Northumbria Research Link

Citation: Dai, Xuewu, Omiyi, Peter, Bür, Kaan and Yang, Yang (2014) Interference-aware convergecast scheduling in wireless sensor/actuator networks for active airflow control applications. Wireless Communications and Mobile Computing, 14 (3). pp. 396-408. ISSN 1530-8669

Published by: Wiley-Blackwell

URL: http://dx.doi.org/10.1002/wcm.2190 <http://dx.doi.org/10.1002/wcm.2190>

This version was downloaded from Northumbria Research Link: https://nrl.northumbria.ac.uk/id/eprint/16920/

Northumbria University has developed Northumbria Research Link (NRL) to enable users to access the University's research output. Copyright © and moral rights for items on NRL are retained by the individual author(s) and/or other copyright owners. Single copies of full items can be reproduced, displayed or performed, and given to third parties in any format or medium for personal research or study, educational, or not-for-profit purposes without prior permission or charge, provided the authors, title and full bibliographic details are given, as well as a hyperlink and/or URL to the original metadata page. The content must not be changed in any way. Full items must not be sold commercially in any format or medium without formal permission of the copyright holder. The full policy is available online: http://nrl.northumbria.ac.uk/policies.html

This document may differ from the final, published version of the research and has been made available online in accordance with publisher policies. To read and/or cite from the published version of the research, please visit the publisher's website (a subscription may be required.)

Northumbria University NEWCASTLE

UniversityLibrary

Interference-Aware Convergecast Scheduling in Wireless Sensor/Actuator Networks for Active Airflow Control Applications

X. Dai *Member, IEEE*, P. E. Omiyi, K. Bür and Y.Yang *Member, IEEE*

Abstract

Emerging wireless sensor/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 comprising of multiple linear sensor clusters terminated by actuators is proposed for active airflow control with the objective of minimizing convergecast latency. Here the convergecast delay is defined as the time required from the beginning of a sampling period to all all sensor's data of this sampling period is received by the actuator. The objective is achieved by minimizing the separation distance of concurrent data transmission so that the number of nodes sending data in the same time slot is maximized. The problem turns into a scheduling problem with a proper selection of interference separation. However, most existing work on the scheduling in linear networks use the minimum separation of 2 hops to avoid collisions. This paper examines the relationship between the hop separation, signal-noise-ratio and the latency to make a selection of interference separation. A new interference aware hybrid line scheduling (HLS) algorithm is proposed and its energy consumption is analyzed. Compared with other line scheduling policies. the analysis and simulation results show that, at moderately high node densities, the proposed HLS with carefully selected hop separation is able to reduce both the delay by up to 15 % and the energy consumption somehow.

Index Terms

Wireless sensors, actuators, convergecast, scheduling, interference.

Y.Yang is the correspondence author and is with Shanghai Research Center for Wireless Communications, SIMIT, Chinese Academy of Sciences, China. email: yang.yang@shrcwc.org

I. Introduction

In aircraft, parasitic (pressure) drag and stall occur because of boundary layer separation [1] on the wings due to high angles of attack in take-off/landing, sudden pilot manoeuvres, or 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 [1], showing that the active flow control achieved through the local modulation of aircraft skin surfaces will offer great potential for significantly reducing profile drag. 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 to control the flow.

Implementing the active flow control will require a reliable network connecting hundreds of sensors, controllers and actuators embedded across the aircraft wings and fuselage. With the rapid development and successful implementation of wireless sensor networks (WSNs) in consumer products and non-time critical applications (e.g. environment monitoring, home automation), wireless sensor/actuator networks (WSANs) have been proposed for semi-autonomous, distributed monitoring and remote supervision/control in industrial processes [2], [3]. The main benefits of applying WSANs to active flow control is the removal of complex, heavy wiring. Hence, it has recently attracted attention of academic researchers and industrial engineers, for example, the Wireless Inter-connectivity and Control of Active Systems (WICAS) project [4].

A WSAN comprises of a system of sensor and actuator nodes 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 or multi-hop communications. Using the received information, the actuators perform actions to control and/or supervise the behavior of the environmental or physical process. In distributed control applications, sensors deliver periodic snapshots of the process to the relevant actuators which provide real-time control of the process. This inherent capability of WSANs to interact with and influence the physical world differentiate them from the much more common WSNs. The authors propose exploiting the capability of a WSAN to interact and influence its environment for active airflow control over aircraft wings.

One of the main challenges in applying WSAN to active flow control is how the data can be transmitted to the actuator in an efficient way. This is a convergecast protocol design problem, where convergecast is the data collection process of all sensors in the network sending data to a base station within a relatively short time period (i.e. a sampling interval in the active control application). Although a number of contention-based protocols (e.g., Carrier Sensing Multiple Access, CSMA) have been proposed for convergecast, considering the periodic nature of the data traffic in most control applications, the deterministic scheduling policies such as time-division-multiple-access (TDMA) algorithms can provide better performance in terms of spatial reuse, latency and jitter [5] [6] [7] [8]. In TDMA, the time domain is sliced into time slots with multiple, interfering transmissions assigned to different time slots. However, two or more transmissions spatially separated in such a way that they offer little or no interference to one another, can be scheduled in the same time slot. TDMA algorithms have already been applied to the convergecast problem in WSNs, with the objective of avoiding packet collisions and minimizing the convergecast latency.

Convergecast scheduling optimized for data gathering WSN applications is, however, not optimized for the interactive WSAN application proposed for active airflow control [9]. Unlike traditional WSN convergecast, where there is a single data sink [7] [6], the WSAN for the proposed application has several sinks (actuators), with sensors organized into multiple chains and sending their data to their local actuator using convergecast [9]. Convergecast scheduling for the WSAN is, therefore, required to satisfy the conflicting demands of 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 strict guarantees on packet delivery.

This paper is an extension of our previous work [8] on TDMA-based hop-by-hop WSAN convergecast scheduling strategies, where a serial linear scheduling (SLS) and parallel linear scheduling (PLS) policies [8] were studied. We now propose a new hybrid linear scheduling algorithm to further improve the latency performance and the extensions are threefold: (a) a hybrid line scheduling scheme; (b) the energy consumption analysis; (c) the relationship between hop separation and SNR. The main contribution of this paper is that the proposed hybrid scheme improves the performance of convergecast delay and a good trade off can be achieved between the latency and energy consumption by a properly selecting the hop separation. In this article, we assume that data packets are always forwarded via the next-hop node, even when there are other nodes farther downstream, i.e. more than one hop away, overhearing the transmission. In fact, hop-by-hop communication is a general case that covers longer hop communications when the intermediate nodes are bypassed. In the latter case, however, our aim is to show the basic trade-off between delay, hop separation and energy consumption. Thus, we constrain our system model to next-hop communication. It is worth noting that, the delay analysis in this article is simplified by assuming a general physical layer, as here we focus on the MAC layer and its performance comparison.

