Fig.)

- 4).In the following, we details how to construct VGN graph in terms of original network topology and working schedule of each node. First, we set the nodes in VGN based on the working schedule of each node in G by Rule 1 and the red nodes in VGN represent the virtual active nodes. Next, we describe how to construct corresponding edges in VGN according to Rule 2 − 5.

For the leaf node v_1, it can send a packet to its neighbor v_3 when v_3 is active. Thus, we establish the edges of $N_{1;1} \rightarrow N_{3;2}$, $N_{1;1} \rightarrow N_{3;6}$ and $N_{1;1} \rightarrow N_{3;10}$ in VGN by Rule 2. For another leaf node v_2, it can also send a packet to its neighbor v_3 when v_3 is active. According to the same rule, we build the edges of $N_{2;4} \rightarrow N_{3;6}$ and $N_{2;4} \rightarrow N_{3;10}$ in VGN.
For the intermediate node v_3, it can send a packet to its neighbor v_4 when v_4 is active. Hence, we establish the edges
from $N_{3;2} \rightarrow N_{4;3}$, $N_{3;2} \rightarrow N_{4;7}$, $N_{3;2} \rightarrow N_{4;11}$ in VGN.
At the same time, as a relay node, node v_3 can receive packets when it turns to be active, and forwards packets when its neighbor is active. Accordingly, we establish the edges
$N_{3;6} \rightarrow N_{4;7}$, $N_{3;6} \rightarrow N_{4;11}$ and $N_{3;10} \rightarrow N_{4;11}$ in VGN.
For the sink v_4, we build the edges from the corresponding virtual active nodes to the super sink d by Rule 4 , i.e., $N_{4;3} \rightarrow d$, $N_{4;7} \rightarrow d$ and $N_{4;11} \rightarrow d$. Finally, we build the connection from the super
source s to the corresponding first active virtual nodes of all source nodes by Rule 5, i.e., $s \rightarrow N_{1;1}$, $s \rightarrow N_{2;4}$ and $s \rightarrow N_{3;2}$. The whole mapping process for one of the best cases is completed shown in Fig. 4.

- The Max-Flow Problem: Given the VGN, our next step is the formulation of an optimization problem whose objective is to maximize the flow from s to d, i.e., the most number of source nodes from which the sink can collect data successfully under the given time duration T . f(*,*) indicates the binary traffic flow over an edge connecting two vertices in VGN. Denoting by F(*,*) the total flow from the source to the destination, our objective can be described as

$$\max\{F(s; d)\} \qquad (4)$$

The data collection delay is defined as the total time required to complete data collection, which depends on the maximum EED for each source node. Minimizing the maximum EED problem is equal to the max-flow problem in VGN, which needs to be solved taking into account several constraints due to, e.g., interference influence, half-duplex transceiver limitation. We detail such constraints below.

- Constraints:Non-negative flow and flow conservation: the flow on every

existing edge must be greater than or equal to zero. Also, for any vertex in the VGN, the amount of flow entering the vertex must equal to the amount of outgoing flow. Mapping to the VGN, the constraint can be expressed as

$$\sum_{p \in N(i)} f(N_{p;t_p}; N_{i;t}) = \sum_{q \in N(i)} f(N_{i;t}; N_{q;t_q})$$

where $t_p < t < t_q \in [0; T]$

The function N(i) indicates the set of node i's neighbors in the original network G.

Half-duplex transceiver limitation: Due to the hardware function limitation, a node cannot transmit and receive packets simultaneously shown in Fig. 1(a). Mapping to the VGN, the constraint can be expressed as

$$f(N_{i;t_1}; N_{j;t_2}) + f(N_{j;t_3}; N_{m;t_2}) \leq 1$$

where $t_1; t_2; t_3 \in [0; T]$

where the binary function f(*,*) is equal to 1 if the edge carries a flow, otherwise is 0. We notice that the case shown in Fig. 1(a) happens only when j and m are the neighbors and active at the same time in the original topology.

Meanwhile, a node cannot receive from more than one neighbor at the same time, shown in Fig. 1(b). For this constraint, we set the node capacity in VGN to one unit, i.e., $C_{Ni;t} = 1$. We can deduce that the edge capacity in VGN satisfies the conditions below.

$$\sum_{p \in N(i)} f(N_{p;t_p}; N_{i;t}) \leq C_{N_{i;t}} = 1$$

where $t_p \leq t \in [0; T]$

$$\sum_{q \in N(i)} f(N_{i;t}; N_{q;tq}) \leq C_{Ni;t} = 1$$

where $t \leq t_q \in [0; T]$

Since we assume that the interfering links are eliminated, the max-flow problem is completed to formulate.

