Distributed Data Gathering Scheduling Protocol for Wireless Sensor Actor and Actuator Networks

Wei Shen, Tingting Zhang
Department of Information Technology and Media
Mid Sweden University, Sundsvall, Sweden
Email: {wei.shen, tingting.zhang}@miun.se

Mikael Gidlund
ABB Corporate Research
Västerås, Sweden
Email: mikael.gidlund@se.abb.com

***Abstract*—This paper presents a cross-layer distributed scheduling protocol for sensor data gathering transmission in wireless sensor actor and actuator networks. We propose the parent-dominant decision scheduling with collision free (PDDS-CF) algorithm to adapt the dynamics of links in a realistic low-power wireless network. In addition, the protocol has a light-weight mechanism to maintain the conflict links. We have evaluated the protocol and implementation in TinyOS and Telosb hardware. The experiment shows that our protocol has robustness to the topology changes and it has significant improvements to reduce the traffic load in realistic wireless networks.**

***Index Terms*—Data Gathering, Distributed Scheduling, WiSAAN, Wireless HART**

I. Introduction

Using wireless technologies within industrial automation is becoming more and more popular, especially within the process automation domain. Being able to quickly and cost effectively obtain real-time data from anywhere in the field at any time is now essential for an industrial automation system. The obvious advantage of wireless automation is the immediate savings that can be realized in installation and maintenance-wireless installation typically costs as much as 50-60 percent less than the wired alternative [1]. Typically today there are mostly wireless installations on field instrumentation level and a prime example is ABB's installation on the oil rig "Goliat" which contains more than 400 wireless vibration sensors connected to several gateways. These sensors are based on Wireless HART [2] which is the only standard for wireless sensor networks within process automation.

In these applications, the traffic load sharply increases when the network scale increases, which makes the performance of the whole network, e.g., delay, throughput and power consumption, becomes worse and worse. In such a network, there is typically a short-length sensor data with a relatively large overhead in each packet. The scheme of the data gathering transmission is therefore able to improve the QoS of the network by means of efficiently scheduling packets transmission and combining packets from multiple sources and then forwarding a new packet. Previous work evolving data gathering transmission for sensor networks can be classified into two types. The first one is a "randomly waiting gathering" which typically uses a gathering mechanism in the buffer of each node without any scheduling [3][4][5]. This method cannot guarantee the gathering performance because the outbound link is not always transmitted after the inbound link. The other type is a scheduled gathering which generates a minimum possible traffic load, i.e., the number of slots, with collision-free schedules [6][7][8].

However, there are two key issues which have not been dealt with in a satisfactory manner for the previous gathering scheduling algorithms. Firstly, most of the solutions, e.g., [6]-[8], have not considered the case involving topology changes and retransmission in realistic wireless networks. Wireless sensor and actuator network undergo frequent topology changes due to physical environmental changes, node join, node failures, and time-varying channel conditions [9]. An individual wireless link is inherently unreliable. It has observed from using real-platform testbeds with a low power wireless network (IEEE 802.15.4) that links have a wide range of packet reception ratios (PRR) which can vary significantly over time even without intra-collisions and heavy external interference [10][11]. It is even worse for industrial applications in harsh environment as both humans and machinery alter the RF environment. In addition, the case involving retransmission due to unreliable and asymmetric wireless links have to be considered when designing the protocols.

Secondly, most of the previous gathering scheduling algorithms has not considered another important problem: the scheduling configuration overhead. The previous work needs to maintain one-hop and two-hop neighbors. Each node has to construct its "competitor set" from the two neighbors and decides its schedule by itself. We call these algorithms as node-dominant decision scheduling (NDDS). They only consider the delay after scheduling. But the scheduling procedure should reduce traffic load and delay.

