Wireless Sensor Network, 2009, 1, 482-488
doi:10.4236/wsn.2009.15058 Published Online December 2009 (http://www.scirp.org/journal/wsn).

Dynamic Load Balancing with Overlay-Based Reconfiguration for Wireless Sensor Networks

Hang QIN[1], Li ZHU[2], Zhongbo WU[3]
[1]*Computer School, Yangtze University, Jingzhou, China; State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China*
[2]*Computer School, Wuhan University, Wuhan, China*
[3]*Department of Computer Science, Xiangfan University, Xiangfan, China; Information School, Renmin University of China, Beijing, China.*
Email: {*hangqin*100, *yeah*_1397118 }*@hotmail.com, rucwzb@*163.*com*
Received August 7, 2009; *revised September* 20, 2009; *accepted September* 25, 2009

Abstract

Wireless sensor networks are characterized by multihop wireless links and resource constrained nodes. In terms of data collection and forwarding scheduling, this paper investigates the load balancing in sensor nodes and wireless link based on the performance of wireless sensor networks. Leveraging the property of dissimilarity distribution, a method to quantitatively evaluate the benefits of load balancing is presented, in order to access the profitability. Then a novel Dynamic Load Balancing of Overlay-based WSN (DLBO) algorithm has been put forward. In particular, the tradeoff between transferring ratio and the load imbalance among nodes is discussed. The load balancing method in this paper outperforms others based on balancing factor, different nodes number and data scales of applications. The proposed model and analytical results can be effectively applied for reliability analysis for other wireless applications (e.g., persistent data delivery is involved).

Keywords: Wireless Sensor Networks; Workload; Dynamic Load Balancing; Dissimilarity Measure; Reconfiguration

1. Introduction

Wireless Sensor Networks (WSN) is an accumulation of sensors interconnected by wireless communication channels. Under the control of the network, every sensor node is a small device that can collect data from surrounding area, communicate with each other, and carry out computation. Long distance communication is normally achieved in a multi-hop manner. Thanks to recent advances in remote monitoring system [1–3], such networks are progressing rapidly, and are expected to be popular in applications such as environment monitoring, intrusion detection, and earthquake warning.

Whereas sensor networks scale up in size, effectively managing the distribution of the networking workload will be of great concern [4–5]. Actually, one of the most urgent challenges in designing protocols for a WSN is to make them more energy-efficient by maximizing the network performance without any loss of sensing capability. By extending the workload across a sensor network, load balancing reduces hot spots in sensors nodes and increases the direct communication of the sensor network.

Nonetheless, existing strategies miss one vital measure in the optimization space for routing correlated data, namely, the data communication and nodes dispatching cost. Common intuition tells us that the hot spot problem can be solved by varying the static data apportionment among sensor nodes at different distances to the sink，so that data content can be more evenly dispensed [6–8]. Moreover, as far as the static and mobile sensors in a hybrid WSN is concerned, this is only true to some extent. In terms of the overlay-based WSN, the dynamic load balancing scheme has been shown evenly distribute packet traffic generated by sensor nodes across the different branches of the routing. Therefore, analysis and improvement of sensor nodes dispatching schedule and dissimilarity is the critical metric in collaborative information processing.

2. Related Work

The dynamic load balancing routing problem focused on

here is, as indicated in [9], directly bound up with dynamically reallocate incoming external loads at each node [10]. Fatta and Berthold [9] propose distributed process algorithms that may use initiated receiver and still achieve good load balancing, measured by the load balancing index, simultaneously for a dynamic partitioning of the search space. They also show that it naturally tolerates node failures and communication latency and supports dynamic resource aggregation. Furthermore, in [11], the tradeoffs between load balancing and fail-over implementation strategies, and present quantitative performance measurements collected on a commercial multi-homing load balancing system, are addressed in details.

Note that the majority of load balancing strategy developed heretofore is based on such time delays [12], and it proceeds with the assumption that delays are deterministic. In reality, delays are random in such communication media, especially in the case of WLANs. Furthermore, load balancing is attributable to uncertainties associated with the amount of traffic, congestion, and other unpredictable factors within the network. However, the scale of these load balancing methods is partially restricted. For example, the general dispatching mobile sensors problem is overlooked and these nodes are presented as a cluster head usually within a cluster. Thus the load balancing is somewhat limited.

