Distributed convergecast scheduling for reduced interference in wireless sensor and actuator networks

Omiyi, Peter; Bür, Kaan; Yang, Yang

Published in:
Proceedings of WCNC – IEEE Wireless Communications and Networking Conference

DOI:
10.1109/WCNC.2010.5506170

2010

Document Version:
Peer reviewed version (aka post-print)

Distributed Convergecast Scheduling for Reduced Interference in Wireless Sensor and Actuator Networks

P. E. Omiyi, K. Bür, and Y. Yang*

Dept of Electronic and Electrical Engineering
University College London (UCL)
Torrington Place, London WC1E 7JE, UK
{e.omiyi, k.bur, y.yang}@ee.ucl.ac.uk

***Abstract*—Emerging wireless sensor and actuator network (WSAN) technology has the potential to enable semi-autonomous air-flow control to improve the aerodynamic performance of aircraft. In this paper, a WSAN topology comprising of multiple linear sensor clusters terminated by actuators is proposed for active flow control. Two interference aware convergecast scheduling strategies are presented and analyzed, with the objective of jointly minimizing latency and energy consumption. The proposed schemes are required to coordinate local convergecast communications in each cluster, on the one hand, and to arbitrate between mutually interfering nodes of different clusters contending for the wireless channel, on the other. Initial results show that a 19s and 11s convergecast delay moderate to high energy consumption can be achieved.**

***Index Terms*—Wireless sensor and actuator networks, convergecast, scheduling, interference.**

I. Introduction

A wireless sensor and actuator network (WSAN) provides a means of semi-autonomous, distributed monitoring and supervision/control of remote processes [1, 2]. A WSAN comprises of a system of sensors and actuator nodes that are distributed over the environment or physical system of interest and interconnected by wireless links. Sensors gather local information about the system and transmit the collected data to actuators in the vicinity of the measured parameter through single-hop or multi-hop communications. Using the received information, the actuators perform actions to control and/or supervise the behavior of the environmental or physical processes. This inherent capability of WSANs to interact with and influence the physical world differentiate them from common wireless sensor networks (WSNs). Application areas of WSANs include disaster relief operations, autonomous search and rescue operations, traffic control/enforcement and target identification/tagging in military missions.

Convergecast is a typical communication pattern in WSN and WSAN applications. In WSNs, many or all sensors send their measured data towards a single data sink. Given the periodic nature of the generated traffic, distributed time-division-multiple-access (TDMA) based algorithms can be used to provide convergecast scheduling, with the objective of avoiding packet collisions and minimizing convergecast latency. Radial coordination proposed in [3] is a distributed TDMA approach that addresses the problem of packet loss due to congestion and collision near the sink. Specifically, radial coordination requires the sensors to adjust their transmission times according to a quadratic formula based on the estimated hop distance to the sink. It is extended with packet aggregation and duplication in [4] to tackle the bandwidth bottleneck problem experienced by the sink. An inward delay mechanism is also introduced in [4] to reduce the waiting time for far-away sensor nodes. Several distributed convergecast scheduling algorithms are proposed in [5, 6] to minimize the total transmission time of a convergecast session, wherein each sensor node has exactly one packet to send to the data sink, in linear, multi-line, tree and general WSNs.

As a new application of WSANs, this paper aims to investigate the capability of using a WSAN for active airflow control over aircraft wings. Different from the traditional WSN convergecast problem with a single data sink, the WSAN for the proposed application has many data sinks, i.e. actuators. In this case, sensors are organized into multiple clusters and, within each cluster, sensors send their data to its local actuator using convergecast. Convergecast scheduling for such a WSAN needs to satisfy the conflicting demands of jointly minimizing inter-cluster interference and total convergecast delay across all clusters. Furthermore, the proposed airflow control application imposes stringent bounds on energy-efficiency, convergecast latency and packet delivery rate.