In this paper, we present a novel cross-layer scheduling protocol for data gathering transmission. A parent-dominant decision scheduling with collision free (PDDS-CF) algorithm is proposed to adapt the dynamics of links in a realistic low-power wireless network. In addition, the protocol reduces the scheduling traffic through maintaining distributed light-weight conflict-links tables. We have implemented the protocol in TinyOS and Telosb hardware. The results show that the protocol has robustness to the topology changes and it has significant improvements to reduce the traffic load.

The remainder of the paper is organized as follows. Section II presents the network and transmission model. We propose

the adaptive cross-layer data gathering scheduling protocol in section III. Section IV provides the performance evaluation of the protocol and implementation in real platforms. We conclude the paper in section V.

II. Network and transmission model

A. Network topology and routing protocol

The Collection Tree Protocol (CTP), which has become a de-facto standard in collection routing in CSMA based sensor networks, uses adaptive beaconing and datapath validation mechanisms [12] to form and maintain a spanning tree topology in dynamic low power wireless environment. We distill the routing discovery, routing change mechanism of the CTP routing protocol. The protocol we proposed in this paper is smoothly compatible with the CTP protocol.

B. Wireless interference model

In a wireless network, wireless links whose transmissions may interfere with each other cannot be scheduled to transmit at the same time. Wireless transmission may collide in two ways that are typically referred to as primary and secondary interference [13]. Primary interference occurs when one node has to perform more than one action in a single time slot, i.e., this node receives messages from two different transmitters, transmits a message to two different receivers, or transmits a message to a receiver and receives a message from a transmitter at the same time. Secondary interference occurs when the receiver Rx of transmitter Tx is within the radio range of another transmitter which does not intend to transmit a message to Rx.

III. Proposed distributed data gathering scheduling protocol

This section presents the distributed data gathering scheduling protocol that we have proposed, namely parent-dominant decision scheduling with collision free (PDDS-CF).

A. Overview

PDDS-CF builds and maintains a distributed scheduling for packet gathering transmission in low power and lossy networks. It is designed to adapt the inconsistencies of the topology, node join, node failures and link transmission failures. The data gathering procedure is triggered by a route found or a route changed and it proceeds iteratively from the end nodes towards the root by multi-hop routing. An end node is one whose in-degree is zero, i.e., which has no children nodes. Upon the discovery of either a found or changed route at a node, a one-shot timer called a "route timer" is initiated. The motivation behind the setting up of this timer is to await the change of route for the local node's neighbors. The reason for this is because both a new node joining the network and finding its route and a node changing its route may cause its neighbors to change the routes. When this timer is expired, the node sends its expected schedule according to its conflict table to its parent if it determines that it is an end node. After collecting all the schedules of its children, a parent node will make them schedules. If it is a non-end node, the node is not allowed to populate its schedule until all its children have valid schedules. The detailed description is discussed in the rest of the section and a schematic of the protocol is shown in Fig. 2.

During the sensor data transmission procedure, each scheduled non-end node receives packets from its children, decodes the sensor data, jointly collects the sensor data into one packet and forwards this packet to its parent. The performance of the data gathering transmission also depends on the medium access protocol (MAC), the exceptions being the scheduling protocol and the routing protocol.

The PDDS-CF supports both non-slotted carrier sense multiple access with collision avoidance (CSMA-CA) and slotted CSMA-CA, and also the time division multiple access (TDMA) based MAC, e.g., WirelessHART MAC, which is a hybrid protocol that uses slotted Aloha and TDMA with channel hopping. In the TDMA based MAC, a schedule assigned by our protocol represents a time slot number in a superframe composed of a continuation of time slots which are repeated periodically. In the slotted CSMA, a schedule is an order of a back-off time unit order, while a schedule represents an order of a global backoff logic-time unit order in non-slotted CSMA. A distributed total order algorithm is required in order to execute the scheduling result in the non-slotted CSMA due to the fact that there is no global time synchronization.

Our protocol has currently not considered the case that exceeding the maximum packet size when combining packets from multiple sources.

B. Messages and tables maintained by PDDS-CF

The PDDS-CF specifies five messages and maintains three tables as shown in Table I.