By spreading the workload across a sensor network, H. Dai and R. Han [13] developed a node-centric algorithm that constructs a load-balanced tree in sensor networks of asymmetric architecture, and select the heaviest nodes with the maximum growth space for growth. Y. Wang *et al.* [14] have revealed a notion of a hybrid sensor network consisting of static and mobile sensors. They propose an efficient clustering scheme to group event locations so that the maximum-matching approach can still be applied.

3. System Architecture

3.1. Overlay-Based Reconfiguration Policy

The sensor nodes are organized through three layers in *Dynamic Load Balancing of Overlay-based* WSN (DLBO). As shown in Figure 1, *Ordinary Sensors Overlay* belongs to the first layer, which keeps the records of some sensor nodes tracking the current data dispatching information. It can be viewed as a large list structure, initialized by the sensor nodes list fetched from *Reconfiguration Tracker* and updated by received data accumulation messages. *Grid Heads Overlay* remains with the second layer which forms a ring-assisted control overlay. The repetitive connection is between any two neighboring nodes, where data collection messages are distributed. *User Overlay* pertains to the third, which constructs the data overlay of DLBO. The data fusion is only interchanged between associates.

For containing the dynamics of overlay topology and enhancing the end-to-end performance of load balancing, a selective approach is presented to raise better dissemination candidates to the upper layer. The sensor node first selects some nearest sensor nodes from *Ordinary Sensors Overlay* as its neighbors according to their distance calculated by the difference between their positioning. According to the feature, some neighbors are selected from the neighbor list into its associate list, in terms of the relevance that can be calculated by the time latency of their dynamic clustering. When a grid head has not been exchanging any data packets in a predefined time interval, it can be removed from the *Grid Head Overlay* and contribute to a procedure to find new triggered grid head; similarly, when a node record of *Ordinary Sensors Overlay* has not been updated for a predefined time interval, it can be dropped from *Ordinary Sensors Overlay*.

3.2. The DLBO Framework

Actually, to deploy mobile sensors more efficiently, it should not only cut down the total moving energy but also *balance* the loads of mobile sensors. Therefore, a distributed solution is outlined. The focus here is the load balancing gathering of the data content from the sensor nodes to the sink node. And Figure 2 illustrates a sensor network where source sensors are disseminated on grid and sensed information of the sensors is to be routed to the *grid head*. Arrow lines construct the fusion relation in which sensor nodes respectively aggregate data of different fields, initially. Note that the fusion tree may be changed later due to failures or load balancing, but the resulted relation must also satisfy the above conditions.

Specifically, each mobile sensor acquaints its location and remaining energy to its pertinent grid head. While detecting events, static sensors notify to their grid heads. For acquiring such information, a grid head performs dynamic load balancing algorithm to dispatch mobile

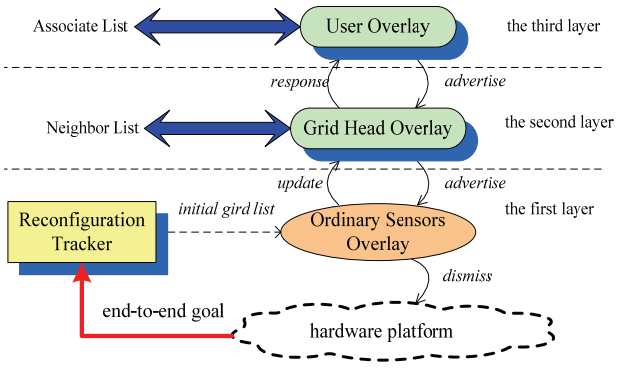

Figure 1. Overlay management of dynamic load balancing with reconfiguration tracker.