This article is organized as follows: The related work is presented in Section II. The network topology of a WSAN for active airflow control and the objective of protocol design are presented in Section III. In Section IV, three WSAN convergecast scheduling strategies are presented and analyzed mathematically to derive closed-form expressions for latency and energy consumption. Numerical and simulation results are given and discussed in Section VI, followed by our key conclusions in Section VII.

II. Related Work

Generally, the MAC protocols for WSNs can be classified into two categories: contention-based CSMA and contention-free TDMA. The reader is referred to [6] for the MAC protocols in WSN-based convergecast applications. It is worth noting that the communication reliability is essential for control applications, and data collection latency is important for applications that are required to take certain actions based on deadlines, such as the active flow control. Therefore, minimizing packet loss is a much desired feature for convergecast protocols, from which a better network throughput, smaller latency and jitter will be benefited. It is well-known that contention-based MAC protocols are not good at channel utilization, due to collisions. It is particularly true in high traffic load or high node density scenarios (for example, active airflow control requires many sensor nodes at the surface), where collisions result in loss of packets and recovery methods (e.g., backoff and retransmissions) increase the latency. Hence bandwidth and time are wasted by the collisions and backoff and additional overhead is introduced by retransmission. In this regard, avoiding packet collisions becomes particularly important [6]. On the other hand, TDMA protocols are designed to avoid collisions. Therefore, the preferred protocols are those that avoid, or at least minimize, collision and, thus, packet loss [10]. As TDMA is well suited to avoid the problem of collisions, in this section, a couple of recent research efforts in TDMA-like protocols for convergecast applications are summarized.

A number of different TDMA techniques for various networks and various design objectives (e.g. minimizing latency, minimizing energy consumption, maximizing fairness) are discussed and compared in [6], [11], where it is shown that an interference-aware TDMA scheduling is good at enabling spatial reuse. RT-Link [12] is a TDMA protocol assigning time slots in a centralized way at the gateway node to either maximize throughput or minimize end-to-end delay. Radial coordination [13] is a TDMA approach that addresses the problem of loss of packets due to congestion and collision near the sink. In radial coordination, the nodes adjust their transmission times according to a quadratic formula based on the estimated hop distance to the sink. When a query is received, a node waits for a certain time before replying, thus trying to avoid collisions. The waiting time of the node is based on its hop distance to the sink. Radial coordination also uses constrained flooding (e.g. geocasting), where each node forwards packets only if it is not too much farther away from the sink than the original sender of the packet. It is extended farther with packet aggregation and duplication in [14] to address the bandwidth bottleneck problem experienced by the sink during convergecast.

In TDMA protocols, optimum slot assignment is the key to achieving efficient channel utilization and reducing power consumption. However, most existing centralized algorithms present poor scalability, whereas most existing distributed algorithms suffer from high complexity and overhead. One solution proposed to achieve optimum slot assignment without the deficiencies mentioned is deterministic distributed TDMA (DD-TDMA) [15], according to which each node decides on its own slot assignment based on the information about its neighbors. In order to avoid packet collision, the hop distance between two transmission nodes is fixed to 2, which means no two nodes within two hop distance are allocated the same slot to transmit. This protocol requires a strict assumption that the interference range is the same as the transmission range. The node with the smallest identification number kicks off the slot assignment by broadcasting a information packet to its one-hop neighbors. Nodes receiving this information update their one-hop neighbor lists and forward the information in a random slot to the two-hop neighbors. The process is repeated until all nodes are assigned a slot. DD-TDMA also considers the energy consumption of a node due to having to wake up frequently, and proposes an optimization heuristic to avoid short duty cycles. Thus, total energy consumption can be reduced.

A heuristic TDMA protocol, called distributed convergecast scheduling, is proposed in [7], [16] aiming at minimizing the total time (measured by the number of time slots) to complete the convergecast session with one packet per node to be transmitted to the sink. It is shown that this minimization problem can be solved as an integer linear program with constraints. However, as a centralized solution, this solution is not scalable due to its exponential running time. Thus, a distributed heuristic is presented for linear, multi-line, tree and general networks, where each node is in one of three states, namely *Receive*, *Transmit* and *Idle*, and a finite state machine determines the state transitions. Nodes act according to their state (e.g., sends a packet when it is in Transmit state) and then change states simultaneously. For linear networks, the initial state of each node is determined by its hop distance mod by 3. For multi-line networks, the network is decomposed into multiple linear networks, called *branches*. Within each branch, the algorithm runs as in linear networks and the transmissions are scheduled in *parallel* along multiple branches. The sink receives packets in a turn from one branch at a time slot, where the selection of branch to deliver depends on the priority and the one with the highest number of packets left is selected. The author claims that the heuristic protocol requires only a limited buffer of 2 packets per node, saves more than

50% energy with its sleep schedule, and has a bounded latency for timely event detection. As mentioned previously, the heuristic is based on the assumption that each node has only one packet to transmit and thus a node only requires a buffer of two packets. It is also worth noting that there is an assumption that, in one linear network, two nodes separated by 2 hops can transmit without collision. Hence the proposed scheduling can achieve the lower bound on the number of time slots required for convergecast is $max(3n_k, N)$, where n_k is the number of nodes in the k-th linear network and N is the total number of nodes in the network.

As interference has impacts on both the data reliability and energy-efficiency (life-time) of the sensor network, interference-awareness is addressed as part of the convergecast tree generation process in the localized area spanning tree (LAST) protocol [17]. LAST assumes that nodes know their position as well as the positions of their neighbors, and that they can compute an interference metric based on the distance. The interference metric introduced is called the total path interference. Thus, a node can compute how its other neighbors are affected when it communicates to one of its neighbors.

Compared with those existing protocols, the network structure of our WSAN for active flow control are different from most WSN networks. Our WSAN has multiple sinks, however those existing algorithm only are designed for single sink. In most existing protocols ([6], [7]), only the hop-separation of 2 is considered, where it is assumed that the interference range is equal to the transmission range thus the hop-separation can be set to the minimum value of 2. Obviously, a smaller hop-separation allows more concurrent transmission and results in a smaller delay. Then the delay performance presented in [7] is the minimum one in theory. It, however, may be problematic in practice because of its assumption that the interference range is equal to the transmission range. In practice, the interference range is usually larger than the transmission range and the protocol using minimum hop-separation suffers a serious packet loss, which makes nonsense of the low convergecast delay and is not suitable for active flow control.