C. Collision free scheduling

As discussed in section I, collision free is one of the important goals for the data gathering scheduling problem. We classify the contenders of a node as three types: parent and children contenders, sibling contenders and hidden terminal contenders. The first-type conflict can be avoided by means of aggregating the schedules from bottom to top when populating the schedule. The sibling nodes may be one-hop neighbors or two-hop neighbors, e.g., v_1, v_2, v_3 and v_4 are sibling nodes in Fig. 1. Node v_4 and node v_5 in Fig. 1 neither have parent-children relationship nor are sibling nodes. Node v_4's parent is a neighbor of node v_5. We call node v_4 and v_5 are hidden terminal contenders each other. Previous work, e.g., [6]-[8], avoids this type of conflict through maintaining both one-hop and two-hop neighbor table. Only one-hop neighbor table is enough in our protocol. We use two tables, *HCT_Expc* and *HCT_Par*, allocated separately at one node and its parent. A node, e.g., v_5, records every *P2C_Sched* message from the parent of its hidden terminal contenders (e.g., v_1-v_4) in *HCT_Expc*. Each entry of *HCT_Expc* contains the address of the hidden terminal contender's parent, a schedules set and the size of the schedules set. The parent of each node (e.g., v_4's

TABLE I
MESSAGES AND TABLES IN THE PDDS-CF PROTOCOL

Messages /Tables	Description
C2P_New	Unicast message from a child to its parent to inform the parent of its presence.
C2P_Del	Unicast message from a child to its parent to notify its parent to delete this child.
C2P_Expc	Unicast message from a child to its parent to send its expected schedule and the underlying conflict schedules.
P2C_Sched	Broadcast message from a parent after making schedules for its children. This message contains child-schedule pairs.
C2P_Conf	Unicast message from a child to its parent to confirm or not to confirm the schedule assigned by the parent.
ChildrenTbl	A table for each entry which contains a child id, a schedule, a "valid" bit, an "expected" bit, a "scheduled" bit and a "confirm" bit.
HCT_Expc	A table is used to avoid a conflict from hidden terminal contenders when calculating an expected schedule.
HCT_Par	A table is used to avoid a conflict from hidden terminal contenders when populating schedules for children.

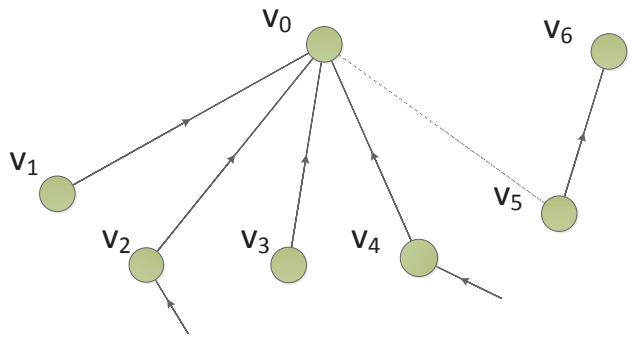

Fig. 1. An example of the topology.

parent v_0) records every *C2P_Conf* message it snooped from its hidden terminal contender (e.g., v_4's contender v_5) in the *HCT_Par* table. An entry of *HCT_Par* contains the address of the hidden terminal contender and a corresponding schedule. In our protocol, the schedule of a node is decided by its parent and this node. Traffic can be significantly reduced using this method because it is not necessary to maintain the two-hop neighbor table.