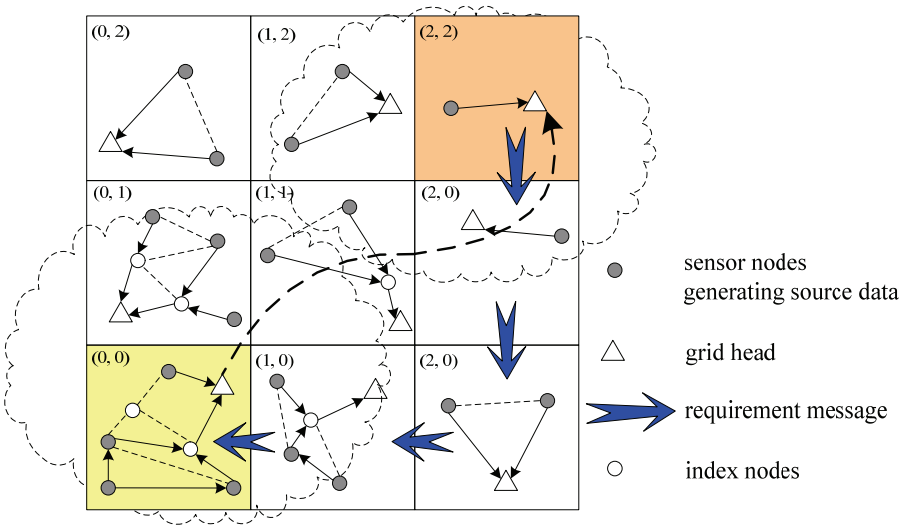

Figure 2 .The Infrastructure to show how reconfiguration works.

sensors to the events occurred in its grid. However, if there is no mobile sensor in the grid, the grid head will search available mobile sensors in other grids. Consequently, employing the distributed framework, the area covered by WSN is divided into small *virtual grid quorum*. Based on the adjoining units, the virtual grid quorum is depicted such that all nodes in one grid quorum can communicate with all nodes in the other grid quorum. In each virtual quorum, nodes operate to keep awake and work as grid head, whereas others only need to wakeup periodically. The grid head makes for forwarding messages that pass through the grid quorum.

To decrease the number of message transferring when a grid head seeks for mobile sensors in other grid quorum, each grid head sends issue messages containing the number of mobile sensors in its grid quorum to the same column of units. Furthermore, each grid head has the information of mobile sensors in other grid quorum placed in the same column. When a grid head wants to search mobile sensors in other grid quorum, it sends a *demand* message to the grid head in the same row. Because of the grid structure, it must be a grid head receiving both issue and demand messages.

This framework avoids both unnecessarily removing the sensing data and flooding control messages throughout WSN. Surely, this scheme introduces additional overhead for maintaining index nodes. However, as displayed by the analysis and evaluation results, it can still enhance the end-to-end performance.

4. Dynamic Load Balancing Algorithm Design

In this paper, a hybrid WSN consisting of static and mobile sensors is considered. Static sensors form a connected network and fully cover the area of interest to continuously monitor the environment. Table 1 lists all

Table 1. Notation.

Symbol	Definition
G	$=(V, E)$, the sensor network
V	the set of sensors (nodes)
E	the set of edges representing the communication links between pairs of sensors
T,	the Scheduling Task
D	the Sensing Data
s_m	$s_m \in T$, message
w	the workload to be moved
LB_t	the Load Balancing tag
n	the iteration number
j	the interrupting time, which is a integer
$\mathbb{V}$	the number of sensors
$\mathbb{V}'$	the physical connection number of the given sensor
m	the round number
$\mathbb{E}_m$, $\mathbb{E}_{max}$	every sensor's independent load function and maximal load
$\mathbb{E}_m(k)$	the workload of sensor m in kth stationary load time interval
B	networks bandwidth
ε	workload for computing new distribution
$\gamma_m(n)$	the iteration distribution
$\chi_m(n)$	the no-executing iteration number for sensor m after the time of the synchronization n
t_n	the time cost in the synchronization n
τ	the iteration's time cost can be depicted in agreement loop
f	the first sensor which has finished the task allocation
ξ	the synchronization workload
η	the demanding synchronization number
$\mathbb{E}(n)$	the data transmission workload
$\psi(n)$	the workload in the synchronization n's communication
$\varsigma(n)$	the number of transmission task and array's message amount
TW	the total workload
$\Omega_g(n)$	balancing factor based on the synchronization time of the different grids
$v(j)$	the number of the grids in the waited queue of the centralized load balancing manager

notation in this paper, and the problem of dispatching mobile sensors is modeled in Table 1.