From the perspective of the active flow control application, a convergecast protocol needs to first guarantee a very low packet loss, and then reduce the delay. This paper studies various hop-separations and the hop separation is selected according to the Signal-to-Interference-Noise (SINR) and the energy consumption, which is able to achieve an optimized delay performance.

III. Network topology and Problem Description

Regarding the active flow control, the authors propose closed-loop control of the airflow for more effective and energy efficient control using a WSAN, where wireless sensors are deployed on the wings to provide real-time airflow information and decision metrics to the actuators.

A. Network topology

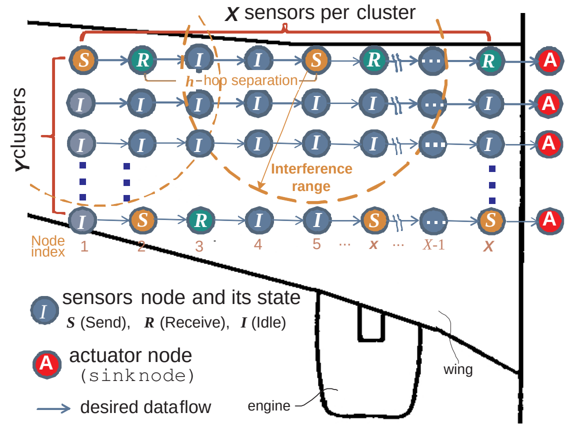

Fig. 1. WSAN with a grid topology (hop separation h=3)

As shown in Fig. 1, the sensors are deployed as a grid evenly along the surface of the airplane wing. In the application of active flow control, the sensor and actuator nodes are stationary and the separation distance between nodes are d. Specifically, we assume that sensors are organized in a chain, referred to as *linear cluster*, and each linear cluster comprises of X nodes and one sink (actuator) uniformly spaced along a straight line. The whole network consists of a group of such linear clusters. This is similar to the "linear network" in WSNs [7], [16]. The difference is that, in these WSNs, there are only one sink and, in our WSAN, there are multiple sinks and each of them is associated with a linear cluster. Each sensor has an index x depending on its hop-count from the sink (actuator), such that the sensor farthest away from the actuator of its cluster is the 1^{st} sensor of the cluster, while the sensor one-hop away is the X^{th} sensor. 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.

As shown in Fig. 1, the Y parallel linear clusters are grouped into a 'rectangular' patch, where the clusters are aligned in such a 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 two adjacent clusters is the same as the one-hop distance d within the cluster. Each linear cluster has an index y, where clusters are indexed in increasing order from one end of the patch to the other (up-to-down in Fig. 1). All the nodes are equipped with a single omnidirectional transceiver. Hence, the nodes cannot transmit and receive at the same time. All the communication is carried over the same frequency channel. The data rate of every wireless link in the network is the same and all data packets have the same length. The duration of a time slot is equal to the transmission duration of a packet allowing the transmission of exactly one packet. We consider applications wherein each actuator has to receive every data packet sent by the sensors in its cluster without aggregation of the data, as required by the study of active flow control. This is because, at the earlier stage of WSAN-based active flow control, one of the main purposes is to understand the correlation among sensor data, rather than to make use of the correlation to reduce the data amount. It is assumed that every node is aware of its hop-count, and hence its index, and its cluster index through an initialization phase and system updates.

B. *Problem Description*

Similar to all communication networks, all sensor nodes in the proposed linear cluster network have 3 states, namely receive ($\mathcal{R}$), send ($\mathcal{S}$), and idle ($\mathcal{I}$) states. The exception are the 1^{st} sensor node with only two states, $\mathcal{S}$ and $\mathcal{I}$, and the actuator with only two states $\mathcal{R}$ and $\mathcal{I}$. From the view point of communication, the actuator nodes (denoted by **A** in Fig.1) is also working in one of these three states. Using less states we can avoid frequent switch of radio state and save energy. In any given linear cluster, if the x^{th} sensor is in the $\mathcal{S}$ state, then the $(x+1)^{\text{th}}$ sensor next to it must be in the $\mathcal{R}$ state. The states of all the downstream nodes i, where $x+1 \leq i \leq X$, depend on the minimum interference separation required by the x^{th} node.

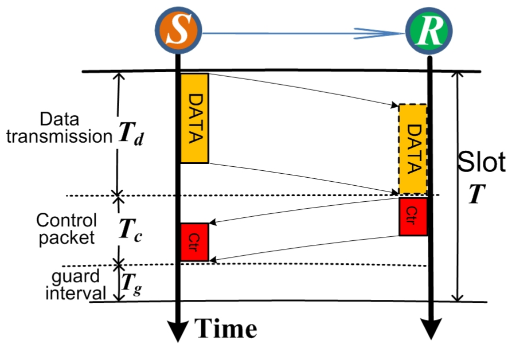

Fig. 2. The package exchange scheme between the adjacent $\mathcal{S}$-$\mathcal{R}$ nodes

The package exchange mechanism between the adjacent $\mathcal{S}$-$\mathcal{R}$ nodes is illustrated in Fig.2. The duration of T_d is for the data packet transmission and T_c is reserved for the downstream receiver and the sink to reply with a control message once a data packet is received. The control message can be used for network management or carres the time information for synchronization purposes. For example, the sink may want to update the sensor node's parameters by sending a special control message. The guard interval T_g is used to ensure data transmission in successive time slots do not interfere with one another, so that the requirement of high accurate time synchronization is relaxed. Therefore, we have

$$T = T_g + T_d + T_c. \tag{1}$$

In this paper, the minimum interference separation is the distance required between a node in $\mathcal{R}$ state and an interfering transmitting node (in $\mathcal{S}$ state) either in the same cluster or neighboring clusters. The minimum interference separation is measured in hops. An h hop separation means the concurrent transmission nodes in $\mathcal{S}$ state must be separated by a distance of hd, so that concurrent transmissions do not interfere with each other. Here, h is referred to as the *separation coefficient*. In other words, 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 $\mathcal{S}$ state is the $(x+h+1)^{\mathrm{th}}$ node. This separation rule also applies to the inter-cluster interference. That is, if a node is in the $\mathcal{R}$ state, all the other nodes within the radius of hd must be in the $\mathcal{I}$ state, whatever they are in the same cluster or in neighboring clusters. Other nodes in the neighboring clusters, which are on the border or outside a radius hd of the $\mathcal{R}$ node, *may be* in the $\mathcal{S}$ state depending on its scheduling scheme. This separation is required to ensure that, for the x^{th} node in $\mathcal{R}$, the interference power from either the $(x+h)^{\mathrm{th}}$ $\mathcal{S}$ node in the same cluster or the nodes in neighboring clusters is sufficiently attenuated and the received signal strength from its desired $(x-1)^{\mathrm{th}}$ node is relatively strong to ensure successful packet transmission without collisions.

