Coverage Aware Buffer Management and Scheduling for Wireless Sensor Networks

Eugene Chai, Mun Choon Chan and A. L. Ananda
School of Computing
National University of Singapore
{chaisong, chanmc, ananda}@comp.nus.edu.sg

Abstract— **Environmental monitoring and surveillance is a popular application of wireless sensor network. In such an application, the data transmitted are tagged with geographic information. A network with better coverage provides better quality-of-service since it will be able to monitor its area of responsibility more effectively.**

In this work, we study the impact of congestion on coverage of the sensor network. Congestion can negatively impact the performance since it can result in reduced coverage and power wastage. In this paper, we present a buffer management scheme called *Most Redundant Drop (MRD)* and a scheduling algorithm called *Coverage Transmit (CT)* that make use of spatial information in sensor data to improve network coverage. Compared to drop-tail and FIFO, MRD and CT improve coverage by up to 75% when exact sensor location is available. Since, exact location may not be available in practice, we present a modified DV-Hop scheme that provides approximate localization. Simulation results show that substantial improvement can also be obtained.

I. Introduction

The race towards miniaturization has produced, amongst other things, relatively minuscule wireless sensors that, when organized into a network, have the potential to fundamentally change the ways in which we interact with our environment. One common application of wireless sensor network is environmental monitoring where a large number of sensor nodes is scattered within an area of interest, and data is collected by one or more sink nodes strategically placed within the sensor field.

One characteristic of such networks is the high fan-out of nodes near the sinks. As nodes closest to the sinks serve as the only links from the sinks to the rest of the network, all data from the sensor network must pass through them. The effects of congestion among these nodes becomes more pronounced as the size of the network grows. Since the bandwidth of these nodes is limited, it is important to select the appropriate packets for buffering and transmission, so as to maximize network coverage.

Another characteristic of interest is the use of geographical information. Sensor data usually comes attached with location information. A packet indicating a spike in temperature somewhere within the area under observation also contains the location of the event. Such information can be conveyed by including in the packet header either (a) the node ID or (b) coordinates of the node that detected the event. In the former case, the sink node will then need to map the node ID to its location using pre-computed data. Obviously, this would be feasible only for pre-planned deployments. In the latter case, location information can be obtained via GPS or one of the localization algorithms described in [3], [4] and [7]

An important performance objective in sensor network is its *coverage*. As mentioned in [6], *coverage* has different specific interpretations, but it is generally regarded to be a measure of the *quality of service* offered by the network. The coverage of the network affects the ability of the sensor network to detect the occurrence of certain events. A point is said to be *covered* if (a) it lies within the sensing area of *at least 1 node* and (b) packet generated by that node reach the sink. If the network is congested, then it becomes more likely that packets from different sources will be dropped before they reach the sink, resulting in events being undetected. It is therefore useful to make the buffering and scheduling mechanisms *coverage aware*.

Most of the existing work consider either scheduling and buffer management or coverage, but not both at the same time. In this paper, we present a buffer management (Most Redundant Drop or MRD) and a scheduling algorithm (Coverage Transmit or CT) that make use of spatial information when selecting a packet for dropping and transmission to improve coverage. These algorithms are fully distributed and application independent. No inter-node signalling is needed. The main requirement

is for a data packet to carry geographical location of its source node. In addition, we also show that the proposed algorithms work well even without accurate location information. Indeed, an approximate localization, loosely based on DV-Hop, where no node (including the anchor node) has its exact location is sufficient for the purpose of improving coverage.

The rest of this report is organized as follows. In Section II, we review related work before presenting our theoretical model and motivation in Section III. MRD and CT are described in Section IV and the evaluation using exact coordinates is presented in Section V. The modified DV-Hop localization and evaluation of its used in MRD and CT are presented in Section VI.

II. Related work

Buffer management and scheduling have been discussed in sensor network but not for the purpose of improving network sensing coverage during congestion. Reliable Bursty Convergecast [15] uses a novel *windowless queue management* technique to avoid problems with traditional synchronous explicit ack and stop-and-wait implicit ack schemes. RBC can also enforce a *differentiated contention control* to reduce the congestion brought about by any necessary packet retransmissions.

Scheduling schemes are often used to minimize energy usage, such as *lazy packet scheduling* described in [10]. Lazy scheduling stems from the idea that it is more efficient, in terms of the amount of power expended, to transmit a packet at low power and a lower bitrate, then to transmit the same packet at a higher bitrate and power level. The authors describe a scheduling scheme that gives each packet the longest tranmission duration possible while taking other temporal constraints, such as the maximum useful lifetime of the packet, into consideration.

However, in [8], the authors point out that using lazy scheduling alone may result in sub-optimal energy usage. Since the radio circuitry is a significant consumer of node energy, a good way of extending the useful lifetime of the node would be to shutdown the radio periodically, which runs against the mantra of lazy scheduling since it means that packets have to be transmitted as quickly as possible. The authors propose an alternative scheduling algorithm that assigns *transmit opportunities* (TXOP) to nodes while taking both radio shutdown and lazy scheduling policies into account.

In sensor networks, event detection is crucial. The Event-to-Sink Reliable Transport (ESRT) described in [9] makes use of end-to-end congestion control to ensure the reliable delivery of network events to the sink. ESRT defines five states that a network can be in: (1) No Congestion, Low Reliability, (2) No Congestion, High Reliability, (3) Congestion, High Reliability, (4) Congestion, Low Reliability and (5) Optimal Operating Region. By adjusting the reporting rate at fixed sensing intervals, ESRT aims to keep the network operating in state (5), where the best possible balance between event detection and reporting rate is achieved.

