Message-ferrying delay-tolerant routing in linear wireless sensor networks

Imad Jawhar*

Faculty of Engineering,
Al Maaref University,
Beirut, Lebanon
Email: imad.jawhar@mu.edu.lb
*Corresponding author

Sheng Zhang

State Key Laboratory for Novel Software Technology,
Nanjing University, China
Email: sheng@nju.edu.cn

Jie Wu

Department of Computer and Information Sciences,
Temple University,
Philadelphia, Pennsylvania, USA
Email: jiewu@temple.edu

Nader Mohamed

Department of Computer Science, Information Systems, and Engineering,
California University of Pennsylvania,
California, Pennsylvania, USA
Email: mohamed@calu.edu

Abstract: Due to linearity of a monitored structure, wireless sensor network (WSN) topology exhibits a linear form and the resulting network is named a linear sensor network (LSN). In order to communicate data from sensor nodes (SNs) to the sink, a multi-hop approach can be used. However, this would result in significant transmission energy loss, as the SNs would be involved in the transmission of their data as well as the data from other sensor nodes. In this paper, we introduce a ferry-based LSN (FLSN) model, where a ferry node such as a robot or other types of moving vehicles goes along the LSN, collects the data from the SNs and delivers it to the sink at the end of the network. Overall, this approach saves SN energy, increases network lifetime, and reduces transmission interference. Four ferry movement algorithms are presented, simulated and analysed under various network conditions.

Keywords: routing; delay-tolerant networks; DTNs; mobile ad hoc networks; MANETs; wireless sensor networks; WSNs; ferry; monitoring.

Reference to this paper should be made as follows: Jawhar, I., Zhang, S., Wu, J. and Mohamed, N. (2022) 'Message-ferrying delay-tolerant routing in linear wireless sensor networks', *Int. J. Sensor Networks*, Vol. 39, No. 1, pp.1–17.

Biographical notes: Imad Jawhar is a Professor and the Chairman of the Department of Computer Engineering and Technology at Al Maaref University, Lebanon. He has received his BS/MS in Electrical Engineering from UNC-Charlotte, an MS in Computer Science, and PhD in Computer Engineering from FAU, Florida. He published numerous papers in international journals, and conferences. He worked at Motorola in the design and development cutting-edge communication systems and was the President and Owner of Atlantic Computer Training and Consulting, Florida, USA. His research focuses on wireless, ad hoc, and sensor networks, cyber-physical systems, mobile computing, distributed, and multimedia systems. He is a member of IEEE, and ACM.

Sheng Zhang is an Associate Professor in the Department of Computer Science and Technology, Nanjing University. His research interests include edge computing and edge intelligence. To date, he has published more than 80 papers, including those appeared in JSAC, TON, MobiHoc,

and INFOCOM. He received the Best Paper Award of ICCCN 2020 and the Best Paper Runner-Up Award of MASS 2012. He is the recipient of the 2020 ACM Nanjing Rising Star Award and the 2015 ACM China Doctoral Dissertation Nomination Award.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell Professor at Temple University. His current research interests include mobile computing and wireless networks, routing protocols, network trust and security, distributed algorithms, applied machine learning, and cloud computing. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards. He is/was the General Chair/Co-chair for IPDPS'08, DCOSS'09, ICDCS'13, MobiHoc'14, ICPP'16, CNS'16, WiOpt'21, and ICDCN'22 as well as the Program Chair/Co-chair for MASS'04, INFORCOM'11, CNCC'13, and ICCCN'20. He is a Fellow of the AAAS and Fellow of the IEEE.

Nader Mohamed received his PhD in Computer Science from the University of Nebraska-Lincoln, Lincoln, NE, USA. He was a faculty member with the Stevens Institute of Technology, Hoboken, NJ, USA, and United Arab Emirates University, Al Ain, United Arab Emirates. He also has several years of industrial experience in information technology. He is currently a Professor of Computer Science and Information Systems with the California University of Pennsylvania, California, PA, USA. His current research interests include cybersecurity, middleware, Industry 4.0, cloud and fog computing, networking, healthcare systems, and cyber-physical systems.

This paper is a revised and expanded version of a paper entitled 'Message-ferrying delay-tolerant routing in linear wireless sensor networks' presented at IEEE Global Communications Conference Exhibition & Industry Forum (IEEE Globecom 2013), Atlanta, Georgia, USA, 9 December 2013.

1 Introduction

In a considerable number of applications for wireless sensor networks (WSNs), the linear nature of the structures that are monitored, such as oil, gas, and water pipelines, roads, bridges, sea coasts, rivers, and borders, impose deployment of sensors in a linear form. Such alignment of the sensing devices can form a 'thin' or 'thick' line, each of which consists of a cross section containing one or more sensors. In addition, the sensors may be placed at a regular fixed distance, or deployed in a more random fashion, such as throwing them from a low flying airplane along the linear structure or area that is being monitored. In a previous paper (Jawhar et al., 2011), we introduced classification of resulting linear sensor networks (LSNs) from a hierarchical and topological point of view.

In this paper, we present a framework for ferry-based LSNs (FLSNs). Two types of nodes are defined: the sensor node (SN) and the ferry. The SNs are placed at regular intervals between two sinks. The FLSN system provides added flexibility in the design of the network to satisfy application requirements by not requiring the distance between each of the two consecutive SNs to be smaller than the communication range of the SNs. This is the case, most of the time as a classic multihop approach for providing SN-to-sink connectivity is not used.

This flexibility allows the transmission range of the SNs to be reduced to a minimal value, resulting in considerable sensor energy savings, elimination of transmission interference and the hidden terminal problem, and thereby increased network lifetime. In addition to eliminating the multihop overhead, the use of a ferry also addresses the problem of disproportionate node usage near the sink. Since sensors do not need to form a connected network and can use a reduced transmission range, designers only have to worry about the sensing aspect of the network during deployment. This eliminates the need to add nodes just to keep the data transfer feasible. Furthermore, research has shown that this approach increases the capacity (Chandrasekaran et al., 2003), helps in sensor calibration and failure detection, and also provides added security support. In this model, connectivity between the SNs and the sinks is provided by using a ferry, which moves back and forth between the sinks and collects the sensing data from the SNs as it comes within range of each sensor node. The ferry can also perform other functions such as data aggregation, scheduling, sensor operating system and software configuration, programming, and updating. This is the case, since the ferry is also able to transport data and programs from the sinks to the SNs.

Since end-to-end delay for the transmitted data is a parameter that is significantly affected by the movement of the ferry and the length of the network, different options to reduce this application-related parameter are considered and analysed. In addition, the resulting ferry buffer size requirements are also discussed.

The rest of the paper is organised as follows. Section 2 overviews some related work. Section 3 offers applications and motivations for FLSNs. Section 4 proposes the network model and ferry movement algorithms. Section V provides energy analysis, comparing the FLSN and multihop models. Section 6 shows additional analysis and a discussion involving energy consumption, and various models to reduce end- to-end delay. Section 7 presents simulation

results, which evaluates the performance of the proposed algorithms, and Section 8 concludes the paper.

2 Related work

An architecture with multiple data *MULEs* that are used for data collection in WSNs is presented in Jea et al. (2005). In this model, the MULEs are set in motion along straight parallel lines, in a field with randomly deployed sensors. This divides the field into parallel regions of two types, depending on whether they have sensors which are in the range of a MULE or not. Sensors in range of a MULE use direct transmission to communicate data to the MULE when it is in range. Sensors which are in the regions that fall out of range of any MULEs can transmit their data, using a multihop approach, to one of the two nearest MULEs in the two adjacent regions on either side. These sensors are labelled as 'shareable nodes', and a load balancing strategy is used to divide these nodes equally between the two nearest MULEs.

In Zhao and Ammar (2003), the authors introduce a message ferrying scheme which uses a mobile ferry to provide communication between sensor nodes in a highly-partitioned ad hoc network. In Zhao et al. (2004), the authors present an extension of the ferry scheme. They determine that the mobility of the ferry can be *task-oriented*, where its route is determined for non-messaging reasons such as a campus bus, or it can be *message-oriented*, where ferry mobility is specifically designed to improve messaging performance. In Zhao et al. (2005), the authors extend the model to multiple ferries with an emphasis on designing ferry routes. Later in Tariq et al. (2006), the ferry model is further extended to sparse ad hoc networks with mobile nodes. In Polat et al. (2011), the message ferrying and the concept of a dominating set (Wu et al., 2002) were combined to provide a framework for delay tolerant routing in mobile ad hoc networks (MANETs). Additional research that presents using mobile sinks or ferries for data collection from WSNs is presented in Oualhaj et al. (2015), Eyadeh and Amerah (2018), Zhou et al. (2017) and Zema et al. (2015). A survey on data collection protocols in WSNs using mobile data collectors is provided in Mukherjee et al. (2015). Another survey is presented for deploying mobile sinks in WSNs (Khoufi et al., 2017). In Anagha and Binu (2015), a data collection strategy in WSNs using Rendezvous points is presented. In Xie et al. (2015), a mobile platform for wireless charging is presented. Other works on using mobile elements for data collection in WSNs are also done in Zhao et al. (2015), Vanarotti et al. (2016), Kim et al. (2016) and Kartha and Jacob (2015).

In addition, some work has been done in designing and analysing linear networks. However, it is primarily for wired or general wireless networks that are not specific to sensor applications. For example, linear wired or wireless networks are used for connecting emergency telephones on highways. Wireless mesh routers can also be installed in linear infrastructures along downtown streets, to enable mobile users to access the internet (Akyildiz and Wang, 2005). Another example is using linear wired sensor networks for monitoring and controlling pipelines. Sensors are distributed along the long pipeline to report the status of the pipeline regarding temperature, pressure, flow speed, etc. Such a network is usually constructed using copper or fibre optic cable (Mohamed et al., 2008). Wireless technology has been proposed to enhance the connectivity and reliability of long wired networks for pipelines (Mohamed and Jawhar, 2008). There has been some work in studying the performance of linear wireless networks, and one example is given in MomCilvic and Squillante (2008).