It is worth noting that, in previous work [6], [7], [16], it was assumed the interference range was the same as the transmission range and a fixed interference separation of 2 hops was adopted. It is true only when two concurrent transmissions are allowed in a network. The more common practice in wireless networks is to have more than two concurrent transmissions. Due to the additive feature of receiving signal power, the interference range is larger than the transmission range. Hence, we use a variable separation rather than a fixed separation of 2-hop to address the issue.

In order to minimize convergecast delay, on the other hand, 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 $x+1^{\mathrm{th}}$ 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 $(x-1)^{\mathrm{th}}$ 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.

Although $h = 2$ is the absolute minimum value of the hop separation coefficient, it may not be good enough. 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 to define an interference-aware schedule that minimizes total convergecast delay and energy consumption across all clusters, by finding an optimum hop separation coefficient h so that a good trade-off can be achieved between packet transmission time and sensor channel access delay.

C. Notation

In the investigated WSAN application, all sensor data are generated periodically within the same time-frame and nodes are aware of their hop-count x and one-hop neighborhood. All the data packets have the same length of L bytes and T_d denotes the transmission duration of one data packet. The length of a time slot is T as given in (1). The cluster convergecast delay is defined as the time required for a linear cluster to send all its data to its actuator measured from the time the cluster begins sending. The cluster convergecast delays of the y^{th} linear cluster in PLS, SLS and HLS are denoted by $D_{\text{P}y}$, $D_{\text{S}y}$ and $D_{\text{H}y}$, respectively. The total network convergecast delay is defined as the time between the first node starting to send data and all nodes' data (in a single sampling period) have been received by the actuators. The network convergecast delay in PLS, SLS and HLS are denoted by D_P, D_S and D_H, respectively.

IV. Convergecast Scheduling

This section presents three interference-aware TDMA based convergecast scheduling policies and their convergecast delays are analyzed.

A. Parallel Line Scheduling

The goal of parallel line scheduling (PLS) [8] is to maximize the number of linear clusters communicating in parallel. Only one sensor is transmitting at any given time per cluster. In order to maintain the required separation (hd) between nodes in $\mathcal{R}$ state and adjacent interfering nodes in $\mathcal{S}$ state, adjacent linear clusters do not begin communication simultaneously but in a staggered fashion. Each cluster waits for the preceding adjacent cluster to communicate through its first $h + 1$ hops before beginning with its first hop, which ensures the required minimum separation.

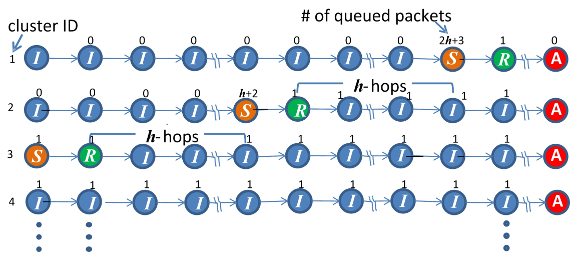

Fig. 3. Illustration of parallel line scheduling

Fig. 3 shows a snapshot of PLS operation with 4 linear clusters, each of which is terminated by an actuator . In this snapshot, Cluster 3 is in the process of its first hop, with its first node (far left) in $\mathcal{S}$ state and its second node in $\mathcal{R}$ state. Cluster 2 began sending its data $h+1$ hops before Cluster 3. Thus, its first $h+1$ nodes are in $\mathcal{I}$ state, while its $(h+2)^{\text{th}}$ node is in $\mathcal{S}$ state, which provides the necessary (hd) separation for the second node of Cluster 3 that is in $\mathcal{R}$ state. Similarly Cluster 1 began $h+1$ hops before Cluster 2, and its only node in the $\mathcal{S}$ state is separated by a distance of hd from the $(h+3)^{\text{th}}$ 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.

The total convergecast delay D_{P} of PLS is [8]

$$D_{\text{P}} = 0.5X(X+1)T + (Y-1)(h+1)T. \tag{2}$$

This can be explained as follows: Each sensor in PLS sends its data together with the data of all its upstream nodes in a single burst to its next hop neighbor. Therefore, the cluster convergecast delay for y-th cluster is [8]

$$D_{\text{P}y} = \sum_{x=1}^{X} xT = 0.5X(X+1)T, \tag{3}$$

In PLS, although the length of convergecast delay of each cluster is the same, they may start at different time. Given the staggered manner with which clusters begin sending their data in PLS, the last cluster has to wait for $Y-1$ preceding adjacent clusters to complete their first $h+1$-hop data transmission before it starts its first hop. Therefore, the waiting time for the last cluster (the Y^{th} cluster) is $(Y-1)(h+1)T$. The whole convergecast delay is equal to the Y-th cluster's waiting time plus the duration that the Y-th cluster needs to complete its data transmission. Let $D_{\text{P}Y}$ denote the cluster convergecast delay of the Y-th cluster, the convergecast delay of PLS is $D_{\text{P}Y} + (Y-1)(h+1)T$ and then it is easy to verify Eq. (2).

B. Serial Line Scheduling

In contrast to PLS in which only one transmission is allowed per cluster in any time slot, serial line scheduling (SLS) [8] attempts to reduce the convergecast delay of per-cluster by maximizing the number of nodes per cluster that are simultaneously in the $\mathcal{S}$ 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 $\mathcal{S}$ state. In this case, in order to maintain the required separation (hd) 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. 4 shows a snapshot of SLS operation.

