# Ferry-Based Linear Wireless Sensor Networks

Imad Jawhar[1], Mostafa Ammar[2], Sheng Zhang[3], Jie Wu[4] and Nader Mohamed[1]

[1]College of Information Technology, UAE University, Alain, UAE
[2]School of Computer Science, Georgia Institute of Technology, Atlanta, Georgia, USA
[3]State Key Laboratory for Novel Software Technology, Nanjing University, P. R. China
[4]Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania, USA

***Abstract*— Many environmental, commercial, military, and structural monitoring applications of wireless sensor networks (WSNs) involve lining up the sensors in a linear form, and making a special class of these networks; we defined these in a previous paper as Linear Sensor Networks (LSNs), and provided a classification of the different types of LSNs. A multihop approach to routing the data from the individual sensor nodes to the sink can be used in an LSN. However, this can result in a rapid depletion of the sensor energy, due to the frequent transmissions performed by the sensors to transmit their own, as well as other sensor data. In addition, in many applications, the distance between the sensors deployed to monitor the linear structure might be much greater than the communication range leading to a disconnected network where the multihop approach cannot be used. This paper presents a framework for monitoring linear infrastructures using ferry-based LSNs (FLSNs). The data that is collected by the sensors is assumed to be delay-tolerant. In such a system, a moving robot, vehicle, or any other mobile node (named a *ferry*), can move back and forth along the linear network, and collect data from the individual sensors when it comes within their communication range; The ferry can deliver the collected sensor data when it reaches the sink. It can also perform other functions, such as data processing, and aggregation, and can also transport messages from the sink to the sensor nodes (SNs). Four different ferry movement approaches are presented, simulated, and analyzed.**

***Index Terms*— Wireless sensor networks (WSNs), mobile ad hoc networks (MANETs), routing, ferry, monitoring, delay-tolerant networks (DTNs).**

## I. Introduction

Constant and important advancements in wireless sensor technology have led to the design of small, inexpensive devices which are increasingly capable of more accurate sensing, storage, processing, and communication functions. This significant technological evolution is making it possible to use wireless sensor networks (WSNs) in numerous environmental, health, military, inter-vehicular, and infrastructure monitoring applications. In a considerable number of these applications, the linear nature of the structure that is monitored imposes deployment of the sensors in a linear form [1] [2][3]. Such alignment of the sensing devices can form a "thin" or "thick" line, consisting of a cross section containing one or more sensors, respectively. In addition, the sensors may be placed at regular fixed distances, or deployed in a more random fashion, like deploying them by throwing them from a low-flying airplane along the linear structure or area which is monitored. In a previous paper [4], we provided various multihop routing protocols for data communication from the sensor nodes in a linear sensor network (LSN) to the network control center (NCC). In another paper, [1], we introduced a classification of LSNs from hierarchical and topological points of view. The paper provides some potential applications for LSNs. Such applications include: (1) Oil, gas, and water pipeline monitoring; (2) Railroad/subway monitoring; (3) Monitoring of AC powerlines; (4) A driver-alert network on long roads; (5) Border monitoring, as well as other applications. In addition, the paper provides several reasons why new frameworks and architectures are needed for different categories of LSNs. Such reasons include: (1) Increased routing efficiency; (2) Increased network robustness and reliability; (3) Improved location management algorithms; (4) Increased network security.

In this paper, we present a framework for ferry-based LSNs (FLSNs), where a mobile node (named a *ferry*) is used to collect data from the sensor nodes (SNs). The collected data is assumed to be delay-tolerant. The FLSN system provides added flexibility in the design of the network to satisfy application requirements; it does this by not requiring the distance between each of two consecutive SNs to be smaller than the communication range of the SNs. This is the case, since the classic multihop approach to provide 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, hidden terminal problems, and an increased network lifetime. In addition to eliminating the multihop overhead, the use of a ferry also solves the problem of the disproportionate use of nodes 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, eliminating the need to add nodes just to keep the data transfer feasible.