Some researchers studied the characteristics of one-dimensional ad hoc networks. Diggavi et al. (2005) studied the characteristic of wireless capacity with the existence of mobility in one-dimension. Ghasemi and Nader-Esfahani (2006) provided an approximation formula for the connectivity probability of one-dimensional ad hoc wireless networks. Miorandi and Altman (2006) analysed the connectivity issue in one-dimensional ad hoc networks, using a queuing theory approach.

All these algorithms are designed for multi-dimensional WSNs or ad hoc network architectures, or use a multihop strategy for LSNs. In addition, some of the above work focuses on the routing of messages in multi-dimensional DTN networks (Li and Bartos, 2014; Jeong and Kang, 2013). Our work differs from these approaches, since it is designed for sensor networks with linear topologies. The existing multi-dimensional works do not take advantage of the predictable linearity of the WSN, and therefore, are not optimised for such characteristics.

3 FLSN model: motivation and applications

3.1 The FLSN model versus the multihop routing approach

One of the main advantages of the FLSN model is the considerable savings in energy consumption by the SNs and the resulting increase in their lifetime, as well as that of the entire network. In order to justify this motivation to use the FLSN model, we provide an analysis of the energy consumption by each SN node using the FLSN model, and compare it to the multihop model with and without data aggregation.

We assume that the ferry is a special node with considerably more energy resources than the SN nodes. Therefore, we focus on the energy consumption at the SN nodes, which have a fixed communication range R_c. Therefore, the energy consumption by the i^{th} SN for transmission of data during the passing of the ferry is (Cordeiro and Agrawal, 2011):

$$E_{txi} = E_{tx-elec}(k) + E_{tx-amp}(k, d_{tx}) = E_{elec} * k + \varepsilon * k * d_{tx}^2 \quad (1)$$

where $E_{Tx\text{-}elec}$ is the transmission electronics energy consumption, $E_{Tx\text{-}amp}$ is the transmission amplifier energy consumption, k is the number of bits in the message, and d_{tx}

is the transmission distance. Let B_i be the size (in bytes) of the data buffer of node i. So, for the i^{th} SN: $k = 8 * B_i$, and $d_{tx} = R_c$. Therefore: $E_{txi} = E_{tx\text{-}elec}(8 * B_i) + E_{tx\text{-}amp}(8 * B_i, R_c)$.

So, $E_{txi} = E_{elec} * 8 * B_i + \varepsilon * 8 * B_i * R_c^2$.

Therefore, the average transmission energy used to route a byte of data from an SN to the sink in the FLSN model is given by:

$$E_{txb}^{FLSN} = E_{txi}/B_i = E_{elec} * 8 + \varepsilon * 8 * R_c^2. \quad (2)$$

Typical values $E_{tx\text{-}elec}$ and ε, as indicated in Cordeiro and Agrawal (2011), are: $E_{tx\text{-}elec} = E_{elec} = 50$ nJ/bit and $\varepsilon = 100$ pJ/bit/m^2.

In the multihop model, the SN data is routed to the nearest sink by multihopping it across the intermediate SNs towards the sink. In the data aggregation case, each SN adds its own data to the received message before sending it to the next SN towards the sink. In this model, the total energy consumption that is needed by each SN transmission to reach the sink is:

$$E_{txi} = E_{elec} * B_i * 8 + \varepsilon * B_i * 8 * (d_{tx} + 2 * R_c)^2 \quad (3)$$

where d is, as defined earlier, the length of the no-communication distance between two consecutive SNs.

However, the transmission by node i must contain the data previously stored in node $i - 1$, B_{i-1}, in addition to the data stored in its own buffer B_i. Similarly, the transmission by node $i + 1$ must contain the data previously store in node i, B_i, in addition to the data stored in its own buffer, B_{i+1}, and so on. Therefore, the total energy consumed by the transmission of data in all of the buffers of n nodes is: $E_{tot} = \sum_{i=1}^{n} E_{txi}$.

So,

$$E_{tot} = \sum_{i=1}^{n} \left(E_{elec} * \left(\sum_{k=1}^{i} 8b_k + \varepsilon * \sum_{k=1}^{i} (8b_k)(d_{tx} + 2R_c)^2 \right) \right). \quad (4)$$

Therefore, the average transmission energy used to route a byte of data from an SN to the sink in the MHA model is: $E_{txb}^{MHA} = E_{tot} \Big/ \left(\sum_{k=1}^{n} B_k \right)$.

In the case of the multihop model without data aggregation, each message is routed separately through the intermediate SNs to the nearest sink. Consequently, the energy consumption will be significantly increased, even beyond that of the multihop model with data aggregation.

On the other hand, we acknowledge that the ferry used to transport the data will incur added energy consumption. However, this ferry is assumed to be a special mobile node such as a robot, unmanned aerial vehicle (UAV), autonomous underwater vehicle (AUV), or another device, which has considerably more energy resources with the ability to do periodic recharging.

As mentioned earlier, this analysis provides motivation to use the FLSN model in situations where transported data traffic can tolerate added delay.

1 *Additional advantages of the FLSN model:* another advantage of the FLSN model over the multihop one is the significantly increased flexibility in the deployment, and the SN node density of the WSN. The WSN can have a highly sparse topology with a low geographic deployment density of the SNs. A SN does not have to be in range of any other SN, and can still have connectivity to the sink through the ferry. In addition, it may not be required to have a full sensing coverage of the entire geographic area. This is a highly desirable feature in some applications, especially in cases where deploying a large number of SNs to cover a given geographic area is prohibitively expensive due to a higher cost of the individual SNs needed by some applications. Such applications can include ones where the SN must be equipped with special expensive sensing technology, security, special communication, and environmental protection provisioning (e.g., underwater SNs, video SNs, etc.)

2 *Types of data traffic that can tolerate delay:* naturally, the FLSN model is not intended to support all of the possible data traffic types in sensor networks. It is mainly intended to support delay-tolerant data traffic. Such data traffic includes but is not limited to, the following:

   a store-and-forward sensing: this includes archival/offline storage, which can be generated in applications such as habitat monitoring, seismic activities and volcano monitoring, etc.

   b store-and-forward pictures: this type of data can be generated in applications such as military surveillance, and environmental surveillance pictures

   c store-and-forward videos: this type of data traffic can be generated in disaster recovery, and long-term military surveillance.

3.2 Assumptions and Limitations of the FLSN model

The FLSN model is not intended to be used for applications where the data traffic that is being collected by the sensors is for real-time monitoring and control. In addition, the FLSN model is not useful in environments that do not tolerate or allow a moving ferry such as a UAV, AUV, or a moving robot. In such cases, the multihop approach can be used for routing the WSN data, and several protocols have already been introduced for such applications.

3.3 FLSN applications

In order to further motivate our work, some potential applications for linear sensor networks are listed below:

- *Oil, gas, and water pipeline monitoring:* a ferry such as a robot can be used to collect data from sensors along the pipeline.

- *Unmanned aerial vehicle (UAV)-based WSN monitoring:* a UAV can be used to collect data from on-ground sensors in a WSN. One option is to have a larger WSN arranged into clusters and the UAV can collect the data from cluster head sensors, which have gathered from sensors in their individual clusters.
- *Autonomous underwater vehicle (AUV)-based WSN monitoring:* an AUV can travel underwater and collect data from the sensor nodes or cluster head nodes of an underwater WSN. One such application is underwater pipeline monitoring. In this case, the AUV can move along the LSN and collect monitored data from the sensor nodes along the pipeline (Jawhar et al., 2013).
- *Railroad/subway monitoring:* this is another area where LSNs can be used to monitor such linear structures. In this case, a ferry such as a robot or a low flying UAV can be used to collect data from the sensor nodes.
- *Other LSN networks:* additional LSN networks where a ferry can be used to collect data are: monitoring of AC powerlines, road monitoring, border monitoring, and river and sea-cost monitoring (Jawhar et al., 2011).

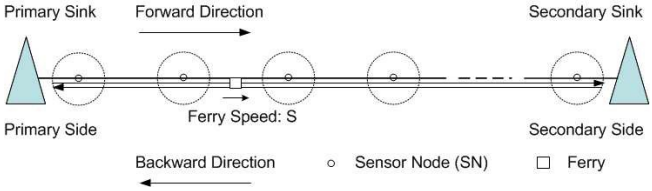
There are many reasons why a new framework and architecture are needed for different categories of LSNs. These include: increased routing efficiency, increased network reliability, better handling of node heterogeneity, improved location management algorithms, and increased network robustness (Jawhar et al., 2011). The speed of the ferry is expected to change between different FLSN applications. Such speed might range from 0.1–1 m/s for a slowly moving robot, on one end, to 1–100 m/s for a rapidly moving UAV. In addition, we realise that the movement of the ferry might cause some physical layer errors, but we do not focus on this aspect in our paper. We assume that these challenges will be managed and handled by underlying physical layer protocols. We are mostly concerned with various ferry movement strategies and their effects on the performance parameters in the data collection process. The ferry speeds used in our simulation are mainly chosen to illustrate the operation of the algorithms and do not cover all possible ranges that can be adopted under various network conditions.

4 Network model

Figure 1 shows an illustration of the network model. It consists of the following nodes:

- *Sensor node (SN):* the sensor nodes are located linearly along the pipeline with equal distances separating them, and they are assumed to be stationary. They are not required to be able to communicate with each other. In fact, in the FLSN model, they are generally expected to be out of range of each other. This provides considerable flexibility that is not available in a model that uses multihop routing in order to transmit the information from the individual SNs to the sink. As the ferry comes within range of each SN node, the latter transmits the sensed data saved in its buffer.
- *Message ferry (MF):* the ferry is a mobile node, which is assumed to have significantly more capabilities in terms of processing, communication, and storage than the SN nodes. It travels along the LSN and collects the buffered data from each SN when it comes within its communication range. In some applications of FLSN networks, the ferry could also have the capability of delivering control, programming, and configuration information to the SN nodes.
- *The sink:* sink nodes are assumed to exist at both ends of the LSN or LSN segment. When the ferry arrives at the sink node, it uploads its collected data to the sink. Then, it turns around and proceeds to move in the other direction to collect newly acquired data from the SN nodes.