- Minimum Data Collection Delay Algorithm: After converting the MDCD problem into a max-flow prob-lem, we present a novel MDCD algorithm inspired by the Ford-fulkerson max-flow method to solve the MDCD problem, and obtain the optimal solution in polynomial time

In MDCD algorithm, the working schedule for each node needs to be known by the sink in the network initialization, which can be easily achieved through exchange of the hello messages. The initial expanding time of VGN is set to be the maximum value of the set of minimum time for each packet requiring to reach the sink by Dijkstra Algorithm. The maximum flow f_m is set to zero at the initialization step. Each iteration of the outmost while loop corresponds to one phase. In each phase, the time threshold T is monotonously increased by the step of cycle period T . The number of phases stops to increase until the maximum flow satisfied by all constraints is equal to the number of source nodes. This is an indication that all packets can be scheduled within T without collision. Obviously, the result P is the optimal collection paths for all source nodes.

Algorithm 1 Minimum Data Collection Delay Algorithm Input: The given VGN

Obtain the maximum flow f_m of (G; s; d; c; Δ)

and

optimal flow paths P .

- k = k + 1
- Among the $N_{i;t}$ through flow in the k^{th} T period, find the the largest t of $N_{i;t}$. over the time expanded.

- Output: The optimal data collection paths with the minimum delay. ⌊ ⌋
- 1: Initialization: $\Delta = \max^{n-1}\{T^{i}\}$, $f^{m} = 0$, $k = {}^{\Delta}$
  - i=1 min T
- while $f_m < n - 1$ do
- Update the sub-graph of V GN under the time threshold
  - kT .
  - while there is an s-d path in the residual graph $G_f(\Delta)$
  - Do

- P be a simple s-d path in $G_f(\Delta)$
- $f' = augment(f; P)$
- if f' satisfies the constraints in the above max-flow
- problem then
- Update f to be f'
- Update the residual graph $G_f(\Delta)$ to be $G_f'(\Delta)$

- T_{min}= the largest t.

- Theorem 1: If all the interfering links are eliminated, MDCD algorithm can obtain the optimal data collection paths with the minimum delay, i.e., $S(sink)+(N-1)T$.
  - Proof: The max-flow problem is to find the max flow from the source s to the destination d. In the VGN, the neighbors of super sink come from the corresponding virtual active nodes of the sink. Therefore, in order to make the flow maximum from s to d, the corresponding virtual active nodes of the sink try to carry flow as much as possible. When all corresponding virtual active nodes of the sink carries flow, the sink keeps receiving packets in all active time. At this moment, the MDCD algorithm returns the optimal collection path with the minimum delay, i.e. $S(sink) + (N-1)T$. In the best case illustrated above, the optimal collection paths returned by MDCD algorithm are shown in bold in Fig. 4, and the minimum collection delay achieves the lower bound.

11.Node ID

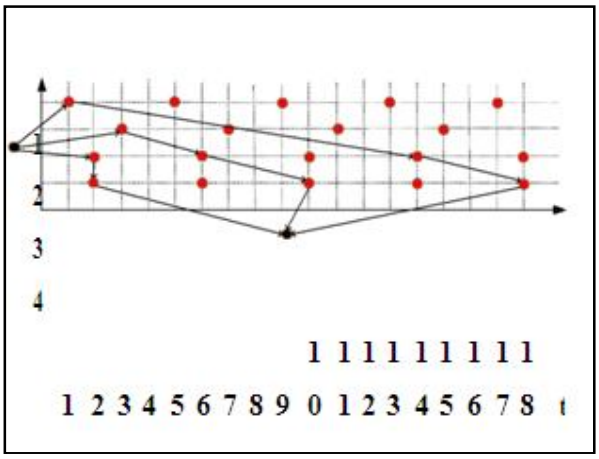

Figure 5: The optimal data collection paths of the worst case returned by the MDCD algorithm

Thus, the MDCD algorithm can obtain the optimal collection paths with the minimum delay.