Several routing protocols have been proposed for WSNs; however, most of these protocols are designed for multi-dimensional topologies, and address a number of issues in different areas of research [5]. On the other hand, in [6], Shah et al. provide an architecture to provide connectivity of sparse

This work was supported in part by UAEU Research grant 01-03-9-12/07, and College graduate research and innovation project of Jiangsu Grant (No. CXZZ12_0055), Program A for outstanding PhD candidate of Nanjing University (No. 201301A08).

WSNs using existing mobile entities in the environment named *MULEs*.

In [7], Zhao et al. introduce a message ferrying scheme, which uses a ferry to provide communication between nodes in a highly-partitioned ad hoc network. In [8], the authors present an extension of the ferry scheme, with *task-oriented*, and *message-oriented* ferry mobility models. In [9], the message ferrying and the concept of a dominating set [10] were combined to provide a framework for delay-tolerant network (DTN) routing in mobile ad hoc networks (MANETs). In [11], Sheng, Wu, and Lu studied the use of ferries to charge static sensors as well as ferries in LSNs.

### A. The Difference Between our Strategy and the Existing WSN and DTN Routing Algorithms

Our architecture is different from existing ones, since it is designed for data collection in sensor networks with linear topologies. The following are some of the specific areas that distinguish our model:

- In some of the DTN works, the ferry must communicate with the SNs in order to determine its route, as well as perform substantial calculations in order to determine the route. This added communication and processing overhead is considerably large in networks that extend over large geographic areas, such as thousands of kilometers. In our model, this communication and processing prior to route determination is not necessary.
- In our model, the placement of the sinks can be done in a controlled fashion which optimizes the routing of the ferry and the end-to-end delay that is required by the application. This type of important and precise control over such design parameters is not possible in routing protocols for multi-dimensional delay-tolerant WSNs.
- The number of ferries that can be used to cover a relatively long LSN, which might extend for thousands of kilometers, is very difficult to determine in the existing multi-dimensional models. However, it is possible in our model to determine the number of the ferries based on the required end-to-end delay, possible available ferry technology and speed, as well as the number of SNs and the length of the size of the network.
- The number and buffering capacity of the SN nodes are also parameters that are able to be controlled, and are subject to network and application requirements in our LSN model. Such a design strategy can be very difficult, and not very scalable, in the multi-dimensional case.

### B. Motivation and Applications of FLSNs

In most of the LSN applications, a ferry can be used to collect data from the sensors which are aligned in a thin LSN or a thick LSN [1]. The ferry nodes take the form of many actual physical devices such as: (1) a moving robot sliding on a rail along a pipeline (outside or inside the pipeline). The robot is used in some cases to scan the wall of the pipeline to detect damage. (2) a moving vehicle used to collect data from sensors placed along the borders of a region that is being protected. (3) an unmanned aerial vehicle (UAV) that can be flying above an LSN which collects data from the sensors as it comes within their transmission range. (4) A mobile underwater vehicle that is used to collect data from underwater sensors that are monitoring an underwater pipeline or border region, and so on.

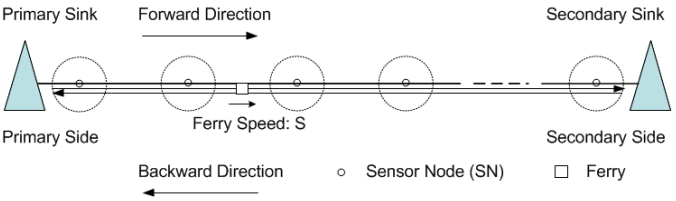



Fig. 1: The network model.

In all of these applications, the speed of the ferry node can vary from in orders of magnitude from a very slowly moving robot (e.g. 0.01 to 1 m/s), to a fast-moving robot or UAV (e.g. 1 to 100 m/s). While communication with a fast-moving object can introduce more issues at the physical layer such as Doppler effects, we do not focus on this in our paper, and we assume that the lower layers (e.g. physical and data link) of the corresponding SN and ferry devices will handle such issues. In addition, the range of speed that is used by the ferry in our simulation does not include all possibilities that can arise in real applications. However, we use this range only to illustrate the operation of the algorithms and study their relative performance and effect over various parameters under different network and SN traffic conditions.