Figure 1 The FLSN model (see online version for colours)



The following parameters are used in our model:

- L_n: this is the length of the LSN or LSN segment.
- n: this is the number of SNs in the LSN or LSN segment.
- d: this is the distance separating two consecutive SN nodes.
- L: the length of the data that is transmitted from the SN to the ferry in bytes.
- R: this is the data rate that is used by the SN to transmit its data to the ferry.
- R_c: this is the communication range of an SN. It is assumed to be a unit disk circle. While the actual physical range might not be a perfect circle depending on the actual environment around the SN node, we believe this is a reasonable assumption for our model, which mainly focuses on the operation of various ferry movement algorithms and their effect on performance parameters in the FLSN network.

The model has the following characteristics:

- All of the nodes are assumed to have the same message generation rate M messages/sec.
- The messages have a time-out timer. When the timer for a certain message expires, the message is discarded.
- The ferry is assumed to have a message buffer with an unlimited size. This is a reasonable assumption since

the ferry is generally a node, such as a robot or a UAV, with considerably more capabilities than a SN.

The segmented LSN model:

It is important to note that the LSN can be divided into multiple segments with sinks at the end of each segment. Individual ferries can service each LSN segment. While this is an extended model that can suite some applications, we mainly focus on the algorithms used by one ferry to service one LSN or LSN segment with sinks at each side.

4.1 Supported types of data

Our model allows both best effort (BE) as well as priority traffic. Examples of BE traffic include periodic measurement of certain monitoring parameters to keep a historic log that can be analysed later, leading to certain decisions such as maintenance schedules, expansions to improve performance or precision, and so on. Examples of priority traffic include data that is generated due to an emergency event such as pipeline leakage, unacceptable vibrations, detection of a fire, sabotage, and so on. BE traffic can tolerate higher delays while priority traffic is more delay sensitive and have more stringent quality of service (QoS) requirements.

4.2 Constant speed ferry

The model we use in this paper is illustrated in Figure 1. In this model, we define the *primary edge* as the edge where the ferry starts moving. In the figure, this is the one on the left hand side. The opposite edge is defined as the *secondary edge*. This is the one on the right hand side of our figure. In addition, a sink is located at each end of the LSN. We define the *primary sink* and *secondary sink* as the ones located at the primary and secondary sides. In the constant speed ferry (CSF) algorithm, the ferry starts moving from the primary side to the secondary side at a constant speed. When it comes within the range of each of the SN nodes, it collects data from it. When the ferry arrives at the sink, located in the direction where it is moving it uploads its data to it, and empties its buffer. Then, it turns around and starts moving in the opposite direction. It collects data form each SN as it comes within its range, till it reaches the other sink, where it uploads the collected data. This constitutes a *cycle*. Then, the ferry turns around and restarts another cycle and so on.

1 *Maximum and average message delay:* in CSF model, the ferry traverses the LSN from the primary side to the secondary side at a constant speed, s_c. As it passes within range of the SNs it collects data that was accumulated in their buffers. In our analysis, we focus on the delay experienced by the data messages due to the ferry movement algorithm. Consequently, a data message experiences a minimum delay of 0 when the SN is the nearest one to the target sink. On the other hand, it experiences a maximum delay when the SN is farthest from the target sink on the opposite side. In this case, the maximum delay would be $T_{\max}^{CSF} = \frac{(n+1)d}{s}$.

This leads to an average delay of:

$$T_{avg}^{CSF} = \frac{(n+1)d}{2s_c} \tag{5}$$

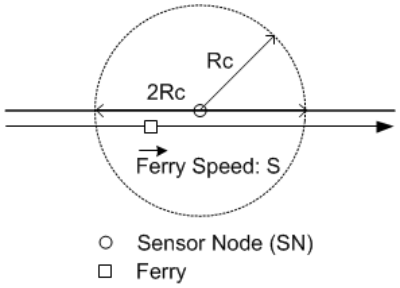
2 *Ferry speed:* in order to not have any dropped messages caused by a delay beyond the message time-out time, we must have $\frac{nd}{s} \le T$, and consequently the following relationship must be satisfied:

$$s_c \ge \frac{nd}{T} \tag{6}$$

3 *Node-ferry data exchange:* we would like to develop a formula that relates the maximum ferry speed with the amount of data that needs to be collected from an SN. As the ferry moves along the LSN, it is able to collect data from an SN only when it is within its range. This is illustrated in Figure 2. The parameters that are involved in this process are: R, R_c, s_c, and L. We designate the time when the ferry is within range of the SN as t_{fp}, which can be calculated as $t_{fp} = \frac{2R_c}{s}$. In order for the ferry to be able to collect all of the data in the SN buffer, designate by L bytes, we must have: $L \le \frac{t_{fp}R}{8}$. Consequently, the maximum speed that the ferry can have in order to be able to exchange L bytes of data from the SN node is given by the following formula:

$$s_c = \frac{R_c R}{4L} \tag{7}$$

Figure 2 Ferry and SN data exchange



4 *Ferry buffer size:* in the case where the ferry has a limited buffer size, we calculate the minimum ferry buffer requirements in order to not have any dropped messages due to buffer over flow. In the worst case, the ferry buffer must be able to accommodate $B_f^{CSF} = nL$ bytes of message data. Therefore, the minimum size of the ferry buffer for the CSF algorithm is: $B_f^{CSF} = nL$ bytes. Consequently, if we replace L with its previously derived expression, we get:

$$B_f^{CSF} = \frac{nR_c R}{4s_c} \tag{8}$$

5 *CSF with different data types:* with respect to the type of data that is collected by the ferry, we can consider two possibilities: the *homogeneous data* case, and the *heterogeneous data* case. When the ferry comes within range of an SN, it collects as much data as possible. If the data is homogeneous, then the ferry will simply collect data from the SN buffer until it goes out of range. In the heterogeneous data case, where the data consists of *priority* and *best effort (BE)* traffic, then the ferry collects as much priority data as is available in the buffer. If the ferry is still within range, then it will take the remaining time to collect BE data from the SN. In this case, the BE traffic might experience *starvation*, due to the fact that it is never collected. This can happen if there are always large amounts of priority traffic. It is the responsibility of the SN node to manage this process and prevent such starvation. Two strategies could be used in this case:

   a allocate a certain percentage of the in-range time of the ferry for the BE traffic

   b designate a time-out-timer for each block of BE data, and if the timer expires, the status of the corresponding block can be changed to priority traffic, which eventually leads to its collection by the ferry.

4.3 Variable speed ferry

The second strategy for ferry movement is the employed by the Variable speed ferry (VSF) algorithm. It is illustrated in Figure 3. In this case, the ferry is assumed to be able to have two different speeds:

- *Ferry speed while in communication (sc):* this is the speed of the ferry while it is within the communication range (R_c) of the SN. This interval is named as *C (communication)-interval*. This speed is usually smaller than the two speeds. It is determined in a way that allows the communication of an acceptable average amount of data from the SN to the ferry. This calculation should take into consideration the SN communication range (R_c), the SN data rate (R), the average size of the data in the SN buffer (L_{avg}), and the acceptable end-to-end delay tolerable by the data messages.
- *Ferry speed while no communication is taking place (s_{nc}):* this is the speed of the ferry while it is traversing the distance between two SN and is not in communication range with either one. This speed is typically greater than or equal to the s_c ($s_{nc} \geq s_c$). This is the case in order to reduce end-to-end delay, without affecting the amount of data that is collected from each SN. The speed of the ferry and its ability to adjust is heavily dependent on the environment (e.g., terrestrial, underwater, or airborne) and the application involved.

Figure 3 Illustration of VSF operation

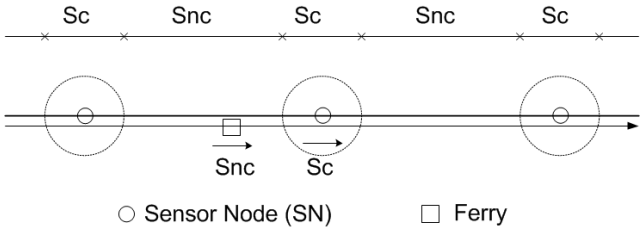
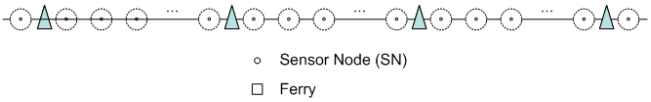


Figure 4 Reduction of end-to-end delay by increasing the number of segments (see online version for colours)



The maximum delay of a message in the VSF algorithm, $t_{\max}^{VSF}$, takes place with the ferry is on the other side of the target sink. In this case, the total delay is the summation of the delays in the C-intervals (intervals when the ferry is able to communication with the SN) and the delays in the *NC (no communication)-intervals* (intervals when the ferry is not in communication with the SN). The *C (communication)-intervals* and NC-intervals are show in Figure 3. Each C-interval has a length of $2R_c$. The sum of C-intervals is $(n-1)2R_c$. Each NC-interval has a length of $(n-1)2R_c$. The sum of NC-intervals is $(n-1)(d-2R_c)$. Consequently, the maximum delay for the VSF algorithm: $t_{\max}^{VSF} = \frac{(n-1)(d-2R_c)}{s_{nc}} + \frac{(n-1)2R_c}{s_c}$. Therefore,

$$t_{\max}^{VSF} = (n-1)\left(\frac{(d-2R_c)}{s_{nc}} + \frac{2R_c}{s_c}\right). \tag{9}$$

1 *Ferry buffer size:* it would be useful to determine the minimum amount of buffer size for the ferry. It is easy to see that it is $B_f^{VSF} = nL$. Replacing L with its previously derived expression, we get:

$$B_f^{VSF} = \frac{nR_c R}{4s_c}. \tag{10}$$

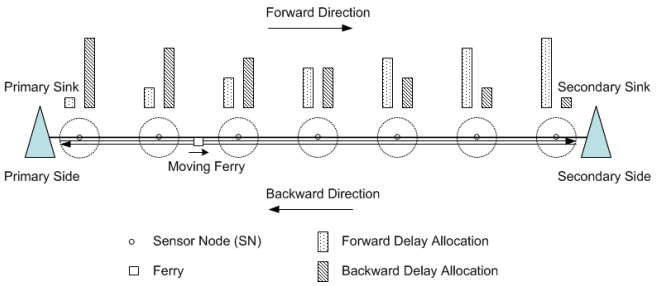
2 *VSF with homogeneous and heterogeneous data:* With regards to the collection of BE and priority traffic, the same strategy used in the CSF algorithm, can be used in VSF. This is the case, since the main difference in the two algorithms is the fact that the ferry moves at different speeds in the *NC-interval* than that in the *C-interval*. Consequently, the overall delay is reduced and this does not affect the way various types of traffic are treated during the data collection process.