In addition, due to the half-duplex transceiver constraints, it sometimes fails to make all corresponding virtual active nodes of the sink carried flow, shown in the worst case mentioned above. But it can also obtain the optimal solution. Fig. 5 shows the optimal collection paths returned by MDCD algorithm. However, the returned minimum delay will not achieve the lower bound, since it applies in the best case.

- Lemma 3: The total time complexity for MDCD algorithm is $O(T^3n^3)$. Here, T is the minimum time threshold when the maximum flow is equal to the number of source nodes.
- Proof: n is the number of vertices in G. n' and m' are denoted as the number of vertices and edges in VGN, re-spectively. We can easily obtain the relationship that $n' \approx kn$, where $k = \lceil T=T \rceil$. To keep network connectivity, we assume algorithm is O(mC) when all capacities are integers, where m is the number of edges in flow network and C is the upper bound of the maximum flow [14]. It is obvious that the upper bound of maximum flow is n, and the number of edges in flow network is m'. Thus, the inner While loop can be implemented to run in $O(nm')$ time. Since the outer While loop run for k iterations, the running time of MDCD algorithm is $O(knm')$, namely $O(kn \cdot \frac{1}{2} n'(n' -1))=O(kn \cdot \frac{1}{2} kn(kn-1))=O(T^3n^3)$.

12.Conclusion

This paper addresses the overload problem around nodes that are aggregating data, and provides a solution to improve system throughput and reduce energy con-sumption. The proposed MAC layer solution makes use of burst transmissions with low-overhead local advertise-ments to avoid contention during the bust-periods. Using extensive simulations, we observe that our proposed ap-proach can achieve up to two times higher throughput and four times higher energy efficiency than CSMA in static event scenarios, with an increasing performance gap as the network gets overloaded (higher node densities and/or lar-ger event sizes). These observations are also supported by the experiments on the Kansei testbed on different data rates. To apply ClearBurst in moving event scenarios, the location of C-node must migrate as the event moves. Through extensive simulations, we showed that ClearBurst with the C-node migration protocol can have four times higher throughput and two times more energy efficiency than CSMA. However, when the event moves fast, Clear-Burst does not show significant performance gain in com-parison to CSMA + SP. The reason for this performance degradation is due to the overhead of C-node migration and the reconfiguration of the data collection tree. The lat-ter is a time consuming operation which limits how fast the protocol can react to the congestion arising at the new location of the event. In summary, our proposed ap-proach is highly suited for data collection applications in sensor networks, especially for static and slow moving events. As to fast moving events, a more efficient data col-lection protocol is needed for event tracking and fast con-gestion resolution. These requirements impose many challenges in different network layers and will be further studied in our future works.

13.Reference

1. Gahng-Seop Ahn, Se Gi Hong, Emiliano Miluzzo, Andrew T. Campbell, Francesca Cuomo, Funneling-MAC: a localized, sink-oriented MAC for boosting fidelity in sensor networks, in: Proc. of SENSYS, 2006, pp. 293–306.
2. S. Gandham, Y. Zhang, and Q. Huang, "Distributed Time-Optimal Scheduling for Convergecast in Wireless Sensor Networks," Computer Networks, vol. 52, no. 3, pp. 610-629, 2008.
3. X. Chen, X. Hu, and J. Zhu, "Minimum Data Aggregation Time Problem in Wireless Sensor Networks," Proc. Int'l Conf. Mobile Ad- Hoc and Sensor Networks (MSN '05), pp. 133-142, 2005.
4. M. Pan and Y. Tseng, "Quick Convergecast in ZigBee Beacon-Enabled Tree-Based Wireless Sensor Networks," Computer Comm., vol. 31, no. 5, pp. 999-1011, 2008.
5. W. Song, F. Yuan and R. LaHusen, "Time-Optimum Packet Scheduling for Many-to-One Routing in Wireless Sensor Networks," Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS '06), pp. 81-90, 2006.
6. W. Song, H. Renjie, B. Shirazi, and R. LaHusen, "TreeMAC: Localized TDMA MAC Protocol for Real Time High Data Rate Sensor Networks," Journal of Pervasive and Mobile Cosmputing, vol. 5, no. 6, pp. 750-765, 2009.
7. H. Choi, J. Wang and E. Hughes, "Scheduling for Information Gathering on Sensor Network," Wireless Networks, vol. 15, pp. 127-140, 2009.