D. Parent-dominant decision scheduling

We discuss the skeleton of the PDDS-CF. Fig. 2 shows the steps that a node passes through in the case of scheduling as a finite state automation. After a new node joins and finds its route or an existing node changes its route from the routing layer, it initiates a one-shot "route timer" as discussed in subsection A. In addition, the node sends a unicast *C2P_New* message to inform its current parent of its presence if it finds its route. When its route changes, it sends a *C2P_Del* to its old parent and sends a *C2P_New* to its new parent. During the period awaiting the expiration of the "route timer", the node may receive either a *C2P_New* or a *C2P_Del*. When the timer is expired, the node is an end node if it does not find any valid children in its *ChildrenTbl*. After calculating an expected schedule and a set of underlying conflict schedules it sends them to its parent via the *C2P_New*. The underlying-conflict-schedules set contains those schedules that belong to the *HCT_Expc* and are bigger than the expected schedule. **Algorithm 1** shows the pseudo code that describes in more detail the calculation of the expected schedule.

There is a possibility that there are valid children, but, all of them are scheduled when the timer is expired. The reason may be that the scheduled children do not need to change their routes after the node changes its route. In this case, the node also calculates an expected schedule and a set of underlying conflict schedules, and sends them to the parent. Then the node waits for the *P2C_Sched* message to receive its schedule which has been assigned by its parent. After receiving the *P2C_Sched* message, the node looks up the *HCT_Expc* table to check whether the schedule is a conflict one with the hidden contenders in the table. If there is a conflict, the node sends a negative confirmed *C2P_Conf* message with a new expected schedule to its parent. Otherwise, it sends a confirmed *C2P_Conf* message.

There is a third possibility when the "route timer" is expired, namely that there are valid children which are not scheduled in the *ChildrenTbl*. In this case, the node has to wait for

Algorithm 1 Calculating an expected schedule

1: Initialize a temporary schedule $\tau \leftarrow 0$.
2: **if** this node is NOT an end node **then**
3: $\tau \leftarrow$ MAX(Children's schedules).
4: $\tau \leftarrow \tau + 1$
5: **while** $\exists$ $entry \in HCT\_Expc$ and $\exists$ $e \in entry.schedules\_set$, satisfying $e = \tau$ **do**
6: $\tau \leftarrow \tau + 1$
7: $Expected\_schedule \leftarrow \tau$.

Algorithm 2 Populating schedules for children

1: Initialize $SiblingSchedulesTable \leftarrow$ empty entries.
2: **for** each entry $\in ChildrenTbl$ **do**
3: **if** $entry.flags$ & VALID_BIT = true and $entry.flags$ & EXPECTED_BIT = true **then**
4: $\tau \leftarrow entry.schedule$.
5: **while** $\exists$ $s \in SiblingScheduleTable$, satisfying $s = \tau$ or $\exists$ $e \in HCT\_Par$, satisfying $e.schedule = \tau$ or $\exists$ $c \in entry.underlyingConflictSchedules$, satisfying $c = \tau$ **do**
6: $\tau \leftarrow \tau + 1$
7: $entry.schedule \leftarrow \tau$
8: Add $entry.schedule$ into $SiblingSchedulesTable$
9: $entry.flags \mid =$ SCHEDULED_BIT