## II. Network Model

Fig. 1 shows an illustration of a ferry-assisted thin one-level LSN. We define the following parameters, which are used in the analysis in this paper:

- $n$: This is the number of sensor nodes that are used to cover the entire network.
- $d$: This the physical distance between each of two consecutive sensor nodes.
- $L$: The length of data which is exchanged with the ferry (in bytes).
- $R$: The bit rate for the communication between the SN and the ferry.
- $R_s$: This is the sensing range of a sensor node.
- $R_c$: The communication range of an SN. In our model, we assume a unit disk communication range in order to simplify the analysis without loss of generality. This is the case, since we mainly focus on the operation of the various algorithms and their relative performances.
- $M$: The message generation rate of the the SNs.

In our model, two types of data traffic are considered: (1) Homogeneous data, where all of the data generated by an SN is of the same type, which is assumed to be best effort (BE) traffic. (2) Heterogeneous data, where the data generated by an SN is assumed be of two different types, depending on the nature and criticality of the related event or parameter.

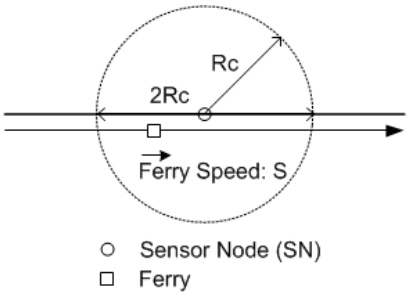



Fig. 2: Exchange of data when the ferry is within range of the sensor node.

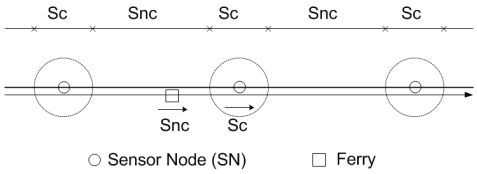



Fig. 3: The Variable Speed Ferry (VSF) model.

### A. Constant Speed Ferry (CSF)

In our basic model, which is illustrated in Fig. 1, there is one ferry in a segment that consists of $n$ sensor nodes between two sinks, which are placed at the two edges of the segment. The ferry moves periodically at a constant speed between the sinks, collecting data from the SNs and delivering it to each of the sinks.

For the CSF model with $n$ SNs between the sinks, and a ferry speed $s$, the maximum message delay for CSF is: $T_{max}^{CSF} = \frac{(n+1)d}{s}$. On the other hand, the minimum message delay is 0. Consequently, the average delay for CSF is: $T_{avg}^{CSF} = \frac{(n+1)d}{2s}$. In order to not have any dropped messages due to delay, we have: $\frac{nd}{s} \leq T$. Therefore, $s \geq \frac{nd}{T}$.

We develop a formula governing the exchange of data between the ferry and the node. This is illustrated in Fig. 2. Let $t_{fp}$ be the time of passing of the ferry within range of the SN. Then, this time is: $t_{fp} = \frac{2R_c}{s}$. However, in order for the data that is needed to be exchanged with the ferry to actually be communicated during the time $t_{fp}$, we must: $L \leq \frac{t_{fp}R}{8}$. Substituting for $t_{fp}$, we get: $L \leq \frac{R_c R}{4s}$. Consequently, if $L$ is already determined as a requirement, and we need to derive the resulting minimal ferry speed for $L$ bytes to be exchanged, we get the following relationship:

$$s \leq \frac{R_c R}{4L} \tag{1}$$

In order to not have any dropped messages due to buffer overflow at the ferry, the minimum requirement for the size of the ferry buffer (in bytes) is: $B_f^{CSF} = nL$. So, combining this with the previously derived expression for $L$, we get: $B_f^{CSF} = \frac{nR_c R}{4s}$.