4.4 Adaptable speed ferry

Depending on the application, the data collection rate for the SNs can vary. This could be due to certain external factors such as abnormal or emergency situations, which might include the detection of an intruder, pipeline or infrastructure vibrations, high pressure, or high temperature.

The SN might be required to increase its data collection rate in a distributed manner in order to provide higher accuracy to allow the system users and operators to follow an unusual situation more closely. For example, high resolution audio and/or video monitoring data might be collected. On the other hand, the decision to increase or decrease the data collection rate, and turn ON or OFF some additional standby equipment (that is required to be awakened only under such circumstances) can follow push or pull strategies. In the push case, the SNs make the decision themselves based on locally collected data. However, in the pull case, the decision is made by the NCC personnel based on local as well as regional or global system information or constraints. The *adaptable speed ferry (ASF)* model is designed to be used in such applications. It is illustrated in Figure 5, and operates in the following way.

Figure 5 Illustration of ASF-DA operation (see online version for colours)



Earlier, we defined the size of the SN buffer upon the arrival of the ferry within its range to be B_i, and the time it takes to communicate the data in the buffer from the SN to the ferry at a data rate R to be t_c. Then, we get:

$$t_c = \frac{B_i}{R}. \tag{11}$$

Therefore, we can determine the ferry speed while it is in range of the SN to be:

$$S_{ci} = \frac{2R_c}{t_c}. \tag{12}$$

Subsequently, when the ferry goes out of the range of the SN, it will reset its speed to S_{nc} to move towards the next SN. Then, it will repeat the same process upon its arrival within range of the new SN.

1 *Entrance of ferry in SN communication range:* the VSF and ASF algorithms rely on the knowledge of the ferry of its entrance within the communication range of an SN. This can be done using one of two strategies depending on the application involved:

- The ferry may be equipped with GPS capability along with the location, and communication range of the SNs. Such information can be programmed in the ferry by system administrators at the network setup. When the ferry arrives within communication range of the SN, it sends a *hello message* to the SN informing it of its arrival. Upon reception of the message, the SN responds with a reply message that contains the SN node ID, and buffer size (size of the collected data).
- The ferry can be programmed to continuously send periodic Hello messages. When the node receives the message it is required to respond with a reply message with the information mentioned earlier.

Algorithm 1 ASF-DA algorithm – initialisation of delay quota

for i = 1 to n **do**
  if *dir = forward* **then**
    $D_i^t = i * T_q$
  else
    $D_i^t = (n - i + 1) * T_q$
  end if
end for

2 *ASF with homogeneous and heterogenous data:* in the CSF and VSF algorithms, end-to-end delay experienced by the data messages is bounded since the speed of the ferry in the no- communication and communication intervals is deterministic. However, in the ASF algorithm, this is not the case. While the speed of the ferry in the no-communication interval between the SNs is fixed, its speed while in the communication range of the SN is determined based on the amount of data in the SN buffer, which is variable. Consequently, the end-to-end delay in the ASF case is unbounded, and can lead to *starvation* of the data in the remaining SNs on the way to the target sink. In order to place a bound on the delay and solve this problem, a new algorithm named *adaptable speed ferry with delay allocation (ASF-DA)*, which constitutes an extension of the ASF algorithm is defined.

When the ferry reaches a sink, the delay allocation strategy described in algorithm 1 is executed. If the ferry is moving in the forward direction, then the delay quota is allocated in a manner that is inversely proportional to the distance of the node from the target sink. Consequently, the delay quota for node i is set as $i * T_q$. This strategy allows less data messages to be downloaded from the nodes that are further from the target sink, since they would experience larger delays, and more data messages to be downloaded from the nodes that are closer to the target sink since they would be experiencing lower delays. As a result, the overall average end-to-end delay for the data messages would be reduced. When the ferry reaches the sink, it uploads

the collected data and turns around to move in the opposite direction. The same strategy is deployed but in a reversed manner. Consequently, the nodes that had the largest delay quota in the previous pass, would have the smallest quota in the next pass in the reverse direction. This provides a balance in the overall delay quota allocation for the SNs in the LSN segment.

Algorithm 2 ASF-DA algorithm – data exchange between the ferry and SN node

if $(D_i^t * R) \geq B_i^t$ **then**

$D_i^a = B_i^t / R$

$s_{ci} = 2 * R_c / D_i^a$

$D_i^r = D_i^t - D_i^a$

distributeRemainingDelay(dir, i, D_i^r)

else

$S_{ci} = 2 * 2R_c / D_i^t$

$D_i^a = D_i^t$

end if

downloadDataFromCurNode(i, D_i^a, λ)

The steps involved in the ASF-DA algorithm are shown in Algorithm 2, which works in the following manner. When the ferry comes within the range of the i^{th} SN, it sends a request message to determine the size of the data accumulated in the SN with the size of the buffer, it checks if the allocated delay for node i, D_i^t, is more that the time require to download the whole buffer at the SN's data rate R. If that is the case, it calculates the actual delay time that is required, and determines the associated ferry speed. Then, it calculates the remaining delay D_i^r that is left from the delay quota for node i in order to distribute it to the delay quotas of the remaining SNs on the way to the target sink using a *distributeRemainingDelay*(dir, i, D_i^r) function. Otherwise, if the delay quota is less than or equal to the required amount of time to download the buffer, the ferry sets its speed according to the delay quota for node i. In both cases, the *downloadDataFromCurNode*(i, D_i^a, λ) is used to download the data from the SN node. This function is presented in Algorithm 3. The parameter λ is used to control the amount of priority and BE data that is downloaded. Specifically, if $\lambda = 0.8$, then 80% of the delay is used to download the priority traffic and the remaining 20% is used to download the BE traffic.

Algorithm 3 *downloadDataFromCurNode*$(i, delay, \lambda)$ function

$B_i^p = R * Delay * \lambda / 8$

Download B_i^p bytes from priority data buffer of i.

$B_i^b = R * Delay * (1 - \lambda) / 8$

Download B_i^b bytes from BE data buffer of i.

Algorithm 4 ARF algorithm – initialisation when ferry leaves sink

if *blackNodeCounter* = n **then**

for (i = 1; $i \leq$ n; i++) **do**

blackNodeCounter = 0

$F_{tip}^i = WHITE$

end for

end if

if (*networkStartup* = *TRUE*) *OR* (*dir* = *FORWARD*)

then

cn = 1

else

cn = n

end if

ferryMode = *NORMAL_MODE*

ferrySpeed = *NORMAL_SPEED*

4.5 Adaptive routing ferry

The adaptive routing ferry (ARF) algorithm, which is illustrated in Figure 6 works in the following manner. In each cycle, the ferry sets its mode to *NORMAL_MODE*, starts at the primary sink and moves towards the secondary sink at a constant speed named *NORMAL_SPEED*. When the ferry comes within range of an SN node, it downloads data from it. If the ferry buffer reaches a certain predetermined threshold amount ρ, the ferry switches to the *GO_FAST_TO_NEAREST_SINK* mode, changes its speed to *FERRY_MAX_SPEED*, and moves in the direction of the nearest sink. When it arrives at the sink, it downloads its buffer and turns around to go back to its previous position. Then, it resumes it path and continues its previous movement and data collection process from the SNs that have not been serviced yet in the current cycle. A *serviced-in-this-trip* flag, named F_{sit}^i is used for each node i to indicate whether it has been serviced by the ferry during the current cycle or not.

The ARF algorithm tries to minimise average end-to-end delay of the data messages by allowing the ferry to move quickly to reach the nearest sink as soon as its buffer reaches a certain threshold ρ. This threshold is a system configuration parameter set by the network administrator and varies according to the application as well as related network environment parameters such as the ferry speed range, transmission data rate, data collection rate, and

SN and ferry buffer sizes. The algorithm also tries to ensure fairness by giving each node an opportunity to be serviced in a timely manner. Furthermore, the use of *serviced-in-this-trip* flag ensures the prevention of starvation for the various SN nodes in the LSN.

Algorithm 5 ARF algorithm – performed by the ferry at an intermediate SN

Let i be the ID of the current node

Let F_{sit}^{i} be serviced-in-this-pass flag. F_{sit}^{i} is set to *BLACK* when node i is serviced in the current pass, and set to *WHITE* otherwise.

$D_i^t = 2 * R_c / ferrySpeed$

$downloadDataFromCurNode(i, D_i^t, \lambda)$

if $F_{sit}^{i} = WHITE$ **then**

  if $curFerryBufferSize \geq \rho$ **then**

    /* Ferry load has reached the critical threshold value and the ferry must go to the nearest sink at its maximum speed to deliver the data. */

    $dir = getDirToNearestSink()$

    $ferrySpeed = FERRY\_MAX\_SPEED$

    $ferryMode = GO\_FAST\_TO\_NEAREST\_SINK$

  end if

  $F_{sit}^{i} = BLACK$

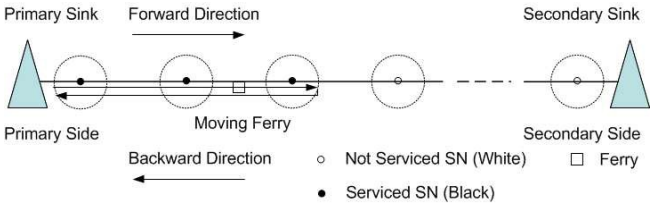
  $blackNodeCounter = blackNodeCounter + 1$

end if

$i = getNextNodeInDir(i, dir)$

Go to i node at the current *ferrySpeed*

Figure 6 Illustration of ARF operation (see online version for colours)



The ARF algorithm involves two parts: The initialisation part presented in Algorithm 4, and the ferry algorithm at an intermediate SN part, presented in Algorithm 5. The initialisation algorithm is executed at the start of the network operation, as well as when the ferry reaches a sink and turns around to go in the opposite direction toward the other sink, starting a new *trip*. However, initialisation of the service flags is only done if the black nodes' counter has reached n, indicating that all nodes in the segment have been serviced, and a new ferry pass is started. Upon initialisation, all of the F_{sit} flags are set to *WHITE*, indicating that the nodes have not yet been serviced. If the network is just starting its operation, or the direction is *FORWARD*, then the current node variable c_n is set to 1, otherwise, the direction must be *BACKWARD*, and *cn* is set to n. Then the *ferryMode* variable is set to *NORMAL_MODE* and the *ferrySpeed* variable is set to *NORMAL_SPEED*.