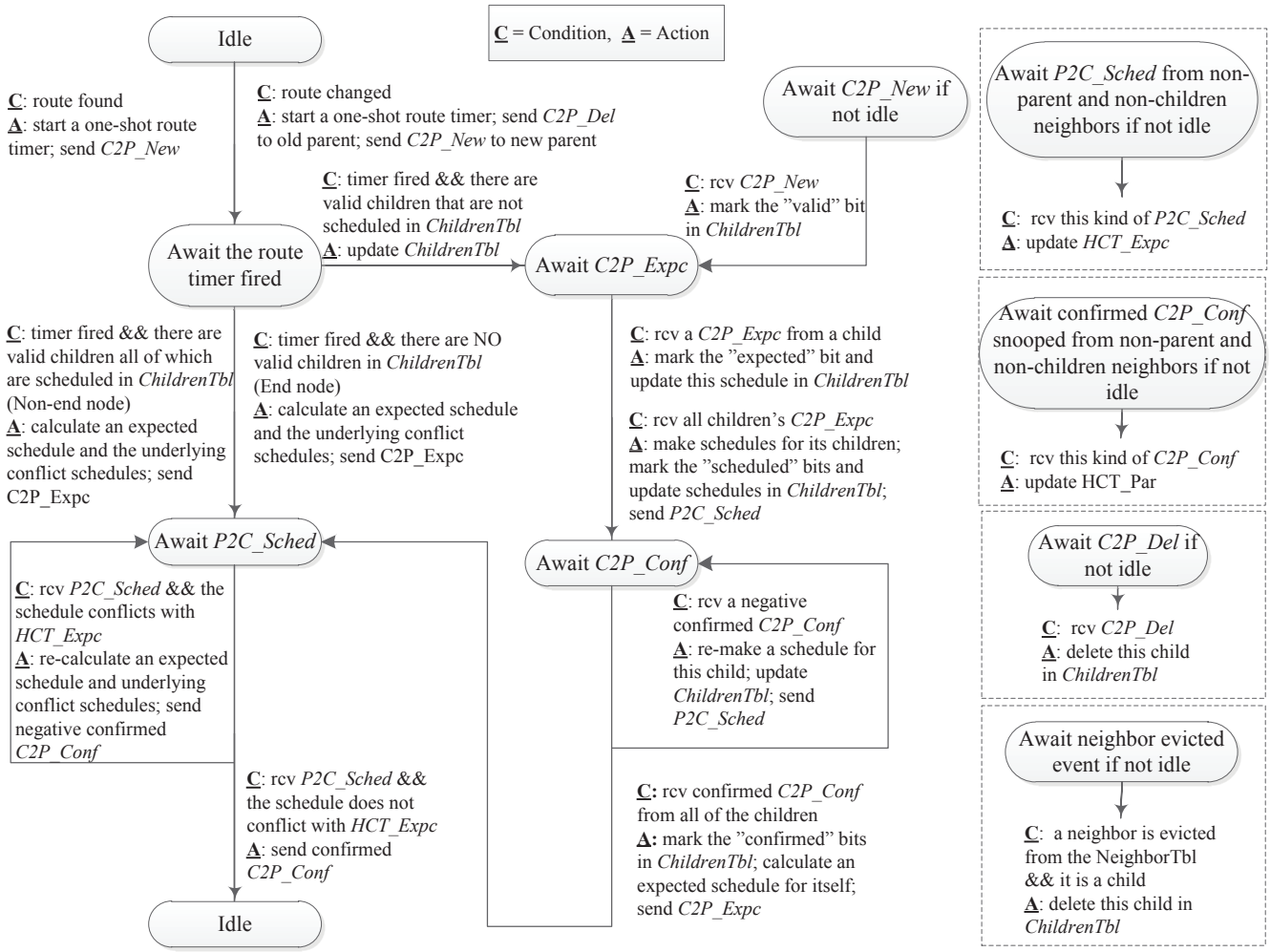

Fig. 2. Schematic of the PDDS-CF protocol.

the *C2P_Expc* message from the unscheduled children. After receiving the *C2P_Expc* messages from all its children, the node populates schedules for its children. The pseudo code is shown in **Algorithm 2**. Then the node broadcasts a *P2C_Sched* message. Upon receiving a negative confirmed *C2P_Conf* message, the node has to calculate a schedule for this child and sends *P2C_Sched* message again until it receives the confirmed *C2P_Conf* messages from all its children. Then it starts to calculate its own expected schedule and a set of underlying conflict schedules, and sends a *C2P_Expc* message to its parent. After that, the node starts to wait for the *P2C_Sched* message and follows the steps described in the previous paragraph.

The reason for using the *C2P_Conf* message for a node to confirm the schedule instead of the reception of the schedule assigned by its parent is explained as follows. After a node sends its expected schedule to its parent, the parent does not immediately assign a schedule for the node until it has received the expected schedules from all its children. During the waiting time, the node may receive a new underlying conflict schedule in the *HCT_Expc* table which has the possibility to conflict with the schedule assigned by the parent.