Two distributed TDMA-based convergecast scheduling strategies for WSANs are investigated and compared in this paper. Specifically, Section II presents the system model of a WSAN for active airflow control and defines the problem. In Section III, two WSAN convergecast scheduling strategies are analyzed mathematically to derive closed-form expressions of convergecast latency and signal-to-noise and interference ratio (SINR). Analytical results are given and discussed in Section IV, followed by the conclusions in Section V.

II. System Model and Problem Definition

In aircrafts, parasitic (pressure) drag and stall occur because of boundary layer separation [7] on the wings due to high angles of attack in take-off/landing, sudden pilot manoeuvres, or from turbulence and wind gusts. It also results from the formation of normal shock waves on the wing at transonic speeds. Several active methods for controlling boundary layer separation have been explored in the literature [7]. The typical approach is to deploy rows of airflow control actuators at strategic locations (expressed as a percentage of the airfoil chord length) on the airfoil and to operate these actuators continuously as an open loop control system. We propose to use a WSAN to achieve efficient closed-loop airflow control, wherein wireless sensors are deployed on the wings to provide real-time airflow information and decision metrics to the local actuators.

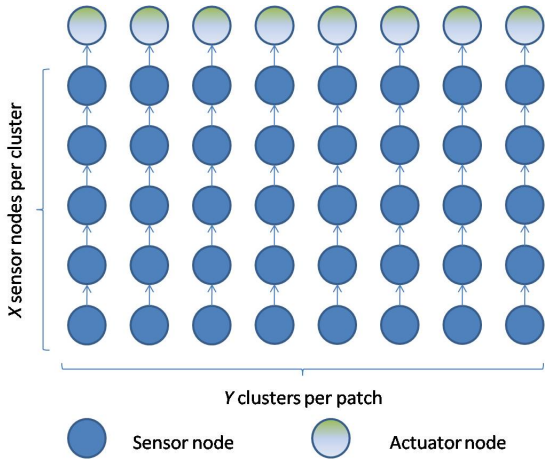

Fig. 1. WSAN patch with a grid topology

As shown in Fig. 1, we consider a stationary WSAN wherein the sensor and actuator nodes are arranged uniformly over the application environment. We assume that sensors are grouped into 'linear' clusters, with each comprising of X nodes uniformly spaced along a straight line, with a separation distance d_1 between nodes. The terminating (or sink) node of each linear cluster is now an actuator node, not a sensor node as in [5, 6]. Each sensor node has an index x according to its hop-count from the actuator node, such that the sensor node that is farthest away from the actuator of its cluster is the 1^{st} sensor node of the cluster, while the sensor node that is one-hop away is the X^{th} sensor node. All nodes with higher hop-counts than any given node in the same cluster are referred to as 'upstream' nodes relative to that node, while nodes with lower hop-counts are referred to as 'downstream' nodes.

Clusters are grouped into 'rectangular' patches, with each patch comprising of Y parallel linear clusters, where the clusters are aligned in such manner that the x^{th} sensor node in all clusters of a patch are aligned and the terminating actuators are also aligned. From the proposed application perspective, the patch represents the minimum surface area over which airflow must be monitored and controlled. The distance between the x^{th} sensor node in two adjacent clusters is the same as the one-hop distance in the cluster d_1. Each linear cluster has an index y, where clusters are indexed in increasing order from one end of the patch to the other. The patches are distributed over aircraft wings, thus forming a grid-like WSAN topology. Each node is equipped with a single omnidirectional transceiver to transmit or receive its data packet (fixed-size) via a common frequency channel. The duration of a timeslot is equal to the transmission time of one data packet. It is assumed that every node is aware of its hop-count and position (i.e. x and y indexes) through an initialization phase and system updates.

All sensors, other than the 1^{st} sensor, have 3 states, namely receive ($\mathcal{R}$), transmit ($\mathcal{T}$), and idle ($\mathcal{I}$) states, where a sensor node in the idle state may or may not have data to send and/or forward. The 1^{st} sensor node only has the transmit and idle states, while the actuator node has two states; the idle and receive states. In any given linear cluster, if the x^{th} sensor node is in the $\mathcal{R}$ state, then the $(x-1)^{\text{th}}$ sensor must be in the $\mathcal{T}$ state. The state of the all the downstream nodes relative to the x^{th} sensor , with indexes $(x+i)$ for $1 \leq i \leq X-x$, depend on the minimum interference separation required by the x^{th} node that is in the $\mathcal{R}$ state.