### B. Variable Speed Ferry (VSF)

In this variation of our model, which is illustrated in Fig. 3, the ferry moves in the same pattern as the constant speed ferry, but with the following two different speeds: (1) *Ferry speed while in communication ($s_c$)*; and (2) *Ferry speed while no communication is taking place ($s_{nc}$)*. This speed is usually the faster of the two speeds. In order to reduce the end-to-end delay, the ferry should move as fast as possible in order to reach the communication range of the next SN. The acceleration and deceleration time periods are assumed to be negligible in our analysis. However, such periods might need to be considered in some applications and specific physical implementations of the ferry. Consequently, in our case, the maximum delay of the message, $t_{max}^{VSF}$, is: $t_{max}^{VSF} = (n-1)(\frac{(d-2R_c)}{s_{nc}} + \frac{2R_c}{s_c})$.

A similar analysis to the CSF case provides the following calculation for the size of the ferry buffer for the VSF case: $B_f^{VSF} = \frac{nR_c R}{4s_c}$.

### C. Adaptable Speed Ferry (ASF)

Another model that we consider is the one where the message arrival rate at the SNs is variable. This kind of assumption would be valid for certain types of applications, where the rate of collection of information at the SNs can be related to certain external factors, such as an emergency or abnormal situation in the area that is monitored by the SN. For example, some SNs might have special audio/video capabilities which would not normally be turned ON, due to power considerations. However, such audio/video collection and transmission might be required in response to certain events, such as the detection of an intruder in a monitored area, high temperature, pressure, or vibration in a pipeline, or detecting any predefined specific or abnormal situation. Consequently, an increase in the quantity, quality, and nature of the collected information might be required. For this model, we propose an *Adaptable Speed Ferry with Delay Allocation (ASF-DA)*, which is illustrated in Fig. 4, and described in Algorithms 1, 2, and 3.

**Algorithm 1** ASF-DA Algorithm - Delay Quota Initialization

```
/* Set total delay quota for node i according to its position. T_q is
a basic quantum (or unit) delay parameter that is set depending
on the application.*/
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
```

In the two previous cases of the CSF and VSF algorithms, the end-to-end delay incurred by the data to the sink is bounded. However, in the ASF case, this delay is un-bounded, while, the no-communication speed of the ferry is constant, and therefore contributes with a deterministic amount to the overall delay. The ferry speed while in communication is determined by the amount of data in the buffer of the SN that is being visited. This can be so large as to cause a significant slow down of the ferry, resulting in unacceptable overall end-to-end delay. In order to place a bound on the delay, a delay allocation strategy is used. Upon network initialization, as well

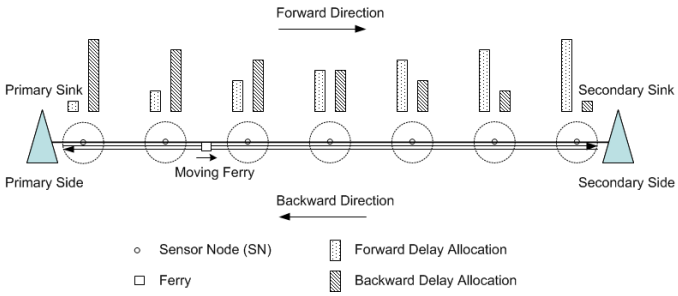



Fig. 4: Delay allocation in the ASF-DA algorithm.

**Algorithm 2** ASF-DA Algorithm - Ferry/SN Data Exchange

**if** ($D_i^t$*$R$) $\geq B_i^t$ **then**
/* Allocated delay quota is more than the time required to download the buffer of SN $i$. So calculate the *actual delay* time to download the current buffer of node $i$ */
$D_i^a$ = $B_i^t$ / $R$
/* Set the ferry speed to download all data at SN $i$ */
$s_{ci}$ = 2*$R_c$ / $D_i^a$
/* Calculate the remaining delay */
$D_i^r$ = $D_i^t$ - $D_i^a$
/* Distribute the remaining delay quota to the rest of the SNs between this SN and the destination sink */
$distributeRemainingDelay(dir,\ i,\ D_i^r)$
**else**
/* Allocated delay quota is less than or equal to the time required to download all. So set ferry speed according to delay quota. */
$s_{ci}$ = 2*$R_c$ / $D_i^t$
/* Set the actual delay equal to the total delay quota */
$D_i^a$ = $D_i^t$
**end if**
/* Download the data using the calculated *actual delay* */
$downloadDataFromCurNode(i,\ D_i^a,\ \lambda)$