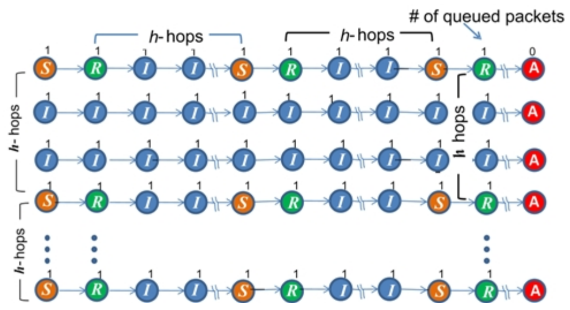

Fig. 4. Illustration of serial line scheduling

As shown in [8], when the number X of sensors in a cluster is less than or equal to $h+1$, the maximum possible interference separation is less than the minimum separation distance hd and only one node per cluster can transmit in the same time slot. Thus, similar to $D_{\mathrm{P}y}$ in (3), the convergecast delay $D_{\mathrm{S}y}$ of the y^{th} linear cluster can be derived as

$$D_{\mathrm{S}y} = \sum_{x=1}^{X} xT = 0.5X(X+1)T, \text{ for } X \leq h+1. \tag{4}$$

When $X > h+1$, multiple sensors can transmit in the same time slot on the same linear cluster. The convergecast delay is comprised of two components. The first is the delay δ_{h+1} to collect all the data of the first $h+1$ sensor nodes to the $(h+2)^{\mathrm{th}}$ sensor, which is given as follows [8]

$$\delta_{h+1} = \sum_{x=1}^{h+1} xT = 0.5(h+1)(h+2)T. \tag{5}$$

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 $(h+2)^{\mathrm{th}}$ sensor to the actuator. As the $(h+2)^{\mathrm{th}}$ node and its downstream nodes have transmitted their data, the data to be forwarded over $X - h - 1$ hops to the actuator is fixed of $(h + 1)$ packets. This delay is $\delta_{h+2} = (X - h - 1)(h + 1)T$. Therefore, the convergecast delay $D_{\mathrm{S}y}$ of the y^{th} linear cluster when using SLS is

$$\begin{aligned} D_{\mathrm{S}y} &= 0.5(h + 1)(h + 2)T + (X - h - 1)(h + 1)T \\ &= (X - 0.5h)(h + 1)T, \text{ for } X > h + 1. \end{aligned} \quad (6)$$

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 complete all their data transmission before beginning its first hop. If the number of clusters Y is less than h, only one cluster can be active. Therefore, the total convergecast delay of the patch using PLS is given as [8]

$$D_{\mathrm{S}} = \min(Y, h + 1)D_{\mathrm{S}y}. \quad (7)$$

C. Hybrid Line Scheduling

Hybrid line scheduling (HLS) combines features of both SLS and PLS with multiple sensors in the same cluster sending data in the same time slot and adjacent clusters sending their data simultaneously. The objective is to achieve the best trade-off between maximizing the number of parallel communicating linear clusters and minimizing the per-cluster data delivery latency. The HLS operation is illustrated using the example shown in Fig. 5, where the separation factor $h = 3$. Like PLS, HLS maintains the required separation between receiving and transmitting nodes in adjacent clusters by staggering the times adjacent clusters begin sending their data by $(h+1)$-hops. In addition, like SLS, HLS permits multiple nodes per cluster to transmit in the same time slot. However, to enable adjacent clusters to simultaneously send their data, HLS uses a larger separation between receiving and transmitting nodes of the same cluster than the h-hops used in SLS. Specifically, HLS uses $[h(h+1)-1]$ hop separation, which is the minimum separation for each cluster that allows both multiple nodes transmitting per cluster and simultaneously active clusters.

In HLS, when the number of sensors in a cluster is less than or equal to $h(h + 1)$, the maximum possible interference separation between a receiving node and an interfering transmitter is $h(h + 1) - 2$, which is less than the minimum separation distance $[h(h+1)-1]d$. Therefore, only one node per cluster can transmit in the same time slot. In this case, each sensor sends its data together with the data of all its upstream nodes in a single burst to its next hop neighbor. This implies that the convergecast delay

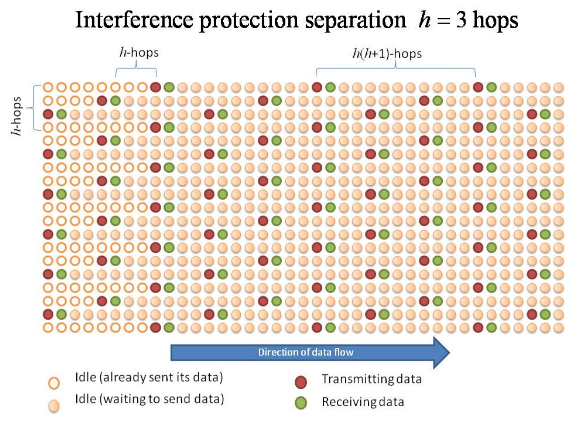
Interference protection separation $h = 3$ hops
h-hops
$h(h+1)$-hops
h-hops
Direction of data flow
Idle (already sent its data)
Idle (waiting to send data)
Transmitting data
Receiving data

Fig. 5. Illustration of hybrid line scheduling

$D_{\mathrm{H}y}$ of the y^{th} linear cluster is derived similarly to $D_{\mathrm{P}y}$ in (3) as

$$D_{\mathrm{H}y} = \sum_{x=1}^{X} xT = 0.5X(X+1)T, \text{ for } X \leq h(h+1). \tag{8}$$

When $X > h(h+1)$, multiple sensors can transmit in the same time slot on the same linear cluster, as it is possible to maintain the minimum separation between a node in $\mathcal{R}$ state in any given cluster and a node in $\mathcal{S}$ state in the same or any other cluster. Only the first $h(h+1)$ sensors forward their data together with that of all their upstream nodes in a single burst. This is because by the time the data of all sensors with indexes less than $h(h+1)+1$ arrive at sensors with indexes $h(h+1)+1$ and higher, the latter have already sent their own data downstream. Thus, the convergecast delay of the cluster is comprised of two components. The first is the delay α_1 to forward all the data of the first $h(h+1)$ sensors to the $h(h+1)+1^{\mathrm{th}}$ sensor that has already sent its own data, which is given as follows

$$\alpha_1 = \sum_{x=1}^{h(h+1)} xT = 0.5h(h+1)[h(h+1)+1]T.$$

The second component of the convergecast delay, α_2, is the delay to forward all the data of the first $h(h+1)$ sensors from the $h(h+1)+1^{\mathrm{th}}$ sensor to the actuator. This data comprises of a fixed length burst of $h(h+1)$ packets to be forwarded over $X - h(h+1)$ hops to the actuator. This delay is given as

$$\alpha_2 = [X - h(h+1)]h(h+1)T. \tag{9}$$

Therefore, the y-th cluster's convergecast delay $D_{\mathrm{H}y}$ in HLS is

$$\begin{aligned} D_{\mathrm{H}y} &= \alpha_1 + \alpha_2 \\ &= [X - 0.5h(h+1) + 0.5]h(h+1)T \\ &\qquad \text{for}\, X > h(h+1). \end{aligned} \tag{10}$$