In general, a minimum h-hop (hd_1) distance separation is required between a node in $\mathcal{R}$ state and an interfering transmitting node in the same cluster or neighboring cluster, where h is referred to as the separation coefficient. This implies that if the x^{th} node of a linear cluster is in the $\mathcal{R}$ state, the next downstream node in the cluster that can be in the $\mathcal{T}$ state is the $(x+h)^{\text{th}}$ node, with all $h-1$ intermediate nodes being in the $\mathcal{I}$ state even if they have data to transmit. This separation is required to ensure that the received interference power at the x^{th} node from the $(x+h)^{\text{th}}$ node is sufficiently attenuated relative to the received signal power from the $(x-1)^{\text{th}}$ node, in order to ensure reliable sensor data communication. Similarly, for every node in the $\mathcal{R}$ state in every cluster, all nodes in its neighboring clusters that are on the border or outside a radius hd_1 of the node in the $\mathcal{R}$ state *may or may not be* in the $\mathcal{T}$ state, while all nodes within this radius *must be* in the $\mathcal{I}$ state even if they have data to send.

On the other hand, in order to minimize convergecast delay, it is required to maximize the number of nodes sending data in the same time-slot by minimizing h. However, the minimum value for h is 2, because when $h=1$ the one hop distance separation between the x^{th} node (in the $\mathcal{R}$ state) and its one-hop neighbors in the same cluster (the $(x+1)^{\text{th}}$ node) or in adjacent clusters (the x^{th} nodes in clusters $y+1$ and $y-1$) is the same as that between the x^{th} and the $(x-1)^{\text{th}}$ node from which it is receiving data. In this case, the received interference at the x^{th} node from each one of its one hop interfering neighbors would be equivalent to the received data signal power. Such a high level of interference results in a low sensor data rate (assuming error correction coding or retransmissions for erroneously received packets) and a higher transmission energy per data packet.

Thus, values of h that are too small, rather than reducing convergecast latency, increase the delay by increasing packet transmission time and result in poor energy efficiency. Therefore, the goal is define an interference-aware distributed schedule that minimizes total convergecast delay and energy consumption across all clusters, by operating at an optimal value of the separation coefficient h that achieves an optimal trade-off between packet transmission time and sensor channel access delay.

III. Distributed Convergecast Scheduling

Given that in the investigated WSAN application sensor data generation is periodic, sensor data is generated within the same short time-frame, and that nodes are aware of their hop-count x and one-hop neighborhood, two interference-aware, distributed TDMA based convergecast scheduling policies are proposed and analyzed, namely parallel line scheduling (PLS) and serial line scheduling (SLS). For the subsequent analysis, let $D_{\mathrm{P}y}$ and $D_{\mathrm{S}y}$ denote the convergecast delays of the y^{th} linear cluster when using PLS and SLS, respectively. The convergecast delay is defined as the time required for a linear cluster to send its data to its actuator measured from the time the cluster begins sending its data, and let T denote the packet transmission time.

A. Parallel Line Scheduling

The goal of parallel line scheduling (PLS) is to maximize the number of linear clusters communicating in parallel. Only one sensor is transmitting at any one time per cluster, with each sensor sending in a single burst both its data and that of all the upstream nodes in the same linear cluster to its next hop downstream neighbor. In order to maintain the required separation (hd_1) between nodes in $\mathcal{R}$ state and adjacent interfering nodes in $\mathcal{T}$ state, adjacent linear clusters do not begin communication simultaneously but in a staggered fashion. Each cluster waits for the preceding adjacent cluster to communicate its first $h+1$ hops before beginning its first hop, which ensures the required minimum separation.