E. Robustness to frequent topology inconsistencies

The authors in paper [6] have discussed how to detect the node join and failure for maintenance. They assume that the wireless sensor network is stationary if there are no nodes joining and failing. However, wireless sensor and actuator network undergo frequent topology changes due to physical environmental changes, node join, node failures, and time-varying channel conditions. It is even worse for industrial applications in harsh environment as both humans and machinery alter the RF environment. Besides, the case involving retransmission due to unreliable and asymmetric wireless links have to be considered when designing the protocols.

The PDDS-CF protocol proposed in this paper is smoothly compatible with the CTP protocol which is a de-facto routing protocol in wireless sensor network and has been tested in many hardware platforms and implemented in TinyOS, Contiki and other operating systems.

The link estimator of the CTP protocol measures the bidirectional characteristics of links through calculating the

reception probabilities of broadcasting and unicast packets. In order to adapt the frequent topology changes and transmission failures, we distill the mechanism of this link estimator. One node pending a schedule from its parent detects the gap of sequence numbers suffixed with the beacon and the *P2C_Sched* and reports this to the link estimator for populating the reception probabilities. If the link towards to a parent has a bad quality, the route change would be triggered and a scheduling combination would be launched. A success or failure of acknowledgment of each unicast message is recorded for populating the reception probabilities of unicast packets. The four unicast messages, *C2P_New*, *C2P_Del*, *C2P_Expc* and *C2P_Conf*, in our protocol stop retransmission when exceeding the maximum retransmission numbers, which introduces the route change and then initiate a new scheduling combination. Both a new node joining and an existing node failing are easily to solve because they directly cause the route changes and then the scheduling combination engine is triggered.

Furthermore, a node always awaits *C2P_New*, *P2C_Sched*, *C2P_Del* and the neighbor evicted event to update the *ChildrenTbl*, *HCT_Par* and *HCT_Expc* if it is not idle, as shown in the right side of Fig. 2.

S(h) / id \ t	t_1	t_2	t_3	t_4	t_5	S(h) / id \ t	t_1	t_2	t_3	t_4	t_5
0x01	1(1)	6(1)	6(1)	6(1)	6(1)	0x0d	-	-	2(2)	2(2)	2(3)
0x02	3(1)	8(1)	8(1)	8(1)	7(2)	0x0e	-	-	4(1)	4(1)	4(2)
0x03	2(1)	7(1)	7(3)	7(3)	8(1)	0x0f	-	-	1(2)	1(2)	1(3)
0x04	-	9(1)	9(2)	9(2)	9(2)	0x10	-	-	-	-	3(3)
0x05	-	1(2)	1(2)	1(2)	1(4)	0x11	-	-	-	-	6(4)
0x06	-	2(2)	3(3)	3(3)	3(3)	0x12	-	-	-	-	5(4)
0x07	-	5(2)	6(3)	6(3)	6(3)	0x13	-	-	-	-	2(4)
0x08	-	4(2)	5(3)	5(3)	5(4)	0x14	-	-	-	-	4(4)
0x09	-	3(2)	4(3)	4(3)	4(3)	0x15	-	-	-	-	1(3)
0x0a	-	1(3)	1(4)	1(4)	1(4)	0x16	-	-	-	-	1(3)
0x0b	-	1(3)	1(2)	1(2)	1(3)	0x17	-	-	-	2(3)	2(4)
0x0c	-	-	3(2)	3(2)	3(3)	0x00	root node				

Fig. 3. "-"means absence of this node at this time; $t_1 \approx$ 2.7min, $t_2 \approx$ 14.0min, $t_3 \approx$ 49.6min, $t_4 \approx$ 86.8min and $t_5 \approx$ 124.3min; "id"is the address of nodes; "S"is the schedule; "h"in the brackets represents the hop count.

IV. Performance Evaluation

This section evaluates the performance of the PDDS-CF protocol and the implementation.