Like PLS, each cluster waits for the preceding adjacent cluster to communicate its first $h+1$ hops before beginning its first hop, and the last cluster waits for $Y-1$ preceding adjacent clusters to begin sending their data before it begins its first hop. Therefore, the total delay before the last cluster of the patch begins its first hop is $(Y-1)(h+1)T$. Given that the convergecast delay $D_{\mathrm{H}Y}$ of the last cluster (the Y^{th} cluster) is given by (8) and (10), then the total convergecast delay of the patch using HLS is given as

$$\begin{aligned} D_{\mathrm{H}} &= D_{\mathrm{H}y} + (Y-1)(h+1)T \\ &= 0.5X(X+1)T + (Y-1)(h+1)T \\ &\qquad \text{for}\, X \le h(h+1), \end{aligned}$$

and

$$\begin{aligned} D_{\mathrm{H}} &= [X - 0.5h(h+1) + 0.5]h(h+1)T \\ &\quad +(Y-1)(h+1)T \\ &= [Xh - 0.5h(h^2+h+1) + Y - 1](h+1)T \\ &\qquad \text{for}\, X > h(h+1). \end{aligned}$$

V. Energy and Interference Management

In the application of active air flow control, the energy consumption is an important metric of the network's performance. Both smaller latency and lower energy consumption are desired. As shown in this section, the energy consumption of a convergecast network is related to the hop separation and thus the interference management, it is necessary to analyze these scheduling algorithms's energy and the impacts of interference.

A. Energy Consumption Analysis

In application of convergecast to the air flow control, the energy consumption for data transmission depends on the hop separation, but is independent of the scheduling sequence. More specifically, once the transmission power is determined by the SNR requirement and the hop separation, the energy consumption is determined by how many packets are transmitted through the network. Since the number of packet to be transmitted is determined by the sampling process and, in turn, the number of sensor, thus all these three scheduling schemes have the same number of data transmission and the number does not change in these three schemes. It is worth noting that, due to the real-time requirements, there is no need of adopting retransmission in the airflow control application.

Assuming the first-order radio model [18] [19], let ϵ_{elec} and $\epsilon_{\text{amp}}(d^2)$ denote the energy consumption rate (J/bit) of the transceiver electronics and the transmitter amplifier, the total power consumption rate (in Watt) for transmission is $S_{\text{T}} = R\epsilon_{\text{elec}} + R\epsilon_{\text{amp}}(d^2)$, where R is the transmission rate (bits/s) and $\epsilon_{\text{amp}}(d^2)$ is a linear function of the square of the one-hop distance d [19]. The transmission energy consumed by the x^{th} sensor of y^{th} cluster is given as

$$E_{\text{T}xy} = xT_dS_{\text{T}}. \tag{11}$$

and the energy consumed by the x^{th} sensor for replying with control packet is

$$E_{\text{T}x+1,y} = xT_cS_{\text{T}}. \tag{12}$$

Furthermore, considering that the energy consumption in reception results from only the transceiver electronics, reception consumes less energy than transmission. The reception energy consumption is αS_{T}, where $\alpha \leq 1$. Therefore, the energy consumed by the x^{th} sensor of y^{th} cluster for reception is

$$E_{\text{L}xy} = (x-1)\alpha T_dS_{\text{T}} + x\alpha T_cS_{\text{T}}. \tag{13}$$

Therefore, the total energy consumed by the x^{th} sensor is obtained from the sum of the components on the right hand side of (11) and (13) to give

$$\begin{aligned} E_{xy} &= xT_dS_{\text{T}} + (x-1)\alpha T_dS_{\text{T}} + x\alpha T_cS_{\text{T}} \\ &= (x+\alpha x-\alpha)T_dS_{\text{T}} + x\alpha T_cS_{\text{T}}. \end{aligned} \tag{14}$$

The total energy consumed by the y^{th} cluster in one data collection cycle is obtained as

$$\begin{aligned} E_y &= \sum_{x=1}^{X} E_{xy} \\ &= \sum_{x=1}^{X} \left[(x+\alpha x-\alpha)T_dS_{\text{T}} + x\alpha T_cS_{\text{T}}\right] \\ &= 0.5X(X+\alpha X-\alpha+1)TS_{\text{T}} \\ &\quad +0.5X(X+1)\alpha T_cS_{\text{T}}. \end{aligned} \tag{15}$$

Given that all linear clusters have the same number of sensors and thus have the same per-cluster energy consumption, the total energy consumption of the entire network of Y clusters is

$$E_{\text{radio}} = 0.5XY(X + \alpha X - \alpha + 1)TS_{\text{T}}. \tag{16}$$

Note that the energy consumption calculated in (16) only takes into account the minimum energy required by the radio transmission. As it does not include node's other power consumption (e.g. CPU data processing), E_{radio} (16) is the lower boundary of the energy consumption indeed. Let E_{cpu} denote the overhead energy consumption on data processing of one packet (e.g., encapsulation, decapsulation, transferring data between CPU and radio chip), E_{cpu} depends on the number of transmitted packets, but is independent of the communication distance. Since the total number of packets transmission in each cluster is $\sum_{x=1}^{X} x = 0.5X(X+1)$, the total overhead energy consumption of all Y clusters is $0.5X(X+1)\cdot Y$. Thus the whole energy consumption now is

$$E = E_{\text{radio}} + 0.5X(X+1)YE_{\text{cpu}}. \tag{17}$$

B. Interference Management

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). 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) is given by (Eq. 2.39 in [20])

$$I = S_{\text{T}}K(hd)^{-\gamma}, \tag{18}$$

where K 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

$$\lambda = \frac{cS_{\text{T}}d^{-\gamma}}{3cS_{\text{T}}h^{-\gamma}d^{-\gamma} + N} = \frac{1}{3h^{-\gamma} + 1/\epsilon}, \tag{19}$$

where N is the thermal noise power and ϵ is the receiver's signal-to-noise ratio (SNR) which denotes the required SNR threshold to meet the desired SINR budget.

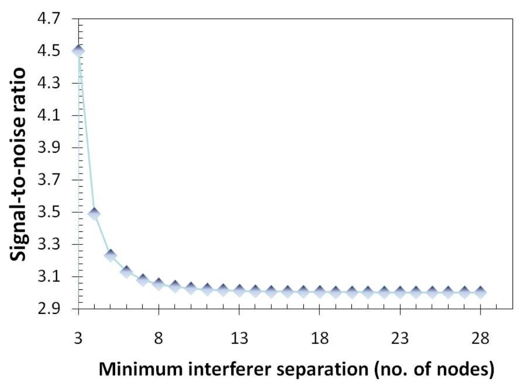

Fig. 6. the required SNR ϵ versus minimum interferer separation h when $\lambda = 3$

TABLE I
PARAMETERS

Parameter	Value
Data rate R	40 kbits/s
Bandwidth B	20 kHz
Data packet size L_d	100bytes
Control packet size L_c	20 bytes
Guard interval T_g	$1ms$
Channel gain coefficient γ	3
Channel gain at unit distance K	1
Overhead energy consumption E_{cpu}	9×10^{-5} nJ
Area A	50×50 cm^2
Number of nodes $(X \times Y)$	{16, 36, 64, 100, ...1444} nodes
Node density Q	{0.0064, 0.0140, 0.0256, 0.0400, ...0.5776} nodes/cm^2

Considering stationary white noises and assuming that nodes communicate at the channel capacity, the required SINR budget λ is determined by$R = B\log_2(1+\lambda)$, where R is the data rate (bps) and B is the channel bandwidth. Fig. 6 illustrates the relationship between the SNR requirement (threshold) ϵ and the interference separation h to achieve a SINR budget $\lambda = 3$. It shows that, given a link budget λ, a smaller interference separation requires a higher signal power to achieve the desired SNR. Thus, a smaller interference separation demands more transmission power and energy consumption. At very low signal powers, the SINR equals SNR and becomes independent of the interferer separation.