Fig. 2 shows a snap-shot of PLS operation with 4 linear clusters, each of which is terminated by an actuator . In this snap-shot, Cluster 3 is in the process of its first hop, with its first node from the actuator in $\mathcal{T}$ state and its second node in the $\mathcal{R}$ state. Cluster 2 began sending its data $h+1$ hops before Cluster 3. Thus, its first $h+1$ nodes are thus in the $\mathcal{I}$ state, while its $(h+2)^{\mathrm{th}}$ node is in the $\mathcal{T}$ state, which provides the necessary hd_1 protection separation for the second node of Cluster 3 that is in the $\mathcal{R}$ state. Similarly Cluster 1 began $h+1$ hops before Cluster 2, and its only node in the $\mathcal{T}$ state is separated by a distance of hd_1 from the $(h+3)^{\mathrm{th}}$ node of Cluster 2. Cluster 4 is still waiting for Cluster 3 to send its data $h+1$ hops before beginning to send its own data.

In PLS, each sensor sends its data together with the data of all its upstream nodes in a single burst to its next hop neighbor. Therefore,

$$D_{\mathrm{P}y} = \sum_{x=1}^{X} xT = 0.5X(X+1)T, \qquad (1)$$

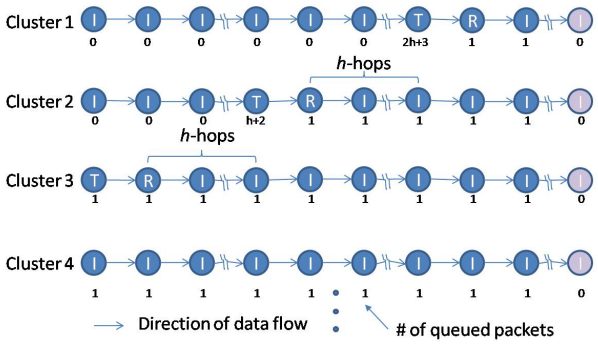

Fig. 2. Illustration of parallel line scheduling

The total convergecast delay for the entire patch is the time for all sensors of the patch clusters to complete sending their data to their respective actuators. Given the staggered manner with which clusters of a patch begin sending their data when using PLS, the total convergecast delay is measured from the time the first cluster of the patch begins sending its to the time the last cluster completes sending its data to its actuator. Each cluster waits for the preceding adjacent cluster to communicate its first $h+1$ hops before beginning its first hop. Therefore, the total delay before the last cluster (the Y^{th} cluster) of the patch begins its first hop is $(Y-1)(h+1)T$.

Given that the convergecast delay $D_{\mathrm{P}Y}$ of the last cluster is given by (1), then the total convergecast delay D_{P} of the patch using PLS is given as follows

$$\begin{aligned} D_{\mathrm{P}} &= D_{\mathrm{P}Y} + (Y-1)(h+1)T \\ &= 0.5X(X+1)T + (Y-1)(h+1)T. \end{aligned} \qquad (2)$$

B. Serial Line Scheduling

In contrast to PLS in which only one node per cluster is in $\mathcal{T}$ state in any time-slot, serial line scheduling (SLS) attempts to maximize the speed of per-cluster sensor data delivery by maximizing the number of nodes per cluster that are simultaneously in the $\mathcal{T}$ state. A minimum h-hop separation of idle nodes exists between a node in the $\mathcal{R}$ state and an interfering node of the same cluster in the $\mathcal{T}$ state. In this case, in order to maintain the required separation (hd_1) between transmitting and receiving nodes in adjacent clusters, there are $(h-1)$ idle clusters (with all nodes in the $\mathcal{I}$ state) between any two active clusters that are sending data. Each idle cluster waits for the preceding adjacent cluster to send all its sensor data to its terminating actuator before beginning to send its own data. Fig. 3 shows a snap-shot of SLS operation.