The sensor network is modeled as a graph $G=(V, E)$, where V denotes the set of sensors (nodes) and E the set of edges representing the communication links between pairs of sensors. The dynamic load balancing architecture is executed on G to construct a load-balanced tree. The load associated with a given sensor node represents the amount of data periodically generated by that sensor node. Actually, the above framework absorbs the nodes generating the greatest load to the lightest branches to achieve balance.

The study identifies the Dynamic policy in DLBO, it is composed of a *Scheduling Server's Algorithm* and a *Task Counter's Algorithm*. Firstly, *Scheduling Server's Algorithm* derives a task and divides sensing data to every task equally. The algorithms are detailed below for the sake of completeness:

Epoch 1: *Scheduling Server's Algorithm*

Initialization:
1. **derive** T to every sensor node, and equally **allocate** D into the set of T, where T is Scheduling Task, D is Sensing Data

Iteration:
2. **receive** any message from s_m, $s_m \in T$;
3. **if** (message∈workload's neighboring information) **then fill** in corresponding overlay;
4. **if**($w_m = 0$) **then**
5. **search** hotspot task s_m by $MAX\{t_1, t_2, \ldots\}$;
6. get the sensing data's size
7. **if** (w match the tradeoff condition) **then**
8. **mount** LB_t
9. **send(** LB_t, s_n, w**) to** s_n;r
10. **interrupt** s_n, **send(**s_n, w**) to** s_n;
11. **if** (T is Null) **then exit**, all task has been finished.

Secondly, while one task has been finished, *Task Counter's Algorithm* can get the condition and granularity of load balancing from the Task's optimal size.

Epoch 2: *Task Counter's Algorithm*

1. **interrupt** initialization, **set** interrupting time, $j \leftarrow 0$;
2. **if**(j<=*uptoround*−1) **then**
3. sensors nodes' data transferring part
4. **if**(j= =*uptoround*−1) **then**
5. **send** workload information to Scheduler, **return** load balancing parameters;
6. **receive** LB_t, s_n and w;
7. **if** (LB_t trigger scheduling) **then**
8. **receive** w packets from s_n;
9. *uptoround* ← w;
10. **update** node's data distribution;
11. $i \leftarrow i+1$;
12. **goto** (2);
13. **send** task's finishing information to the scheduler.

Finally, the *Interrupting Procedure* is as follows:

Epoch 3: *Interrupting Procedure*

1. **receive** message w from scheduler;
2. **send** w packets to task n_i;
3. *uptoround* ← *uptoround*−1;
4. **update** the sensor node's data distribution.

5. Workload Analysis with Data Communication

In this paper, the workload model is developed in order to examine the effect for load balancing strategy. One strategy's workload is primarily composed of these four parts:

- Workload for Computing New Distribution
- Synchronization Workload
- Workload for Data communication
- Workload for Dispatching Sensors

Let $\mathbb{V}$ denote the number of sensors, where $\mathbb{V}'$ is the physical connection number of the given sensor, and m is round number, n is iteration number. Let $\mathbb{E}_m$ be every sensor's independent load function, $\mathbb{E}_{max}$ is every sensor's maximal load, $\mathbb{E}_m(k)$ is the workload of sensor m in kth stationary load time interval, and B is networks bandwidth.

Firstly, as far as *Workload for Computing New Distribution* is concerned, workload can be defined by ε, and normally it is a small quantum. In this distributed strategy, every sensor node has the cost. Moreover, the workload is relatively smaller in the local strategy because of $\mathbb{V}'$ nodes in the grid event locations, and this difference can be ignored.

Secondly, *Synchronization Workload* is the hotspot node forwards interrupting request to the neighboring nodes in the synchronization process, which consequently reflects their performance and energy parameters to the *reconfiguration tracker* for load balancing. The workload can be calculated in terms of the category of data aggregation.