The ferry algorithm at an intermediate SN, i, works in the following manner. If the ferry is in normal mode, then first, the total delay for node i is calculated using the ferry speed and the SN communication distance, R_c. Then, the ferry calls the $downloadDataFromCurNode(i, D_i^t, \lambda)$. Afterwards, if the F_i^{sip} is *WHITE*, the current ferry buffer is checked. If it exceeds the critical threshold value ρ, then the ferry determines the direction of the nearest sink, using the function *getDirToNearestSink*(), and goes there after setting its speed to *FERRY_MAX_SPEED*, and its mode to *GO_FAST _TO_NEAREST_SINK*. Afterwards, the F_i^{sip} is set to *BLACK*, and the *blackNodeCounter* is incremented by 1. It is important to note that data is downloaded from the SN using the $downloadDataFromCurNode(i, D_i^t, \lambda)$, regardless of the F_i^{sip} flag status, albeit at different ferry speeds and download times.

1 *ARF with homogeneous and heterogenous data:* in this algorithm, the handling of homogeneous and heterogeneous traffic is left up to the internal processing of the node itself. The node contains two buffers, one for BE and another for priority traffic. When the ferry arrives, the majority of the download delay quota can be shared by the two types of traffic, according to the variable λ described earlier in the ASF algorithm; here $\lambda = 0.8$, then 80% of the quota is used for priority traffic, and the remaining portion is used by the BE traffic. Otherwise, more sophisticated mechanisms can also be deployed, which may give all of the quota to the priority traffic if it exists, and may promote the BE traffic to a priority status after the expiration of time-out time (TOT) for each data block, as to prevent BE traffic starvation.

5 Additional analysis and discussion

5.1 Options to reduce end-to-end delay

As in previous sections, a very critical parameter in FLSNs is end-to-end delay. In this section, we discuss and analyse the advantages and disadvantages of the various extensions, which can lead to reduced end-to-end delay in the proposed basic model. Table 1 contains a summary of these options and the related characteristics. One of the most important means of reducing end-to-end delay in FLSNs is by increasing the ferry speed. In order to analyse this option, we have to consider the CSF and VSF models separately.

1 *Increasing s_c in the CSF model:* in the CSF model, the ferry speed s_c is constant. In order to reduce message end-to- end delay, we can increase the value of s_c. However, this is limited by two factors:

a the *physical and mechanical nature* of the ferry. This option heavily depends on the application, which will have an upper limit on the physical speed of the ferry. Let us define this limit as S_P.

b The *ferry-node data exchange relationship* must satisfy equation (7). Specifically, given fixed values for the SN communication range R_c, and SN data rate R, increasing the speed of the ferry will reduce the amount of data that can be exchanged during one pass of the ferry. Therefore, depending on the requirements of the application with respect to the size of the data that must be exchanged during one pass, and the maximum amount of acceptable end-to-end delay, the maximum ferry speed that can be achieved can be determined. Let us define this value as S_E.

After determining these two maximal values, the actual ferry speed would have to be the smaller of the two. Namely: sc = minimum (S_P, S_E).

2 Increasing s_{nc} in the VSF and ASF models: as discussed earlier, in the VSF model, the ferry has two speeds, depending on its location as it is moving along the LSN, s_c and s_{nc}. Increasing speed can provide for reduced end-to-end delay. However, the value of s_c is constrained by the parameters R_c, R and L, which might be fixed for the particular SN node technology, and application that is involved. Therefore, increasing the value for s_{nc} might be a more appropriate and flexible solution to reduce the total end- to-end delay. In this case, only the maximum speed dictated by the physical and mechanical nature of the ferry, S_P, would be the limiting factor. In that case, the designers of the LSN would not have to be concerned with the maximum speed S_E set by the ferry-node data exchange relationship constraint. This is a considerable advantage of the VSF model, especially in *highly disconnected LSNs*, where distance between SNs, d, is considerably greater than the SN communication range R_c. In such situations, the ferry can move much faster once it is out of range, with an SN node to reach the communication range of the next SN node.

In the ASF model, the s_{nc} speed is fixed, while the s_c speed (named sc_i) is set according to the buffer size of the node, i, that is within the range of the ferry. Consequently, the same analysis applies with respect to increasing the s_{nc} speed in order to reduce the end-to-end delay, while the s_{ci} speed is constrained by the buffer size and cannot, otherwise, be changed.

3 *Increasing the number of sinks:* another strategy for reducing end-to-end delay is by increasing the number of sinks, and consequently increasing the number of segments and ferries (since we have one ferry per segment) for the same number of SNs and coverage area. This is illustrated in Figure 4. This process will result in shorter segments, leading to a shorter length of time for the ferry to go across each segment in both directions, which results in a lower end-to-end delay for data delivery. This option can be considered for both the CSF and VSF models. This gain in performance comes at the price of higher complexity, cost, and installation time of the communication system. This networking model might be an effective strategy depending on the application that is involved, and the criticality of the end-to-end delay parameter.

4 *Hybrid approach: increase s_{nc} and the number of sinks:* In order to decrease the end-to-end delay, the designers can increase s_{nc} in the VSF model, and can also increase the number of sinks. This approach can be useful if the desired delay cannot be achieved by only increasing the no-communication speed of the ferry, due to the physical limitations of the ferry technology that is available for a particular application. In such a case, it would be necessary to increase the number of sinks to cover a fixed distance with a given total number of SNs. This leads to more stringent physical technology requirements for the ferry to achieve higher speeds. In addition, this option would result in an increased system cost and management complexity due to the increased number of sinks.

5 *Increase the number of ferries per segment:* another option for decreasing the end-to-end delay is to increase the number of ferries that service each segment. The number of ferries per segment can also vary between different segments. For example, more ferries can be used in segments that have heavier monitoring traffic. If this option is chosen, appropriate optimisation algorithms would be required to determine the best ferry routes for a segment, given the number of ferries servicing a segment, as well as the number of SNs, the distance between them, and the traffic arrival rate of each one.

5.2 FLSNs using various sensor nodes

In Table 2, the main parameters that affect the design and performance of an FLSN network are presented for various existing sample sensor nodes. Specifically, the parameters are the transmission rate, R, and the communication range R_c. The CSF model is assumed in the table. In order to calculate the value for the exchanged data length in one pass between the ferry and a SN, L, we use equation (6) with a ferry speed of 1 m/s. We also use equation (8) to calculate the resulting ferry buffer size for n = 100 nodes. The sensor nodes presented in the table can be divided into three different categories:

1 special-purpose sensor nodes, such as the Spec.

2 generic sensor nodes such as the Rene, Mica-2, Telos and Mica-Z

3 high-bandwidth sensor nodes such as BT and Imote.

Table 1 Options to reduce end-to-end delay

Option	*Limited by*	*Cost type*	*Complexity*
Increase s_c (CSF)	Ferry physical speed, SN data comm. range, SN-ferry data exchange rate	Ferry mobility energy cost	Increased SN-ferry synchronisation and transmission complexity
Increase s_{nc} (VSF, and ASF)	Ferry physical speed, flexibility in adjusting ferry speed while communicating (s_c)	Ferry mobility energy cost	Increased ferry physical movement technology complexity
Increase number of sinks	No technical limitation	Increased total cost of sinks and ferries	Increased sink management complexity
Hybrid approach: increase s_{nc} and number of segments	Ferry physical speed limitation can be remedied by adding more segments	Ferry mobility energy cost + increased total cost of sinks and ferries	Increased ferry physical technology and increased sink management complexity

Table 2 FLSN parameters with various sensor nodes

Sensor node	*Trans. rate (R bps)*	*Trans. range (Rc m)*	*Max. ex. data length in one pass (L KB)*	*Ferry buffer size (B_f^{CSF} KB)*
Spec	100,000	15	381.0	38,100.0
Rene	10,000	30	76.2	7,620.0
Mica-2	76,000	100	1,900.0	190,000.0
Telos	250,000	125	7,812.5	781,250.0
Mica-Z	250,000	100	6,250.0	625,000.0
BT Nodes (Bluetooth)	1,000,000	100	25,000.0	2,500,000.0
Imote (Bluetooth)	1,000,000	100	25,000.0	2,500,000.0

We can clearly see that for sensors with a small transmission rate and communication range, such as the Spec sensor, the value for the exchanged data length and resulting ferry buffer size are relatively low: around 381 KB, and 38,100 KB respectively. For the general purpose sensors, such as the Mica-2, these values increase to 1,900 KB and 190,000 KB respectively. Finally, for the high bandwidth sensors such as the Imote, these values increase further to 25,000 KB, and 2,500,000 respectively. These values are intended to give a sample of the parameters when used with real sensors of different types; as they heavily depend on the application involved, which would dictate what type of sensor nodes, end-to-end delay, and resulting ferry speed that are to be used. In turn, these specified parameters lead to the buffer requirements of the ferry that will be used.

6 Simulation

6.1 Simulation setup

Simulation experiments are performed in order to verify the operation of the presented algorithms and analyse their performance under various network conditions.