In SLS, when the number X of sensors in a cluster is less than or equal to $h+1$, the maximum possible interference separation between a receiving node and an interfering transmitter, $h-1$, is less than the minimum separation distance hd_1. Therefore, only one node per cluster can transmit in the same time slot. In this case, each sensor node sends its data together with the data of all its upstream nodes together in a single burst to its next hop neighbor. This implies that the convergecast delay $D_{\mathrm{S}y}$ of the y^{th} linear cluster is derived

similarly to $D_{\mathrm{P}y}$ in (1) as

$$D_{\mathrm{S}y} = \sum_{x=1}^{X} xT = 0.5X(X+1)T, \text{ for } X \leq h+1. \quad (3)$$

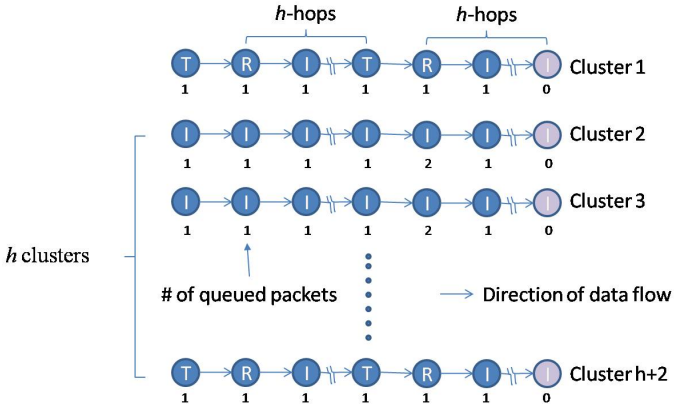

Fig. 3. Illustration of serial line scheduling

When $X > h+1$, multiple sensors can transmit in the same time slot on the same linear cluster, as it is possible to maintain the minimum separation of h-hops between a node in $\mathcal{R}$ state and an interfering node in $\mathcal{T}$ state of the same cluster. Only the first $h+1$ sensors forward their data together with that of all their upstream nodes in a single burst. This because by the time the data of all sensors with indexes less than $h+2$ arrive at sensors with indexes $h+2$ and higher, the latter have already sent their own data downstream. Thus, the convergecast delay of the cluster is compromised of two components. The first is the delay δ_{h+1} to forward all the data of the first $h+1$ sensor nodes to the $(h+2)^{\text{th}}$ sensor that has already sent its own data, which is given as follows

$$\delta_{h+1} = \sum_{x=1}^{h+1} xT = 0.5(h+1)(h+2)T. \quad (4)$$

The second component of the convergecast delay is the delay δ_{h+2} to forward all the data of the first $h+1$ sensors from the $(h+2)^{\text{th}}$ sensor to the actuator. This data comprises of a fixed length burst of $(h+1)$ packets to be forwarded over $X-h-1$ hops to the actuator. This delay is given as $(X-h-1)(h+1)T$. Therefore, the convergecast delay $D_{\mathrm{S}y}$ of the y^{th} linear cluster when using SLS is

$$\begin{aligned} D_{\mathrm{S}y} &= 0.5(h+1)(h+2)T + (X-h-1)(h+1)T \\ &= (X-0.5h)(h+1)T, \text{ for } X > h+1. \end{aligned} \quad (5)$$

Given that at any time there are $(h-1)$ idle clusters between any two active clusters that are sending data, a cluster in a patch must wait for at most $(h-1)$ preceding adjacent clusters to send all their data to their respective actuators before beginning its first hop. If the number of clusters Y in a patch is less than h, such that there can be only one simultaneously active cluster in a patch, the last cluster to send its data must wait for the $(Y-1)$ preceding adjacent clusters to send all their data before beginning its first hop. Therefore, the total convergecast delay of the patch using PLS is given as

$$D_{\mathrm{P}} = \min(Y,h) D_{\mathrm{S}y}. \quad (6)$$

C. Interference

Reliable communication of sensor data requires that the sensor data transmission rate is less than the one-hop channel capacity, which is limited by the anticipated interference levels from neighboring transmissions. In the worst-case, a node x of the y^{th} cluster in the $\mathcal{R}$ state is surrounded by three dominant interference sources, each of which is a transmitting sensor at the minimum separation distance (hd_1). The interference sources consists of a downstream neighbor on the same cluster, and the x^{th} sensors on adjacent clusters on either side of the y^{th} cluster.