5.1. Definition 1: Workload for Dispatching Sensors

The demanding message number can be analyzed according to data transmission and sensors rescheduling. The iteration distribution can be identified as $\gamma_m(n)$, and $\chi_m(n)$ defines the no-executing iteration number for sensor m after the time of the synchronization n. $\Phi_{(n)} = \sum_{m=1}^{\mathbb{V}} \chi_m(n)$, and t_n depicts the time cost in the synchronization n.

Based on the influence of discrete workload, the sensor's *effective speed* is in the inverse proportion to the sensor's workload, thus the computation expression is

$S_m/(\mathbb{E}_m(k)+1)$, and $\mathbb{E}_m(k)\in\{0,...,\mathbb{E}_{\max}\}$. For the anterior synchronization sensor states the performance measurement in different mechanism, the sensor's performance can be determined by the average effective speed actually. As for the synchronization $n-1$ happens among the workload x steady time, take $x=\lceil t_{n-1}/l_{\max}\rceil$ for example, and $y=\lceil t_n/l_{\max}\rceil$ analogously. $\chi_m(n)$ is the effective workload of sensor m between the synchronization n and the synchronization $n-1$, and the average effective speed of sensor m between these two synchronization is:

$$\begin{aligned}\varepsilon_m(n)&=\frac{\sum_k^b=aS_m/(\mathbb{E}_m(k)+1)}{b-a+1}\\&=S_m/\left(\frac{b-a+1}{\sum_{k=a}^b 1/(\mathbb{E}_m(k)+1)}\right)=S_m/\theta_m(n)\end{aligned}\tag{1}$$

As for the finished iteration, the synchronization n's influence can be analyzed in the agreement loop, and the iteration's time cost is the same in the loop. To be more specific, the iteration's time cost can be depicted as τ in agreement loop. After synchronization $n-1th$ finishing, the iteration $\gamma_n(n-1)$ can be allocated to every sensor. Furthermore, f defines the first sensor which has finished the task allocation, then the time cost in sensor f is:

$$t=t_n-t_{n-1}=\frac{\gamma_f(n-1)*\tau}{\varepsilon_f(n)}\tag{2}$$

The unfinished iteration number in sensor m can be obtained from the old iteration distribution subtracting the iteration number of t time interval:

$$\chi_m(n)=\gamma_m(n-1)-\left\lceil\frac{t*\varepsilon_m(n)}{T}\right\rceil=\gamma_m(n-1)-\gamma_f(n-1)\left(\frac{\varepsilon_m(n)}{\varepsilon_f(n)}\right)\tag{3}$$

The disagreement loop can be deal with according to disagreement loop as follows, and the time interval of sensor f having finished its allocated task is:

$$t=t_n-t_{n-1}=\sum_{k=1}^{\gamma_f(n-1)}\frac{\tau_k}{\varepsilon_f(n)}\tag{4}$$

Thus k belongs to the set of allocating iteration for sensor f. The finished iteration of sensor m in time t can be defined as $\gamma\ll\gamma_m(n-1)$, it can be denoted as $\sum_{k'=1}^{\gamma}\frac{\tau_{k'}}{\varepsilon_m(n)}\geq t$.

Take t into this expression, and shift $\varepsilon_m(n)$ into the other side:

$$\sum_{k'=1}^{\gamma}\tau_{k'}\geq\left(\frac{\varepsilon_m(n)}{\varepsilon_f(n)}\right)\sum_{k=1}^{\gamma_f(n-1)}\tau_k\tag{5}$$

So the unfinished iteration number in sensor i is $\chi_m(n)=\gamma_m(n-1)-\gamma$. In the new distribution, the unfinished task number in all sensors is $\Phi(n)=\sum\chi_m(n)$, and it is in the direct proportion to sensor's average effective speed:

$$\gamma_m(n)=\left(\frac{\varepsilon_m(n)}{\sum_{k=1}^{\mathbb{V}}\varepsilon_k(n)}\right)*\Phi(n)\tag{6}$$

The former allocated task quota in every sensor is the same, thus: $\theta_m(0)=1$, $\gamma_m(0)=I(N_{ad})/\mathbb{V}$, and $\chi_m(0)=\gamma_m(0),\forall m\in V$. Supposing that information can be seized in advance, $\theta_m(0)$ will be in the direct proportion to the asynchronous sensors' initial speed or sensor's initial workload. The *recurrence function* can be proposed from the above deduction, while the new distribution and the entire iteration number for every unfinished synchronization can be got in terms of their solution. As all the tasks have been finished, the terminating condition $\Phi(\eta)=0$ happens.