The LSN is generated according to the model stated in Section 4. Based on the observations from realistic sensor nodes in Table 2, the default values of the input parameters are set in the following way. The number of sensor nodes, n, varies between 25, 50, 75, and 100. The distance between two consecutive sensor nodes d = 150 metres. The communication range of each sensor node R_c = 50 metres. The transmission rate R = 2,000 bps. The buffer size of each sensor node B = 8,000 bytes. The ferry speed in CSF s = 10 m/s. The ferry speed in communication in VSF s_c = 10 m/s. The ferry speed while there is no communication in VSF and ASF s_{nc} = 20 m/s. The delay quota in ASF Tq = 0.5 s. The NORMAL_SPEED in ARF is 10 m/s. The FERRY_MAX_SPEED in ARF is 50 m/s. The time-out value T = 1,500 s. Finally, the critical threshold in ARF is 10,000 bytes.

In the simulated algorithms, the best effort and priority traffic are generated according to an exponential distribution with average arrival rates that are generated according to a uniform distribution. The average of the best effort traffic is generated according to a uniform distribution between 1 and 3 bytes per second. The average of the priority traffic is generated according to a uniform distribution between 8 and 16 bytes per second. The lambda is set to 0.8.

In order to evaluate the performance of the algorithms, the following two metrics are used:

1 the delivery ratio of successful packets to the total number of packets generated by the SNs

2 the average end-to-end delay experienced by the successfully delivered packets.

The simulation is intended to evaluate the effect of the various network conditions on these two important parameters. The experiments were run for sufficient time, so that the performance of the algorithms better reflected in the results.

6.2 simulation results of CSF

The simulation results for the CSF algorithm are presented in Figures 7(a), 7(b) and 7(c). In the figures, the delivery ratio is measured as the buffer size, time-out value, and ferry speed are increased. In Figures 7(d), 7(e) and 7(f) the average delay is measured as the same parameters are also increased. We can see that for the CSF case, the delivery ratio is higher and average delay is lower when the number of nodes is small. On the other hand, an increase in the buffer size and time out value produce higher delivery ratio with increased average delay.

Figure 7(a) shows that the delivery ratio increases when the SN buffer size increases. This is due to the fact that a larger buffer size causes a fewer number of packets to be dropped due to buffer overflow. It is also observed that the impact of the buffer size is higher when the number of nodes is small. Figure 7(d) shows that an increase in the buffer size leads to a slightly higher average delay. The reason for this is that a sorted queue is used to keep the packets in the SNs. Consequently, a higher SN buffer size means that the collected packets are generated over longer periods of time.

When Figures 7(b) and 7(e) are examined, we see that the delivery ratio and average delay increase when the time-out value is increased. This is due to the fact that longer time-out values mean that more messages will be delivered before they are dropped due to time-out timer expiration. However, this causes the average delay to increase as well. It can also be seen that the delivery ratio and average delay increase rapidly when the time-out value increase. But, they stay relatively constant after some critical time-out value. These critical values are about 1,000, 1,500, 2,500, and 3,500 in both figures depending on the number of nodes. This is due to the fact that the ferry takes about this much time to go from the primary sink to the secondary sink.

The results for the average delivery ratio and average delay as the ferry speed increases, are shown in Figure 7(c) and Figure 7(f) respectively. An interesting observation can be made in that the delivery ratio increases initially with the ferry speed. However, after some critical speed, the delivery ratio starts to decrease. The reason for this is that as the ferry speed increases, more packets are delivered to the sink before their time-out timer expires. However, if the ferry speed is too large, then less packets are able to be collected from the SNs by the ferry. Consequently, these uncollected packets end up timing- out inside the SN node buffer thereby negatively affecting the average delivery ratio. On the other hand, it is observed in Figure 7(f) that the average delay goes down as the ferry speed is increased. This is due to the fact that the packets collected from the SNs are delivered more rapidly to the sink.

6.3 Simulation results of VSF

Figure 8 shows the results for the simulation of the VSF algorithm as different parameters are changed. Figures 8(a) and 8(d) show that the delivery ratio and average delay increase as the sensor buffer size increases. Figures 8(b) and 8(e) show that the delivery ratio and average delay also increase as the time-out parameter increases. We observe that as the time-out increases beyond a critical value, increase rates of the delivery ratio and average delay become very small. This critical value increases as the number of sensor nodes, n, increases from 25 to 100 nodes.

In Figures 8(c) and 8(f), we see that the delivery ratio increases and the average delay decrease as the non-communication ferry speed increases. This is due to the fact that the ferry is able to reach the sinks more rapidly. Consequently, more packets are able to deliver with shorter delay and before timing out, while the ferry is still able to collect a higher number of packets from the SNs at the lower speed.

6.4 Simulation results of ASF

Figure 9 shows the results for the ASF algorithm simulations. Figures 9(a) and 9(d) show that the delivery ratio and average delay go up slightly as the buffer size increases. We also observe that the effect of the buffer size on the delivery ratio is reduced when the number of nodes increases. Figure 9(b) and 9(e) show that the delivery ratio and average delay increase as the time-out value increases. In addition, a trend which is similar to that in the CSF case is also observed here.

The simulation results for the ASF delivery ratio and average delay as the delay quota Tq increases are shown in Figs. 9(c) and 9(f) respectively. When $n = 25$, it is observed that the delivery ratio peaks at a critical value of $Tq = 0.5$. This is due to the fact that when delay quota is large, there is enough time for the SN nodes to transfer their packets to the ferry. If Tq is very small, then nodes do not have enough time to download their data so that a lot of their packets are not picked up by the ferry and ultimately time-out, resulting in a low delivery ratio. On the other hand, when Tq is too large, the ferry spends too much time at the early nodes and the packets in the latter nodes also time-out, resulting in a low delivery ratio as well. Consequently, for a particular network size, there is an optimal value for Tq. However, our simulation is limited and does not reflect this fact in the other three curves. Figure 9(f) shows that the average delay increases when the delay quota increases. This pattern is reasonable since an increase in the delay quota should lead to an overall increase in the average delay experienced by the successfully delivered packets.

Figure 7 CSF simulation results, (a) effects of buffer size on delivery ratio (b) effects of time-out value on delivery ratio (c) effects of ferry speed on delivery ratio (d) effects of buffer size on average delay (e) effects of time-out value on average delay (f) effects of ferry speed on average delay (see online version for colours)

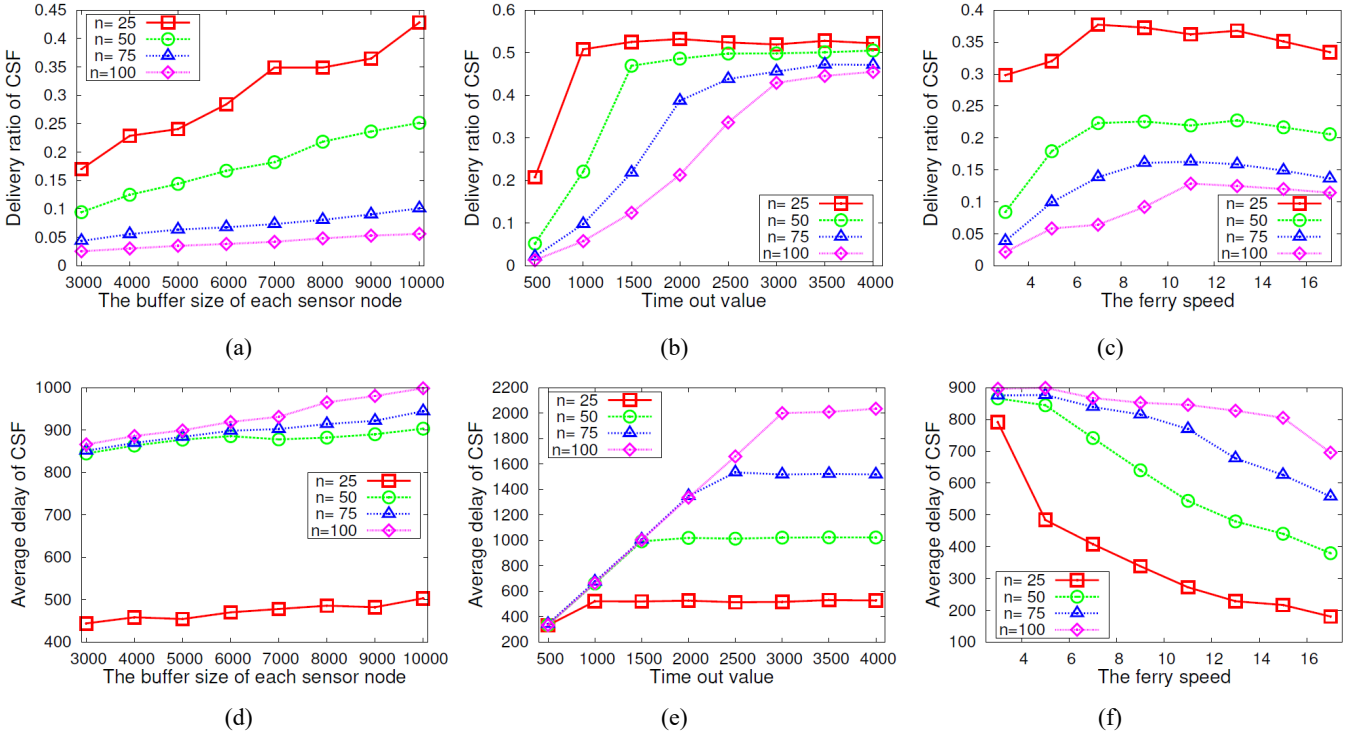


Figure 8 VSF simulation results, (a) effects of buffer size on delivery ratio (b) effects of time-out value on delivery ratio (c) effects of non-com ferry speed on delivery ratio (d) effects of buffer size on average delay (e) effects of time-out value on average delay (f) effects of non-com ferry speed on average delay (see online version for colours)