Equation (19) can be rewritten as

$$P_{TX} = \frac{P_0}{1/\lambda - 3h^{-\gamma}} \tag{20}$$

where P_{TX} is the received signal power and P_0 is the noise power. Equation (20) and Fig. 6 can be understood as the lower boundary of transmission power consumption. Usually, the transmission power in practice is set a bit higher to make the resulting SNR over the SNR threshold.

VI. Simulation Results

This section compares the numerical analysis and the corresponding simulation results from OMNET++ platform with the parameters in Table I. The data packet transmission time $T_d = L_d/R$ is 20 ms, $T_c = L_c/R = 4ms$ and the duration of a slot is $T = 25ms$. The noise power N is 8×10^{-17} W. The node density Q is computed as $\delta = XY/A$, where Y is the number of linear clusters and X is the number of sensors per cluster. Transmit and receive energy consumption is assumed to be equal ($\alpha = 1$).

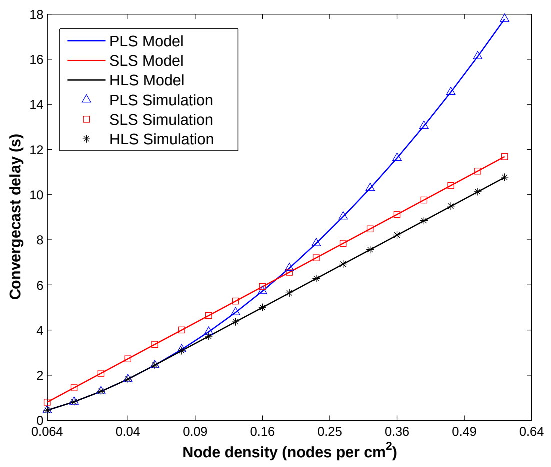

Fig. 7. Total convergecast delay for SNR=4.5(equivalent h=3)

In the simulation, since the hop separation has been selected properly by the proposed interference management algorithm that makes the interferences negligible for the receiver, it is reasonable to assume that it is interference-free under this condition of the proper selection of the hop separation. Fig. 7, Fig. 8 and Fig. 9 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, respectively (using Fig. 6). The results show that SLS has the worst performance in most cases expect the high SNR (small interferer separation h) scenario at high node densities, as shown in Fig. 7. HLS outperforms the parallel schemes

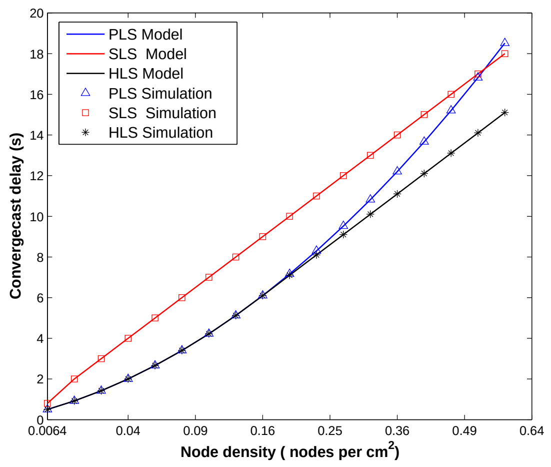

Fig. 8. Total convergecast delay for SNR=3.5(equivalent h=4)

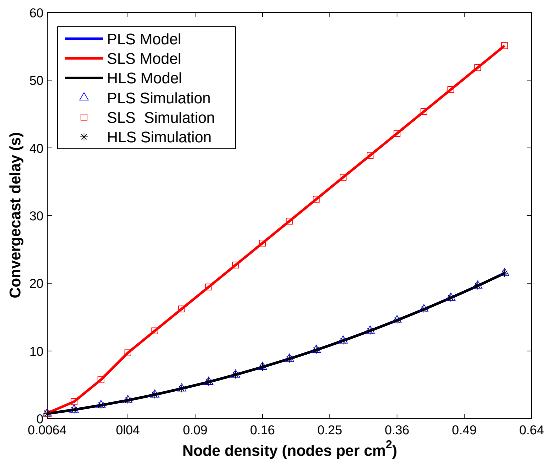

Fig. 9. Total convergecast delay for SNR=3.0 (equivalent h=8)

in most observed observed cases unless the condition $X < h*(h+1)$ is violated. In particular, at high hop separation (Fig. 9), a similar performance of HLS and PLS can be seen. This is because the $SNR = 3.0$ requiring $h = 8$ which makes $h*(h+1)$ greater than X at all node densities, thus equation (8) applies to HLS and both HLS and PLS result in the same performance. Similar phenomena are observable at low node densities in Fig. 7 and Fig. 8.

The delays of all three scheduling increase 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. SLS delay is observed to increase approximately linearly with node density, because the SLS delay is proportional to the number of nodes per cluster and the number of nodes per cluster increases with node density. The PLS delay, on the other hand, increases exponentially with respect to the node density. While HLS delay is a mixture of SLS and PLS. At low density, although HLS's delay increases exponentially as HLS, it is lower than SLS. When the node density increases further, the PLS delay goes over the SLS delay exponentially. However, the HLS delay stops exponential increase in the mediate node density and goes up linearly after that.

It can be concluded from the plots that the average delay per sensor node 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. The average delay per sensor node is obtained by dividing the total convergecast delay by the total number of sensor nodes in the patch M, where $M = XY = Q \times A$. From Fig. 7, for example, the delay for all schemes is approximately 5s for 0.15 nodes/cm^2, while the delay for 0.49 nodes/cm^2 is approximately 17s for PLS and less than 11s for the other schemes. This implies that the delay per sensor node 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 the other schemes.