Let ξ denote the synchronization workload, η depicts the demanding synchronization number, ε is the computation workload in the redistribution, $\mathbb{E}(n)$ is the data transmission workload, and $\psi(n)$ defines the workload in the synchronization n's communication. Based on transferred task amount, the transferred basic task unit(normally it is iteration relation) in one synchronization process:

$$\rho(n)=\frac{1}{2}\left(\sum_{m=1}^{\mathbb{V}}\left|\chi_m(n)-\gamma_m(n)\right|\right)\tag{7}$$

In terms of data transmission workload, the transmission iteration brings about array's removing. $\varsigma(n)$ defines the number of transmission task and array's message amount, and it can be obtained from the computation of the old distribution value and the new distribution value. ρ belongs to the set of distribution array to be renewed. And the total workload in data transmission can be given from the expression:

$$\mathbb{E}(n)=\varsigma(n)*\mathbb{E}+\rho(j)*\sum_{\rho}\left|D_\rho/B\right|\tag{8}$$

5.2. Definition 2: Workload for Data Communication

Because load balancing manager has to forward the task and data transmission message to sensor node $\varsigma(n)$, this workload can be got from the centralized mechanism. The number of instructions and of messages in transmission data is the same, for these instructions are forwarded to the sensors which need to forward data. As a result, the workload for data communication is $\psi(n)=\varsigma(n)\mathbb{E}$, in terms of the centralized mechanism. And the workload for data communication $\psi(n)=0$, it is according to the distributed mechanism.

5.3. Definition 3: Total Workload

In the global strategy, with the solution of the above *recurrence relation* set, the data transmission workload and the synchronization amount (Expression (7)) can be got, and it can conclude the overall workload in the total

strategy:

$$TW = \eta(\xi + \varepsilon) + \sum_{n=1}^{\eta}[\mathbb{E}(n) + \psi(n)] \quad (9)$$

In the local strategy, even if the load balancing manager is asynchronous, it is not true that the grid's handling is independent with the others. This is the reason why the load balancing manager can only handle the other grid after finishing the computation of redistribution and communication.

6. Performance Analysis

6.1. Methodology

In this section we present the performance analysis of DLBO by using simulation programs in C++ programming language. The simulator is set up as follows: It is supposed that the surveillance has a square region of size $50m \times 50m$. We assume that each node produces one unit data (400 bytes) and sends it to the sink located at the bottom-right corner. All sensors act as both sources and routers. We also performed a set of experiments with different numbers of sensors and different sizes of sensing data.

6.1.1. Definition 4: Balancing Factor

This influence can be modeled as every grid's balancing factor, it is dependent on the synchronization time of the different grids:

$$\Omega_g(n) = \sum_{k=1}^{v(n)}[\varepsilon + \psi_k(n)] \quad (10)$$

$v(j)$ is the number of the grids in the waited queue of the centralized load balancing manager. In the local distributed mechanism, for the inexistence of centralized load balancing, this influence does not happen ($\Omega_g(n) = 0$). Maybe it has the influence for the overlap synchronization communication, but it can also be neglected.

For the local mechanism, every grid's workload in different mechanism is not the same, and the total workload in every grid is:

$$TW_g = \eta_g(\xi + \varepsilon) + \sum_{n=1}^{\eta_g}[\mathbb{E}_g(n) + \psi_g(n) + \Omega_g(n)] \quad (11)$$

The total workload in the local strategy is the time of the last grid finishing its computation:

$$TW = MAX\{TW_1, TW_2, ..., TW_{\lceil \mathbb{V}/\mathbb{V}_n \rceil}\} \quad (12)$$

6.2. Simulation Results

Next, the DLBO scheme's capability to achieve load balancing has been evaluated. In this simulation, the workload is measured by the size of packets transmitted. The overall message complexity and the hotspot message complexity are identified when the load balancing overlay-based strategy is turned on (*i.e.*, the load balancing tag is 1) or off (*i.e.*, the load balancing tag is set to 0). Similarly, the same topology is used for all simulated algorithms LEACH, Reference [13], and DLBO. Communication and balancing factor simulation are two major patterns in this procedure.