as each time the ferry reaches a sink where it changes direction in order to go to the opposite sink, the delay quota initialization algorithm presented in Algorithm 1 (and illustrated in Fig. 4) is used. If the ferry is about to go in the forward direction, then the total delay quota for each node $i$ is allocated in an inversely proportional manner to its proximity from the sink where the ferry is leaving. This is done in order to reduce the amount of data that is downloaded from the early SNs in the ferry trip, which will experience the most end-to-end delay. Consequently, more data will be collected from the nodes that are closer to the opposite sink, which will experience the least delay. Therefore, the average end-to-end delay experience by the downloaded data will be lower. When the ferry goes back in the opposite direction, the delay quota allocation is changed again to give the SNs at the beginning of the new trip less delay quota, and the SNs on the opposite side more, thereby balancing the delay quota allocation in both directions. The delay allocation in our case is independent of the message arrival rate. However, for some applications where the arrival rate is highly variable between SNs, the rate of each one can be included in the delay allocation in addition to the SN position. Furthermore, the ASF-DA Ferry/SN data exchange algorithm is presented in Algorithm 2, and the pseudocode for the function *downloadDataFromCurNode(i, $D_i^a$, $\lambda$)* is shown in Algorithm 3.

**Algorithm 3** Download While Ferry is in Communication Range Function: $downloadDataFromCurNode(i,\ delay,\ \lambda)$

/* Total downloadable priority traffic size at SN $i$ (in bytes): */
$B_i^p$ = $R$ * $Delay$ * $\lambda$/8
Download $B_i^p$ bytes from priority data buffer of $i$.
/* Total downloadable BE traffic size at SN $i$ (in bytes): */
$B_i^b$ = $R$ * $Delay$ * (1-$\lambda$)/8
Download $B_i^b$ bytes from BE data buffer of $i$.

**Algorithm 4** ARF Algorithm - Initialization When Ferry Starts in the Direction of the Opposite Sink

/* Initialization of $F_{tip}$ flag */
**if** $blackNodeCounter$ = $n$ **then**
**for** ($i$=1; $i \leq n$; $i++$) **do**
$blackNodeCounter$ = 0
$F_{tip}^i$ = $WHITE$
**end for**
**end if**
/* Current node variable initialization */
**if** ($networkStartup$ = $TRUE$) $OR$ ($dir$ = $FORWARD$) **then**
$cn$ = 1
**else**
$cn$ = $n$
**end if**
$ferryMode$ = $NORMAL\_MODE$
$ferrySpeed$ = $NORMAL\_SPEED$

**Algorithm 5** ARF Algorithm - Ferry Algorithm 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.
/* calculate the total download time for node $i$ */
$D_i^t$ = 2*$R_c$/$ferrySpeed$
$downloadDataFromCurNode(i,\ D_i^t,\ \lambda\ )$
**if** $F_{sit}^i$ = $WHITE$ **then**
**if** $curFerryBufferSize \geq \rho$ **then**
$dir$ = $getDirToNearestSink()$
$ferrySpeed$ = $FERRY\_MAX\_SPEED$
$ferryMode$ = $GO\_FAST\_TO\_NEAREST\_SINK$
**end if**
/* Indicate that this node has been serviced in the current pass */
$F_{sit}^i$ = $BLACK$
$blackNodeCounter$ = $blackNodeCounter$ + 1
**end if**
$i$ = $getNextNodeInDir(i,\ dir)$
Go to $i$ node at the current $ferrySpeed$

### *D. Adaptive Routing Ferry (ARF)*

In this algorithm, which is illustrated in Fig. 5, the ferry starts at the primary sink, and moves in the forward direction towards the secondary sink, using a constant speed *NORMAL_SPEED*. This is the initial mode of operation named

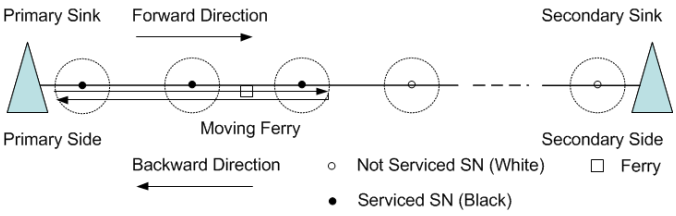



Fig. 5: Illustration of the ARF algorithm.