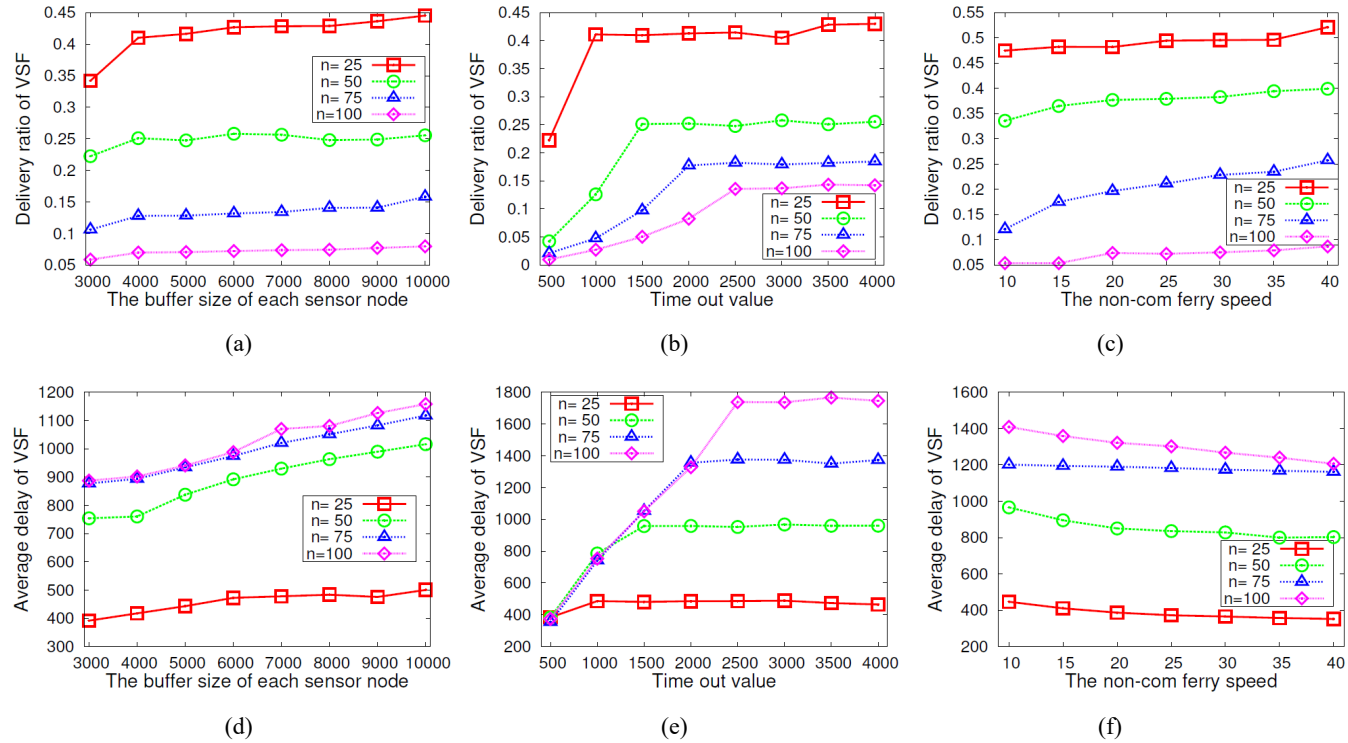


Figure 9 ASF simulation results, (a) effects of buffer size on delivery ratio (b) effects of time-out value on delivery ratio (c) effects of delay quota on delivery ratio (d) effects of buffer size on average delay (e) effects of time-out value on average delay (f) effects of delay quota on average delay (see online version for colours)

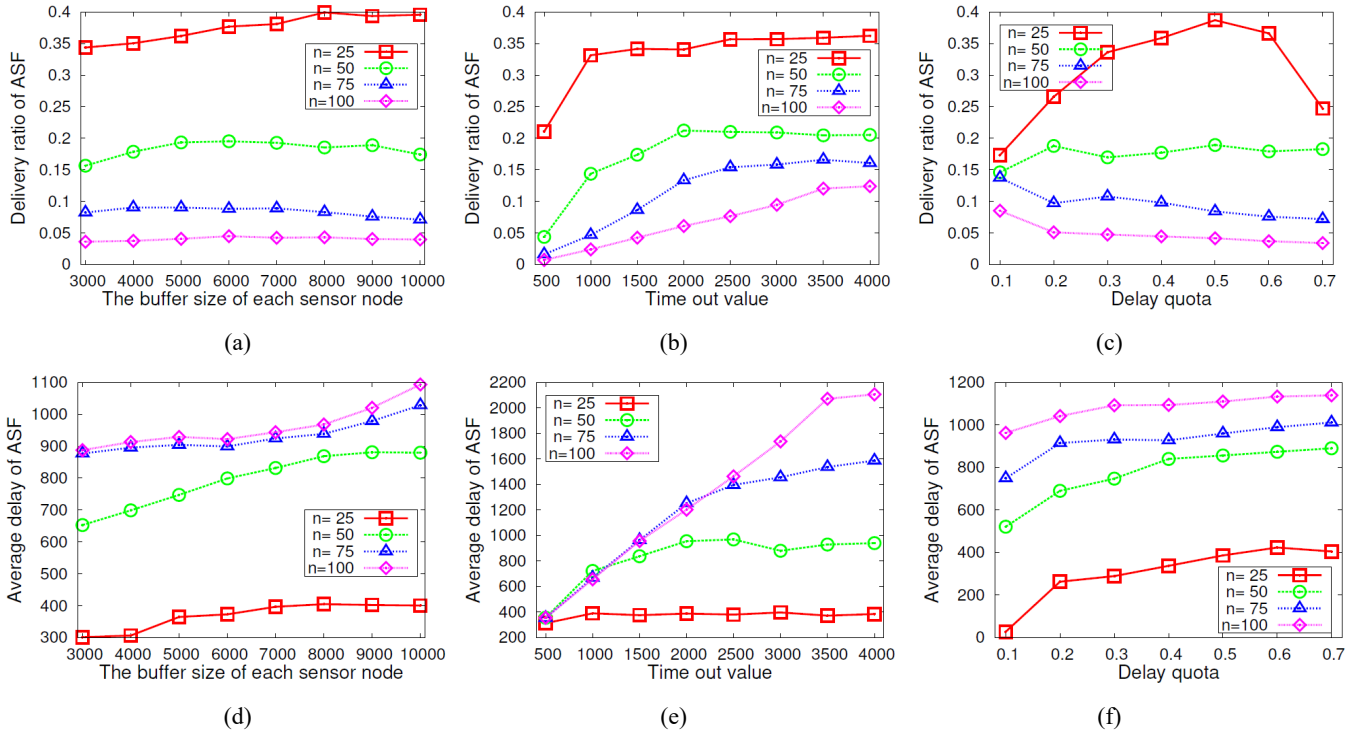
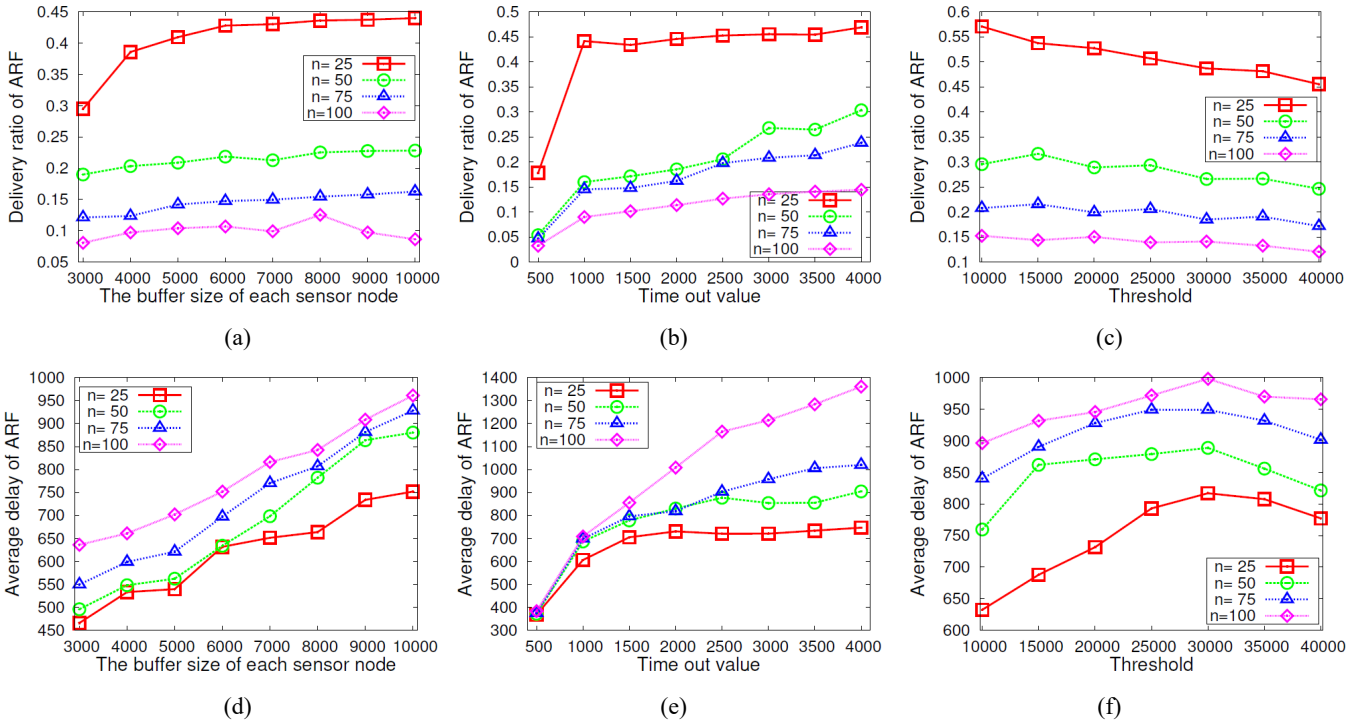


Figure 10 ARF simulation results, (a) effects of buffer size on delivery ratio (b) effects of time-out value on delivery ratio (c) effects of buffer threshold on delivery ratio (d) effects of buffer size on average delay (e) effects of time-out value on average delay (f) effects of buffer threshold on average delay (see online version for colours)



6.5 Simulation results of ARF

The simulation results for the ARF algorithm are presented in Figure 10. In Figures 10(a) and 10(d), the delivery ratio and average delay are measured as the SN buffer size increases respectively. We note in Figure 10(d), an interesting phenomenon that has not been seen before: the average delay increases dramatically as the buffer size increases. The main reason behind this is that the packets in each sensor node are sorted in a queue, thus, when the buffer size increases, the ARF would collect more packets that were generated a long time ago. Figures 10(b) and 10(e) show the results of the effect of the time-out value on the delivery ratio and the average delay, respectively, of ARF. The observations are similar to the previous ones.