A. Experimental Methodology

We have implemented and evaluated the PDDS-CF protocol in TinyOS and Telosb hardware platform. TinyOS uses CSMA-CA as its default MAC layer. Our protocol is running over this MAC layer and CTP routing layer and calculates a schedule for each node in a dynamic topology. The schedule of each node can be used in a hybrid CSMA and TDMA MAC as we discussed in section III-A. A schedule represents an order of a slot occupied by a node in a TDMA superframe. A superframe is a collection of consecutive time slots.

The experiment has been carried out in an office environment in Mid Sweden University. We have deployed 24 Telosb motes on 2nd, 3rd, 4th and 5th floors in the building. We have collected the schedules of each node every 8 seconds through a single hop or multiple hops.

B. Experimental Results

In order to evaluate the robustness to the topology changes, we added the number of nodes at random time during the 10-hours measurement. Fig. 3 shows the schedules of nodes at five time points. The protocol reflects changes in the nodes joining and other topology changes in a few packet times. A new joining node typically chooses a schedule as small as possible and cause its ancestors to increase their schedules.

We assume that one time slot is the time length of a transaction from a node to its parent. The traffic load is defined as the number of slots required for each node in the network to transmit an end-to-end packet to the root. We roughly calculate the average traffic load using the real-time measured schedule of each node and the topology information. The average traffic load is calculated over these time intervals, T_1, T_2, T_3, T_4, T_5, where T_i represents the time interval between $t_i - 1$ and t_i and $i = 1, 2, 3, 4, 5$. t_0 is the time when the first non-root node joining the network. During these intervals, the number of nodes in the network does not change but the topology may change.

In order to evaluate the performance of our algorithm, we have implemented a centralized scheduling algorithm, the level based scheduling algorithm [14] which is a heuristic algorithm to reduce the slots number (the traffic load) without data gathering transmission. We use the collected topology information by the root as an input of this level-based algorithm and calculate the average traffic load over the same time intervals. Since the centralized scheduling algorithm has the whole topology information, there is hardly possibility for a distributed algorithm to have less traffic load under the same topology.

The comparison of our protocol and the level-based algorithm is shown in Fig. 4. There are 15 and 16 nodes in the interval T_3 and T_4, while the traffic load are 15 and 16 respectively. Both the two values of the traffic load are the optimal values for the scheduling algorithm without data gathering transmission because they are the maximum load of the root node. The PDDS-CF which carries out the adaptive data gathering scheduling algorithm performs better than these optimal ones. There is a delay when the PDDS-CF reflects the topology change which may cause the outbound link of a node is scheduled before its inbound link. In this case, the traffic load is at least more than the maximum schedule of the network. This is the reason that the traffic load of PDDS-CF during T_4 has not been reduced much. The amount of the performance improvement of our protocol increases as the

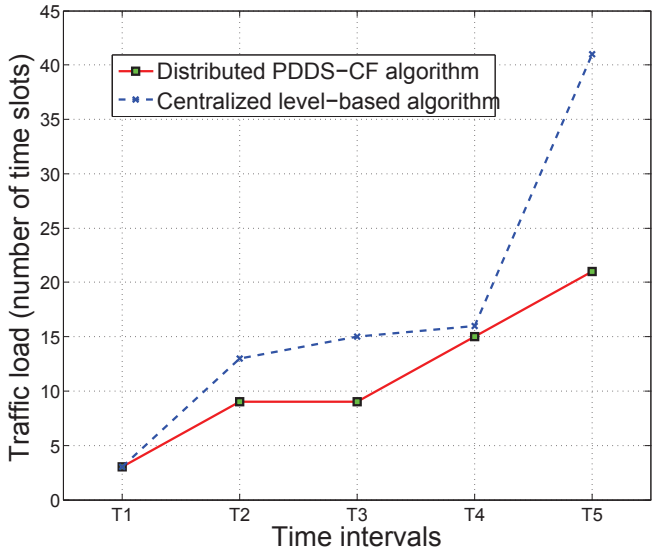

Fig. 4. The average traffic load of PDDS-CF is much lower than the centralized level-based algorithm when the topology scale increases; T_1, T_2, T_3, T_4 and T_5 are time intervals during which the average traffic load is calculated; the number of nodes during these interval are 3, 12, 15, 16 and 23 respectively (not including the root node).