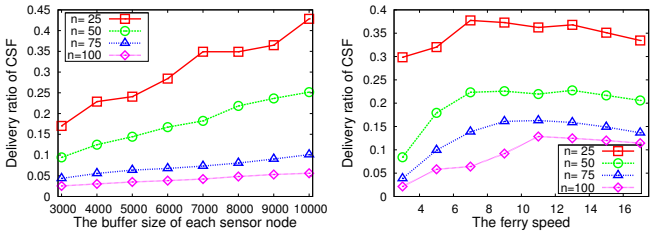
*NORMAL_MODE*. In this mode of operation, as the ferry comes within communication range of an SN, it downloads data from that SN. It continues to do so until the ferry buffer reaches a critical threshold $\rho$. In that case, the ferry switches to a mode of operation in which it goes as fast as possible to the nearest sink. This mode is called the *GO_FAST_TO_NEAREST_SINK* mode, which causes the ferry to set the direction towards the nearest sink, and set the speed to *FERRY_MAX_SPEED*. This is done in order to minimize the average end-to-end delay of the data traffic by making the ferry deliver the data as soon as possible once the data size grows to a certain predetermined threshold $\rho$. The latter threshold is a parameter which is set by the network administrator, and is application-dependent. When the ferry reaches the sink and turns around to make a new trip, it will go quickly past the segment of the network where the SNs were already serviced in the last trip, until it reaches SNs which have not been serviced yet. When this happens, the ferry switches back to $NORMAL\_SPEED$ and proceeds by servicing the unserviced SNs. This is done in order to provide fairness in servicing all of the SNs in the segment, as well as to prevent SN starvation. In order to keep track of the service status of the SNs, each node $i$ has a serviced-in-this-trip flag, $F^i_{sit}$. The operation of the ARF algorithm is presented in Algorithms 4 and 5.

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 $\lambda$. Here, we use $\lambda = 0.8$, which indicates that 80% of the quota is used for priority traffic, and the remaining portion is used by the BE traffic.

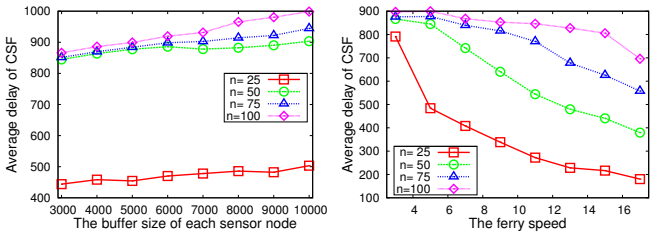
## III. Simulation

### A. Simulation Setup

This section evaluates the performance of the proposed algorithms. The default values of the input parameters are set as follows: $n = 50$, $d = 150$ meters, $R_c = 50$ meters, $R = 2,000$ bps, the buffer size of each SN $B = 8,000$ bytes, for CSF and VSF $s = 10$ m/s, for VSF $s_c = 10$ m/s, for VSF and ASF $s_{nc} = 20$ m/s, the delay quota in ASF $T_q = 0.5$ s, the NORMAL_SPEED in ARF is 10m/s, the FERRY_MAX_SPEED is 50 m/s, the message time-out value $T = 1,500$ s, and $\rho = 10,000$ bytes. It is worth noting that, in our simulation, we mainly focus on the relative performance of the algorithms, and we do not emphasise issues related to the physical layer, which are outside the scope of our work in this paper.