Figure 3 shows the overall task execution time. As the number of nodes increases, the execution time in DLBO is decreased in terms of a change from 0.32 to 0.74. This is due to the reason that the overlay-based grid quorum has been dynamically constructed when the load balancing tracker works. With the dynamic reconfigurations, as shown in Figure 4, the balancing factor in DLBO is comparatively steady.

Figure 5 and Figure 6 compare the transmission cost of load balancing produced by the three algorithms as a result of the grid quorum on a side for a uniform load. For different data size (K) in average effective time, the experiment is executed 20 times. From Figure 5, the unbalancing algorithm produces the most unbalanced trees, while our basic algorithm is slightly better balanced on average than LEACH. In the best, our basic algorithm considerably outperforms both LEACH and node-centric method. Worst cases occur when the grid head is located near the edge or corner of grid quorum, so that both LEACH and node-centric method produce unbalanced sensor nodes. In contrast, our basic algorithm attempts to

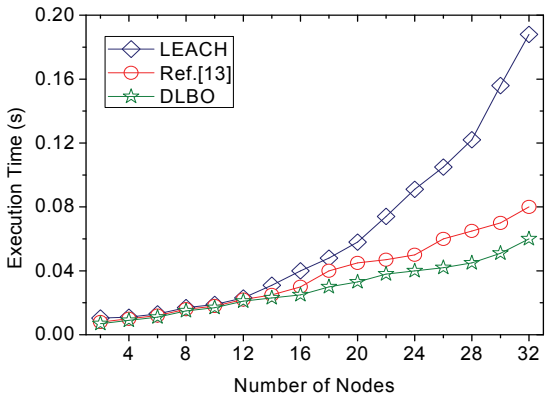

Figure 3. Execution time comparison.

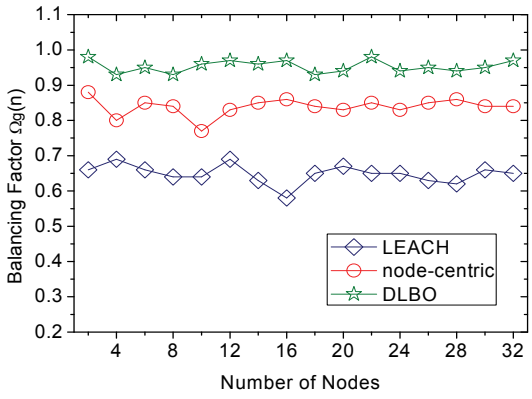

Figure 4. Balancing factor comparison.

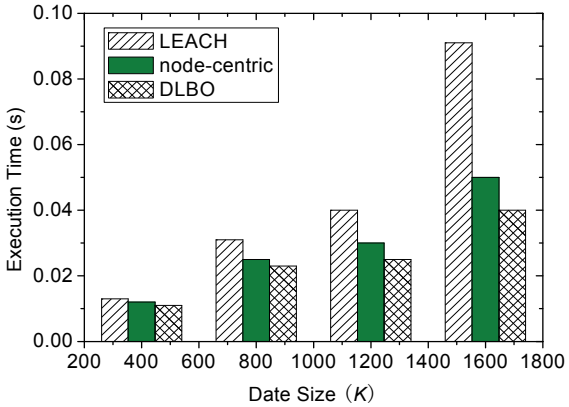

Figure 5. Transmission cost and execution time.