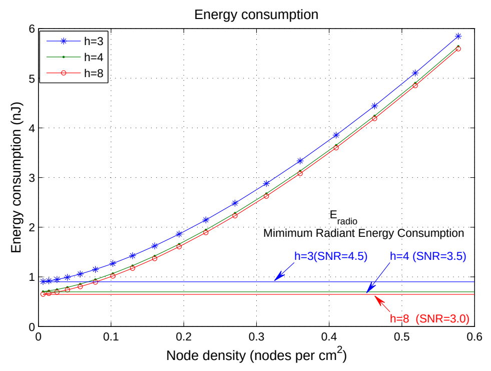

Fig. 10. Energy consumption versus node density

The total energy consumption of the entire network versus node density for different values of hop separation are shown in Fig. 10. The minimum radiant energy consumption E_{radio} for various hop separations are also illustrated in Fig. 10, where it can be seen that E_{radio} does not vary significantly with node density (with the current parameters) and notably does not increase. This is because, with increasing node density, the transmit power required to maintain the given SNR target over the shorter communication distances decreases. Hence, although the number of nodes increases, the individual radiant energy consumption for each packet transmission declines and the total radiant energy consumption is balanced. However, the overhead energy consumption E_{cpu} is constant no matter what the radiant transmission power is. Thus $\sum E_{\mathrm{cpu}}$ increases nonlinearly with respect to the increasing node density, as shown in equation (17). Note that, as the energy consumption is determined by the hope separation and number of packets transmitted through the network, all the three schemes have he same energy consumption for the same hop separation.

Furthermore, it can be seen that the scheduling at SNR of 4.5 (h=3) consumes more energy than that of a lower SNR. This verifies the analysis results of equation (20). Comparing the energy consumption plots to the delay plots, operating at an SNR of 4.5 results in the lowest delay and highest energy consumption. A drop in the target SNR from 4.5 to 3.5 results in apparent energy savings with a relatively moderate delay penalty. Changing SNR from 3.5 to 3.0 results in negligible energy savings but significant increases delay. Therefore, of the SNR values considered, the appropriate operating point is at a SNR of 3.5 which requires a minimum interferer separation of 4 hops.

VII. Conclusions

Although the WSAN technology has the potential to enable semi-autonomous air-flow control to improve the aerodynamic performance of aircraft, the communication protocol has to be designed carefully to reduce the convergecast delay. In this paper, three linear scheduling schemes, namely PLS, SLS and HLS, for linear cluster networks are presented for active flow control and the relationship between the hop separation and the latency is examined. It has been shown that a smaller interference separation requires a higher SNR threshold thus a higher energy consumption. However, the convergecast delay benefits from smaller separation hops. Compared to the parallel line scheduling and serial line scheduling, the proposed hybrid line scheduling results in a 15% reduction in delay and energy saving at moderately high sensor node densities. The appropriate operating point is at a SNR of 3.5 requiring a minimum interference separation of 4 hops.

References

[1] Gad-el-Hak M. Flow Control: Passive, Active, and Reactive Flow Management. Cambridge University Press, New York 2000.

[2] Melodia T, Pompili D, Gungor VC, Akyildiz IF. Communication and Coordination in Wireless Sensor and Actor Networks. *IEEE Trans. on Mobile Computing* 2007; **6**(10):1116–1129.

[3] Akyildiz IF, Kasimoglu IH. Wireless sensor and actor networks: research challenges. *Ad Hoc Netw.* 2004; **2**(4):351–367.

[4] Rolls-Royce Control & Systems University Technology Centre. Wireless interconnectivity and control of active systems (WICAS) (EPSRC project EP/F00477X/1) 2007. URL `http://www.sheffield.ac.uk/systemsutc`.

[5] Ergen S, Varaiya P. Tdma scheduling algorithms for wireless sensor networks. *Wireless Networks* 2010; **16**:985–997.

[6] Incel OD, Ghosh A, Krishnamachari B. *Scheduling Algorithms for Tree-Based Data Collection in Wireless Sensor Networks*, chap. Theoretical Aspects of Distributed Computing in Sensor Networks, Ed. S. Nicoletseas and J. Rolim. Springer Verlag, 2010; 439–477.

[7] Gandham S, Zhang Y, Huang Q. Distributed time-optimal scheduling for convergecast in wireless sensor networks. *Computer Networks* 2008; **52**(3):610–629.

[8] Omiyi PE, Bür K, Yang Y. Distributed convergecast scheduling for reduced interference in wireless sensor and actuator networks. *IEEE Wireless Communications and Networking Conference (WCNC'2010)*, IEEE, 2010; 1–5.

[9] Bür K, Omiyi P, Yang Y. Wireless sensor and actuator networks: Enabling the nervous system of the active aircraft. *IEEE Communications Magazine* 2010; **48**(7):118 –125.

[10] Zhang H, Arora A, Choi Y, Gouda M. Reliable bursty convergecast in wireless sensor networks. *Proc. of ACM MobiHoc'05*, 2005; 266–276.

[11] Incel O, Krishnamachari B. Enhancing the data collection rate of tree-based aggregation in wireless sensor networks. *SECON'2004*, 2008; 569 577.

[12] Rowe A, Mangharam R, Rajkumar R. Rt-link: A time-synchronized link protocol for energy-constrained multi-hop wireless networks. *Ad hoc Networks* 2008; **6**:1201–1220.

[13] Huang Q, Zhang Y. Radial coordination for convergecast in wireless sensor networks. *Proc. of IEEE LCN'04*, 2004; 542–549.

[14] Zhang Y, Huang Q. Coordinated Convergecast in Wireless Sensor Networks. *Proc. of IEEE MilCom'05*, 2005; 1152–1158.

[15] Wang Y, Henning I. A Deterministic Distributed TDMA Scheduling Algorithm for Wireless Sensor Networks . *Proc. of WiCom'07*, 2007; 2759–2762.

[16] Gandham S, Zhang Y, Huang Q. Distributed Minimal Time Convergecast Scheduling in Wireless Sensor Networks. *Proc. of the IEEE ICDCS'06*, 2006.

[17] Johansson T, Osipov E, Carr-Motyckova L. Interference Aware Construction of Multi- and Convergecast Trees in Wireless Sensor Networks. *Proc. of NEW2AN'08*, 2008; 72–87.

[18] Yang Y. SHORT: Shortest hop routing tree for wireless sensor networks. *Int. J. Sensor Networks* 2007; **2**(5):368–374.

[19] Heinzelman WB, Chandrakasan AP, Balakrishnan H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications* 2002; **1**(4):660–670.

[20] Goldsmith A. *Wireless Communicat5ions*. Cambridge University Press, 2005.