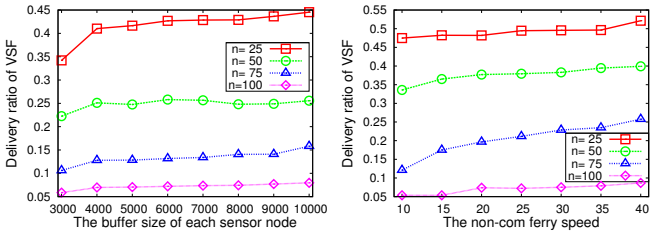



(a) Buffer size vs. delivery ratio (b) Ferry speed vs. delivery ratio

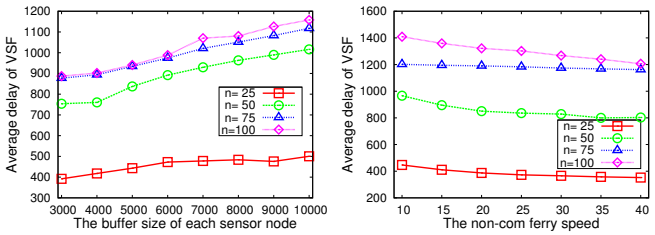



(c) Buffer size vs. average delay (d) Ferry speed vs. average delay

Fig. 6: Performance comparison results of CSF.




(a) Buffer size vs. delivery ratio (b) Non-com Ferry speed vs. delivery ratio



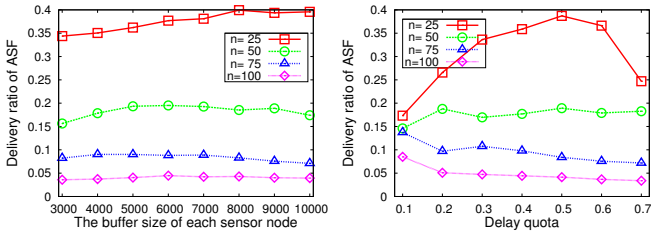

(c) Buffer size vs. average delay (d) Non-com Ferry speed vs. average delay

Fig. 7: Performance comparison results of VSF.

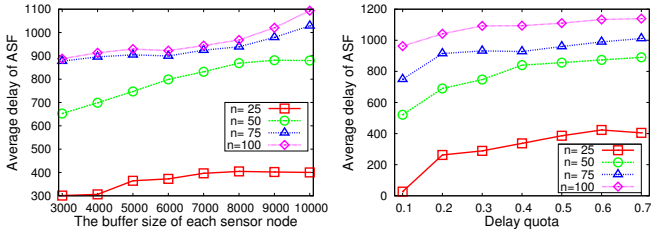
In all four algorithms, the BE and priority traffic are generated based on exponential distributions with an average arrival rate that is uniformly generated from an interval. In our experiments, the average BE traffic in bytes per second is uniformly generated between 1 and 3; the average priority traffic in bytes per second is uniformly generated between 8 and 16. The lambda in our evaluations is always set to 0.8. The performance metrics used in our evaluations are the delivery ratio and the average delay. The delivery ratio is the ratio of the successful packets received by two sinks to the overall packets generated by all sensor nodes. The average delay is the average delay experienced by the successful packets.

### B. Summary of Simulation Results

The simulation results are presented in Figs. 6, 7, 8, and 9. Due to space limitations, we summarize the key observations as follows:

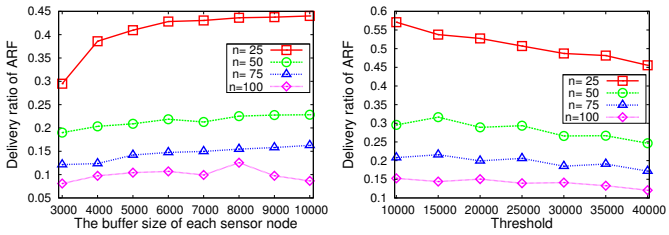



(a) Buffer size vs. delivery ratio (b) Delay quota vs. delivery ratio

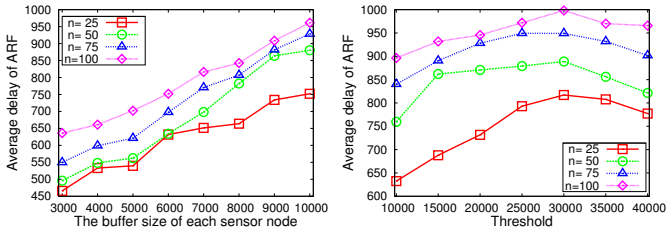



(c) Buffer size vs. average delay (d) Delay quota vs. average delay

Fig. 8: Performance comparison results of ASF.