number of nodes increases. The average traffic load of the level-based algorithm during T_5 is 41, while PDDS-CF only has an average traffic load of 21. In summary, our protocol has significant improvements to reduce the traffic load in realistic wireless networks because it outperforms the centralized level-based scheduling algorithm.

V. Conclusion

The motivation behind this paper is a desire to enable data gathering scheduling algorithms to adapt the dynamics of links in a low power wireless sensor and actuator network and to significantly reduce the traffic load in such a network. The PDDS-CF protocol we proposed targets this problem in practical industrial applications, such as monitoring and supervision of industrial automation in a large-scale network.

We have evaluated the protocol and implementation in TinyOS and Telosb hardware. The experiment shows that our protocol has robustness to the topology changes. The protocol outperforms the centralized level-based scheduling algorithm and therefore it has significant improvements to reduce the traffic load in realistic wireless networks. In the future, we will implement the protocol over multiple MAC protocols, further evaluate and improve the scalability and the scheduling delay.

Acknowledgment

We would like to thank Luyuan Zou who helped us to set up the experiments. This work has been supported by Swedish Knowledge Foundation (KK-stiftelsen) and Sensible Things That Communicate (STC) research program at Mid Sweden University.

References

[1] J. Akerberg, M. Gidlund, and M. Bjorkman, "Future research challenges in wireless sensor and actuator networks targeting industrial automation," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, july 2011, pp. 410 –415.

[2] Hart specification. [Online]. Available: http://www.hartcomm.org

[3] D. Kliazovich and F. Granelli, "Packet concatenation at the ip level for performance enhancement in wireless local area networks," *Wirel. Netw.*, vol. 14, pp. 519–529, August 2008.

[4] H. Zhai and Y. Fang, "A distributed packet concatenation scheme for sensor and ad hoc networks," in *Military Communications Conference, 2005. MILCOM 2005. IEEE*, oct. 2005, pp. 1443 –1449 Vol. 3.

[5] J. Neander, T. Lennvall, and M. Gidlund, "Prolonging wireless hart network lifetime using packet aggregation," in *Industrial Electronics (ISIE), 2011 IEEE International Symposium on*, june 2011, pp. 1230 –1236.

[6] B. Yu, J. Li, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," in *INFOCOM 2009, IEEE*, april 2009, pp. 2159 –2167.

[7] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *Mobile Ad-hoc and Sensor Networks*, ser. Lecture Notes in Computer Science, X. Jia, J. Wu, and Y. He, Eds. Springer Berlin / Heidelberg, 2005, vol. 3794, pp. 133–142.

[8] S.-H. Huang, P.-J. Wan, C. Vu, Y. Li, and F. Yao, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, may 2007, pp. 366 –372.

[9] I. Rhee, A. Warrier, M. Aia, J. Min, and M. Sichitiu, "Z-mac: A hybrid mac for wireless sensor networks," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 3, pp. 511 –524, june 2008.

[10] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: measuring wireless link burstiness," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008, pp. 29–42.

[11] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "An empirical study of low-power wireless," *ACM Trans. Sen. Netw.*, vol. 6, pp. 16:1–16:49, March 2010.

[12] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, 2009, pp. 1–14.

[13] S. Ramanathan and E. Lloyd, "Scheduling algorithms for multihop radio networks," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 2, pp. 166 –177, apr 1993.

[14] S. C. Ergen and P. Varaiya, "Tdma scheduling algorithms for wireless sensor networks," *Wirel. Netw.*, vol. 16, pp. 985–997, May 2010.