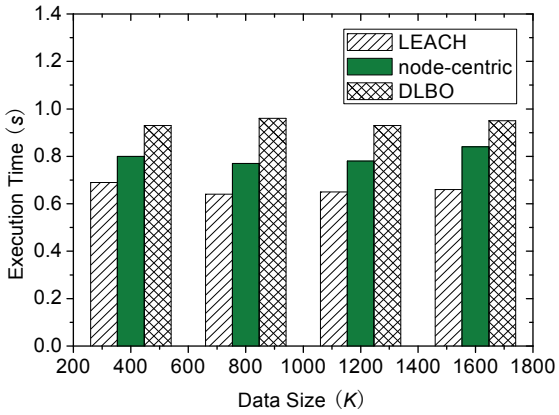

Figure 6. Transmission cost and balancing factor.

expand the lightest branches into open space, to avoid confining the growing scale of sensor nodes.

6. Conclusions

This paper addresses the load balancing in wireless sensor networks. The contributions include: 1) A dynamic nodes workload infrastructure, which is fundamental to this analysis and is believed to be a useful tool in other contexts of overlay-based WSN applications; 2) A discrete model for balancing factor, which provides a worst-case estimation for the steady data delivery between sensor nodes; 3) The optimization schemes and analysis of their reliability; and 4) simulation experiments which provide insight into the performance of dynamic load balancing from a number of major respects. In the future, we will consider how the model and the optimization schemes can be applied to wireless cognitive applications such as habitat monitoring and tracking of office equipment.

7. Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive suggestions to improve the quality and presentation of this paper. This paper is significantly extended from the conference version submitted to the IEEE CIT 2008.

8. References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," IEEE Communication Magazine, Vol. 40, No. 8, August 2002, pp. 102–114.

[2] I. Akyildiz and I. Kasimoglu, "Wireless sensor and actor networks: Research challenges," Elsevier Ad Hoc Networks, Vol. 2, No. 4, pp. 351–367, 2004.

[3] Y. Law, J. Doumen, and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks," ACM Transactions on Sensor Networks, Vol. 2, No. 1, pp. 65–93, 2006.

[4] W. Zhang, G. Cao, and T. L. Porta, "Data dissemination with ring-based index for wireless sensor networks," IEEE transactions on mobile computing, Vol. 6, No. 7, July 2007.

[5] G. Noubir and G. Lin, "Low-power DoS attacks in data wireless LANs and countermeasures," SIGMOBILE Mobile Computer Communication Review, Vol. 7, No. 3, pp. 29–30, 2003.

[6] W. Zhang, G. Cao, and T. La Porta, "Data dissemination with ring-based index for wireless sensor networks," Proceedings of IEEE International Conference on Network Protocols, pp. 305–314, November 2003.

[7] M. Khan, G. Pandurangan, and V. S. Anil Kumar, "Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks," IEEE Transactions on Parallel and Distributed Systems, Vol. 20, No. 1, pp. 124–139, January 2009.

[8] X. Li, Y. Wang, P. Wan, W. Song, and O. Frieder, "Localized low-weight graph and its applications in wireless ad hoc networks," Proceedings of IEEE INFOCOM, 2004.

[9] G. D. Fatta and M. R. Berthold, "Dynamic load balancing for the distributed mining of molecular structures," IEEE Transactions on Parallel and Distributed Systems, Vol. 17, No. 8, August 2006.

[10] S. Dhakal and M. M. Hayat, "Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach", IEEE Transactions on Parallel and Distributed Systems, Vol. 18, No. 4, April 2007.

[11] F. Guo, J. Chen, W. Li, and T. Chiueh, "Experiences in building a multihoming load balancing system", INFOCOM'04, Vol. 2, pp.1241–1251, 2004.

[12] J. Gao and L. Zhang, "Load-balanced short-path routing in wireless networks," IEEE Transaction on Parallel and Distributed Systems, Vol. 17, No. 4, pp. 377–388, April 2006.

[13] H. Dai and R. Han, "A node-centric load balancing algorithm for wireless sensor networks," Global Telecommunications Conference, IEEE GLOBECOM'03, Vol. 1, pp. 548–552, December 2003.

[14] Y. Wang, W. Peng, M. Chang, and Y. Tseng, "Exploring load-balance to dispatch mobile sensors in wireless sensor networks", Computer Communications and Networks, ICCCN'07, Proceedings of 16th International Conference, pp. 669–674, August 2007.