(a) Buffer size vs. delivery ratio (b) Buffer threshold vs. delivery ratio




(c) Buffer size vs. average delay (d) Buffer threshold vs. average delay

Fig. 9: Performance comparison results of ARF.

- When the buffer size of each sensor node increases, all of the proposed algorithms can deliver more packets to sinks. However, as the generated packets in each sensor node are stored in a sorted queue, based on their generation time, and the ferry gives high priority to packets with small generation time in our implementation, the average delay of the proposed algorithm goes up when the buffer size increases.
- The ferry speed, the non-com ferry speed in VSF, the delay quota in ASF, and the buffer threshold in ARF should be chosen carefully, and it may take some time to find the optimal values for a particular network environment.
- Additional simulation results, which could not be included in this paper due to size limitations, show that when the time-out value increases, all of the proposed algorithms perform better in terms of delivery ratio, and the average delay increases. Moreover, there may exist some critical time-out values, which correspond to the length of the time required for a ferry to travel from one sink to another.
- We admit that some challenges remain. However, the thesis of this paper is the introduction of the FLSN framework along with various strategies for data collection. While preliminary, our results indicate that the proposed algorithms perform well in a variety of network environments. We hope that our simulation provides some guidelines for future delay-tolerant data collecting systems.

## IV. Conclusion

In this paper, FLSNs, which constitute a new type of LSNs, were introduced where a mobile node (acting as a ferry) moves along the LSN and collects/delivers information from/to the SNs from one or multiple sinks. The paper provided motivations for the design of FLSNs, and offered four approaches based on the ferry movement strategy. Simulation experiments were performed to study the effect of different design parameters, such as ferry speed, data rate, number of sensors, and transmission range on the performance of the network, with respect to important metrics, such as end-to-end data delivery delay, and buffer size requirements. We believe that the FLSN framework presented in this paper forms a good foundation for future work. We plan to expand the design and analysis further by studying different aspects, such as the effect of using multiple ferries per segment, and also provide algorithms for designing efficient ferry routes and movement synchronization given certain variable network conditions, as well as data traffic distributions.

## References


[1] I. Jawhar, N. Mohamed, and D. P. Agrawal. Linear wireless sensor networks: Classification and applications. *Elsevier Journal of Network and Computer Applications (JNCA)*, 34:1671–1682, 2011.

[2] Z. Sun, P. Wang, M. C. Vuran, M. A. Al-Rodhaan, A. M. Al-Dhelaan, and I. F. Akyildiz. MISE-PIPE: Magnetic induction-based wireless sensor networks for underground pipeline monitoring. *Elsevier Ad Hoc Networks Journal*, 9, 2011.

[3] Y. Guo, F. Kong, D. Zhu, A. Tosun, and Q. Deng. Sensor placement for lifetime maximization in monitoring oil pipelines. *The 1st ACM/IEEE Int'l Conf. on Cyber-Physical Systems*, 2010.

[4] I. Jawhar, N. Mohamed, K. Shuaib, and N. Kesserwan. An efficient framework and networking protocol for linear wireless sensor networks.

[5] K. Akkaya and M. Younis. A survey of routing protocols in wireless sensor networks. *Elsevier Ad Hoc Networks*, 3(3):325–349, 2005.

[6] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: modeling a three-tier architecture for sparse sensor networks. *IEEE SNPA*, 2003.

[7] W. Zhao and M. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. *In Proc. IEEE Workshop on Future Trends in Distributed Computing Systems*, May 2003.

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

[9] B. Polat, P. Sachdeva, M. Ammar, and E. Zegura. Message ferries as generalized dominating sets in intermittently connected mobile networks. *Elsevier pervasive and mobile computing journal*, 2011.

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

[11] S. Zhang, J. Wu, and S. Lu. Collaborative mobile charging for sensor networks. *Proc. of the 9th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2012)*, October 2012.