In Figure 10(c), when the buffer threshold increases, the delivery ratio decreases. This is due to the fact that, when the threshold increases, the time period for the ferry to travel in the NORMAL_SPEED becomes longer, which may impact the delivery ratio. In Figure 10(f), the average delay of ARF first goes up, and then goes down with the increasing threshold. The main reason is that, when the threshold is large enough, it would collect a large number of newly-generated packets from some sensor nodes, which reduces the average delay of successful packets.

6.6 Summary

In summary, we can identify the following observations:

- In all proposed algorithms, when the SN buffer size is increased, the delivery ratio increases as well. This is reasonable, since an increase in the buffer size leads to a lower probability of packets being dropped due to buffer overflow. Consequently, more packets are able to be delivered successfully to the sink. In addition, in our implementation, the packets in an SN are stored in a sorted queue based on their generation time, and the ferry gives higher priority to the packets with a smaller generation time. Consequently, we see that the average delay also increases when the buffer size increases.
- As the time-out time increases, the delivery ratio and average delay increase as well. In addition, it was discovered that there are some critical time-out values which correspond to the number of SN nodes and the time taken by the ferry to reach the sinks.
- Several parameters such as the ferry speed and no-comm ferry speed in VSF algorithm as well as the delay quota in ASF algorithm, and the buffer threshold in the ARF algorithm have significant impact over the efficient operation of the network. Consequently, they must be chosen carefully to optimise the performance of the data collection process.
- Our research and simulation experiments introduced and evaluated new algorithms for data collection in linear sensor networks. However, we realise that there are still many challenges and issues that deserve further investigation, which can be considered for future research. Our aim in this paper was to provide insight into this important area of research and provide possible strategies that can be adopted in data collection systems.

7 Conclusions

LSNs are WSNs that have a linear topology due to the linearity of various areas and structures that are monitored. In this paper, we introduced an FLSN model, which relies on a mobile ferry node for data collection from the SNs in an LSN. The ferry collects data that is stored in the SN data buffers when it comes within communication range of each one and delivers it to the sink at the end of the LSN or LSN segment. We presented four different ferry movement algorithms that can be used for this process. Furthermore, we studied the performance of each of the algorithms by measuring the effect of different network parameters such as SN buffer size, packet time-out value, and ferry speed on important network parameters such as the packet delivery ratio and average delay. In addition, different design options for reducing end-to-end delay are considered, along with a discussion of the application of FLSNs using various existing sensor nodes. We believe that this work paves a way for additional research that is needed. For example, more research can be done to use multiple ferries to service an LSN segment. Moreover, issues and challenges related to ferry route selection and data collection synchronisation strategies can be considered and evaluated under various network and traffic conditions.

References

Akyildiz, I.F. and Wang, X. (2005) 'A survey on wireless mesh networks', *IEEE Communications Magazine*, Vol. 43, No. 9, pp.S23–S30.

Anagha, K. and Binu, G.S. (2015) 'An on demand data collection scheme for wireless sensor network based on rendezvous points', *International Research Journal of Engineering and Technology*, Vol. 2, No. 4, pp.78–82.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S.R., Raman, V., Reiss, F. and Shah, M.A. (2003) 'TelegraphCQ: continuous dataflow processing for an uncertain world', *CIDR*.

Cordeiro, C. and Agrawal, D.P. (2011) *Ad Hoc and Sensor Networks: Theory and Applications*, 28 February, World Scientific.

Diggavi, S., Grossglauser, M. and Tse, D. (2005) 'Even one-demensional mobility increases adhoc wireless capacity', *IEEE Transactions on Information Theory*, November, Vol. 51, No. 11, pp.3947–3954.

Eyadeh, A. and Amerah, M.B. (2018) 'Ferry-based directional forwarding mechanism for improved network life-time in cluster-based wireless sensor network', *International Journal of Communication Networks and Information Security*, Vol. 10, No. 2, pp.256–265.

Ghasemi, A. and Nader-Esfahani, S. (2006) 'Supporting aggregate queries over ad-hoc sensor networks', *IEEE Communications Letters*, April, Vol. 10, No. 4, pp.251–253.

Jawhar, I., Mohamed, N. and Agrawal, D.P. (2011) 'Linear wireless sensor networks: classification and applications', *Elsevier Journal of Network and Computer Applications (JNCA)*, Vol. 34, No. 5, pp.1671–1682.

Jawhar, I., Mohamed, N., Al-Jaroodi, J. and Zhang, S. (2013) 'An efficient framework for autonomous underwater vehicle extended sensor networks for pipeline monitoring', *2013 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, IEEE, pp.124–129.

Jea, D., Somasundara, A.A. and Srivastava, M.B. (2005) 'Multiple controlled mobile elements (data mules) for data collection in sensor networks', *Proc. IEEE/ACM Int. Conf. Distrib. Comp, in Sensor Sys.*

Jeong, I-S. Han, S-W. and Kang, S-H. (2013) 'Low latency and energy efficient routing tree for wireless sensor networks with multiple mobile sinks', *Journal of Network and Computer Applications*, January, Vol. 36, No. 1, pp.156–166.

Kartha, J.J. and Jacob, L. (2015) 'Delay and lifetime performance of underwater wireless sensor networks with mobile element based data collection', *International Journal of Distributed Sensor Networks*, Vol. 11, No. 5, p.128757.

Khoufi, I., Minet, P., Laouiti, A. and Mahfoudh, S. (2017) 'Survey of deployment algorithms in wireless sensor networks: coverage and connectivity issues and challenges', *International Journal of Autonomous and Adaptive Communications Systems*, Vol. 10, No. 4, pp.341–390.

Kim, D., Wang, W., Li, D., Lee, J-L., Wu, W. and Tokuta, A.O. (2016) 'A joint optimization of data ferry trajectories and communication powers of ground sensors for long-term environmental monitoring', *Journal of Combinatorial Optimization*, Vol. 31, No. 4, pp.1550–1568.

Li, Y. and Bartos, R. (2014) 'A survey of protocols for intermittently connected delay-tolerant wireless sensor networks', *Journal of Network and Computer Applications*, May, Vol. 41, pp.411–423.

Miorandi, D. and Altman, E. (20060 'Connectivity in one-dimensional ad hoc networks: a queuing theoretical approach', *Wireless Networks*, September, Vol. 12, No. 5, pp.573–587.

Mohamed, N. and Jawhar, I. (2008) 'A fault-tolerant wired/wireless sensor network architecture for monitoring pipeline infrastructures', *Proc. of The Second International Conference on Sensor Technologies and Applications (SENSORCOMM 2008)*, August, IEEE Computer Society Press, Cap Esterel, France.

Mohamed, N., Jawhar, I. and Shuaib, K. (2008) 'Reliability challenges and enhancement approaches for pipeline sensor and actor networks', *Proc. of The International Conference on Wireless Networks (ICWN 2008)*, July, Las Vegas, USA.

MomCilvic, P. and Squillante, M. (2008) 'On throughput in linear wireless networks', *Proc. of the 9th International Symposium on Mobile Ad Hoc Networking and Computing*, May, pp.199–208.

Mukherjee, R., Roy, S. and Das, A. (2015) 'Survey on data collection protocols in wireless sensor networks using mobile data collectors', *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, pp.632–636.

Oualhaj, O.A., Kobbane, A., Sabir, E., Ben-Othman, J. and Erradi, M. (2015) 'A ferry-assisted solution for forwarding function in wireless sensor networks', *Pervasive and Mobile Computing*, Vol. 22, pp.126–135.

Polat, B.K., Sachdeva, P., Ammar, M.H. and Zegura, E.W. (2011) 'Message ferries as generalized dominating sets in intermittently connected mobile networks', *Pervasive and Mobile Computing*, Vol. 7, No. 2, pp.189–205.

Tariq, M.B., Ammar, M. and Zegura, E. (2006) 'Message ferry route design for sparse ad hoc networks with mobile nodes', *MobiHoc*.

Vanarotti, G.C., Kulkarni, U.M. and Kenchannavar, H.H. (2016) 'Ferry based data gathering in wireless sensor networks', *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, IEEE, pp.165–170.

Wu, J., Dai, F., Gao, M. and Stojmenovic, I. (2002) 'On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks', *IEEE/KICS Journal of Communications and Networks*.

Xie, L., Shi, Y., Hou, Y.T., Lou, W., Sherali, H.D., Zhou, H. and Midkiff, S.F. (2015) 'A mobile platform for wireless charging and data collection in sensor networks', *IEEE Journal on Selected Areas in Communications*, Vol. 33, No. 8, pp.1521–1533.

Zema, N.R., Mitton, N. and Ruggeri, G. (2015) 'Using location services to autonomously drive flying mobile sinks in wireless sensor networks', *International Conference on Ad Hoc Networks*, Springer, pp.180–191.

Zhao, H., Guo, S., Wang, X. and Wang, F. (2015) 'Energy-efficient topology control algorithm for maximizing network lifetime in wireless sensor networks with mobile sink', *Applied Soft Computing*, Vol. 34, pp.539–550.

Zhao, W. and Ammar, M. (2003) 'Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks', *Proc. IEEE Workshop on Future Trends in Distributed Computing Systems*, May.

Zhao, W., Ammar, M. and Zegura, E. (2004) 'A message ferrying approach for data delivery in sparse mobile ad hoc networks', *ACM Mobihoc*.

Zhao, W., Ammar, M. and Zegura, E. (2005) 'Controlling the mobility of multiple data transport ferries in a delay-tolerant network', in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, IEEE, March, Vol. 2, pp.1407–1418.

Zhou, S., Zhong, Q., Ou, B. and Liu, Y. (2017) 'Intelligent compressive data gathering using data ferries for wireless sensor networks', *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp.6015–6019.