The interference from any one of the interferers is a function of the interferer separation factor h. Specifically, the interference received from a single interferer at the minimum separation distance (hd_1) is given by

$$I = cS_{\mathrm{T}} h^{-\gamma} d_1^{-\gamma}, \quad (7)$$

where c is the channel gain at unit distance and γ is the channel gain coefficient. Therefore, the worst-case signal-to-noise and interference ratio (SINR) budget is

$$\begin{aligned} \lambda &= \frac{cS_{\mathrm{T}} d_1^{-\gamma}}{3cS_{\mathrm{T}} h^{-\gamma} d_1^{-\gamma} + N} \\ &= \frac{1}{3h^{-\gamma} + 1/\epsilon}, \end{aligned} \quad (8)$$

where N is the thermal noise power and ϵ is the received signal-to-noise ratio (SIR).

IV. Analytical Results

The following results are obtained using the parameters of Table I. The packet transmission time $T = L/R$ is 20msecs and the noise power N is 8×10^{-17} W. Assuming that the nodes communicate at the channel capacity, the SINR budget λ is calculated using the formula $R = B\log_2(1+\lambda)$ to give $\lambda = 3$. The node density δ is computed as $\delta = XY/A$, where Y is the number of linear clusters and X is the number of sensors per cluster and are set to be equal in the proposed application. Transmit and receive energy is assumed to be equal $(\alpha = 1)$.

TABLE I
Parameters

Parameter	Value
Data rate R	40 kbits/s
Bandwidth B	20 kHz
Packet size T	100bytes
Channel gain coefficient γ	3
Channel gain at unit distance c	1
Patch area A	50×50 cm^2
Number of nodes	{16, 36, ...3136} nodes
Node density δ	{0.0064, 0.0140, ... 1.2554} nodes/cm^2

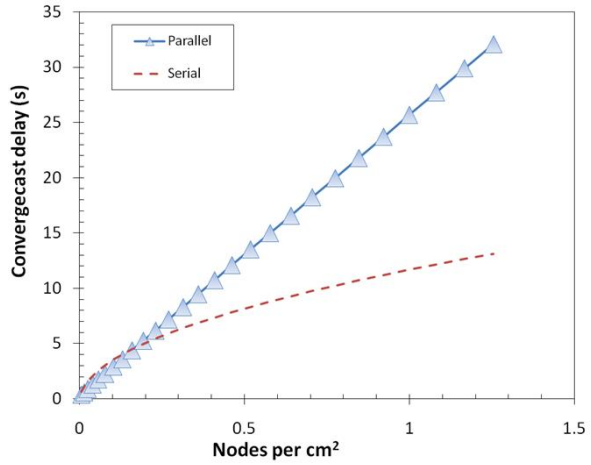

Fig. 4. Total convergecast delay versus node density for SNR=4.5

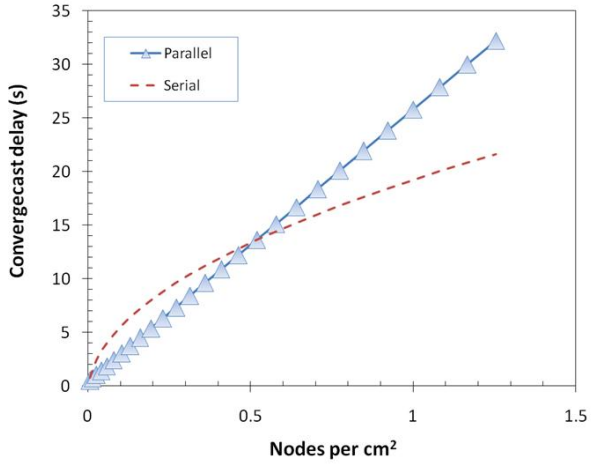

Fig. 5. Total convergecast delay versus node density for SNR=3.5

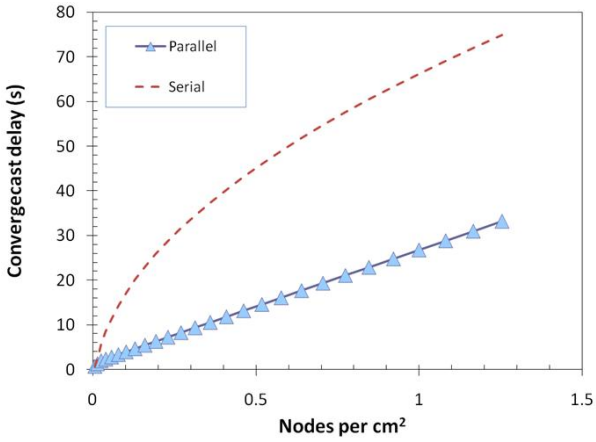

Fig. 6. Total convergecast delay versus node density for SNR=3.0

Fig. 4, Fig. 5 and Fig. 6 are plots of total convergecast delay versus node density for SNR equal to 4.5, 3.5 and 3.0,respectively, which require interferer separation values h of 3, 4 and 8 nodes, respectively using Fig. (8). The results show that SLS performs better than PLS for high SNR (small interferer separation h) and at high node densities. PLS delay is observed to increase approximately linearly with node density. The number of nodes per linear cluster increases with density and the PLS delay is proportional to the number nodes per cluster. SLS delay, on the other hand, increases with the interferer separation, because the waiting time for the last linear cluster to send its data is proportional to the minimum interferer separation h. The PLS delay varies negligibly with changes in SNR and minimum interferer separation.

It can be calculated from the plots that the average sensor delay does not increase as the number of sensors increase because of the spatial reuse of the channel by simultaneously transmitting sensors on the same linear cluster and/or neighboring clusters. From Fig. 4, for example, the delay for all schemes is approximately 5s for $\delta = 0.2$ nodes/cm^2, while the delay for $\delta = 1$ nodes/cm^2 is approximately 26s fro PLS and less than 12s for SLS. The average delay per sensor is obtained by dividing the total convergecast delay by the total number of sensors in the patch M, where $M = XY = \delta \times A$. This implies that the delay per sensor is about 0.01s for all schemes at 0.2 nodes/cm^2, which with 1 node/cm^2 remains constant at 0.01s for PLS but is less than 0.005s for SLS.

V. Conclusions

For a moderately high node density of 1 node per cm^2, a 19s and 11s convergecast delay are required for moderate and high energy consumption, respectively. This is a positive result for active airflow closed-loop control as it suggests that a relatively high density WSAN, which gives high granularity airflow measurements and control, can trade-off significantly higher delay for significant energy savings.

References

[1] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz, "Communication and Coordination in Wireless Sensor and Actor Networks." *IEEE Trans. on Mobile Computing*, vol. 6, no. 10, pp. 1116–1129, 2007.

[2] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges." *Ad Hoc Netw.*, vol. 2, no. 4, pp. 351–367, 2004.

[3] Q. Huang and Y. Zhang, "Radial coordination for convergecast in wireless sensor networks." in *Proc. of IEEE LCN'04, Tampa, USA,November 16-18, 2004.*, 2004.

[4] Y. Zhang and Q. Huang, "Coordinated Convergecast in Wireless Sensor Networks." in *Proc. of IEEE MilCom'05, Atlantic City, USA, October 17-20, 2005*, 2005.

[5] S. Gandham, Y. Zhang, and Q. Huang, "Distributed time-optimal scheduling for convergecast in wireless sensor networks." *Elsevier Computer Networks*, vol. 52, no. 3, pp. 610–629, Feb. 2008.

[6] ——, "Distributed Minimal Time Convergecast Scheduling in Wireless Sensor Networks." in *Proc. of the IEEE ICDCS'06, Lisboa, Portugal, July 4-7, 2006*.

[7] M. Gad-el-Hak, "Flow Control: Passive, Active, and Reactive Flow Management," Cambridge University Press, New York, pp. 150–188, 2000.