An alternative congestion control method is used by the Fusion protocol [5]. Instead of end-to-end techniques, it uses hop-by-hop flow control along with rate limiting and a prioritized MAC protocol to reduce network congestion. Another protocol that uses hop-by-hop congestion control is the Congestion Detection and Avoidance (CODA) protocol, described in [11]. CODA uses *open-loop, hop-by-hop back pressure* to deal with congestion over small timescales and *closed-loop, multi-source regulation* to deal with more persistent congestion.

The authors in [2] describe a congestion control and fairness protocol. The congestion control protocol deals with with two main forms of congestion: congested wireless channels (Type A) and buffer overflows (Type B). Node transmission is regulated by the propagation of an acceptable transmission rate from the parent towards its children. Fairness in the network is achieved by ensuring that the number of packets received from each node over the same time period is approximately equal.

Topology control is yet another alternative to ease congestion and ensure the required coverage. Probing Environment and Adaptive Sleeping (PEAS) [14] is a topology control protocol that switches off redundant nodes to conserve battery power. Only the minimum number of nodes required to maintain sensing coverage is kept in operation. Nodes occasionally wake up to probe the network in order to determine whether they should enter the active state. The Coverage Control Protocol (CCP) [13] operates on a similar idea, but with the added ability for the network to dynamically configure itself to provide the requested degree of coverage.

The nature of sensor networks opens itself to various graph theoric methods to ensure coverage. In [6] the authors define the Maximal Breach Path and the Minimal Support Path using Voronoi diagrams and Delaunay triangulation. These metrics are then used to determine the best and worst case coverage of a sensor network.

III. Theoretical Model

Consider an area A under surveillance to be a unit square. Let the set of randomly deployed wireless sen-

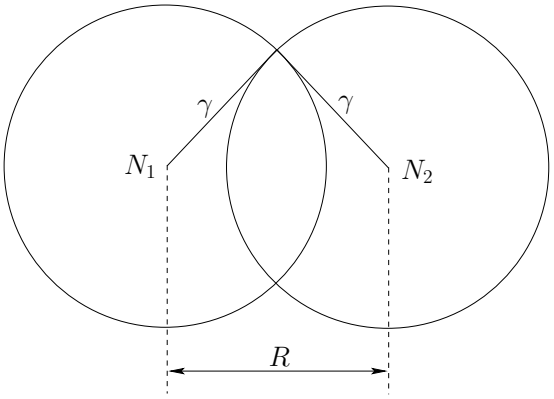


Fig. 1. Two sensors, each with sensing radius γ, at a distance R apart

sor nodes be N. Let X_1, X_2, Y_1, Y_2 be independent random variables representing the coordinates, (X_1, Y_1) and (X_2, Y_2), of two nodes in N.

Let the distance between the two nodes be represented by R. This is the *Square Line Picking* problem as described in [12]. The probability density function (PDF) of R is given by

$$f_R(r) = \begin{cases} 2r(r^2 - 4r + \pi), & 0 \le r \le 1 \\ 2r(4\sqrt{r^2-1} & \\ -4\arctan(\sqrt{r^2-1}) & \\ -2 - r^2 + \pi), & 1 < r \le \sqrt{2} \end{cases} \quad (1)$$

The cumulative distribution function (CDF) of R is given by

$$F_R(r) = \begin{cases} \frac{1}{2}r^4 - \frac{8}{3}r^3 + \pi r^2, & 0 \le r \le 1 \\ -4r^2\arctan(\sqrt{r^2-1}) - \frac{1}{4}r^4 & \\ +\frac{4}{3}\sqrt{r^2-1}(2r^2+1) & \\ +(\pi-1)r^2 + \frac{1}{3}, & 1 < r \le \sqrt{2} \end{cases} \quad (2)$$

Let γ be the radius of the sensing circle of each node. For any two nodes such that $R < 2\gamma$ (as shown in 1), the overlapping sensing area is given by

$$L(R = r) = 2\gamma^2 \arccos\left(\frac{r}{2\gamma}\right) - r\sqrt{\gamma^2 - \left(\frac{r}{2}\right)^2} \quad (3)$$

if $R < 2\gamma$ and $L(R) = 0$ otherwise. Assuming that $\gamma << 1$, the expected overlapping area is

$$E(L) = \int_0^{2\gamma} L(R = r) f_R(r) \quad (4)$$

Let the set $S_j \subset N$ contains j randomly selected nodes from N. The total coverage achieved by these j nodes is denoted by $\mathrm{C}(S_j)$. For each node $n \in S_j$, the

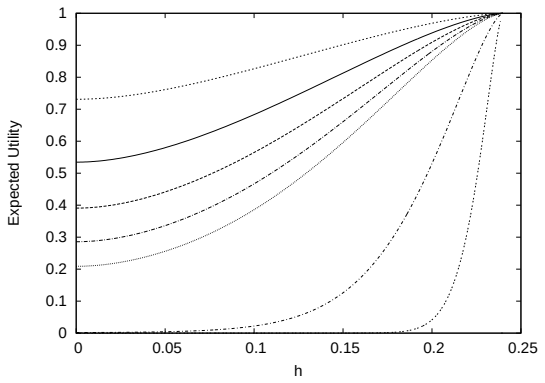


Fig. 2. Utility vs. h, $\gamma = 0.12$. From top to bottom, the plots correspond to $j = 1, 2, 3, 4, 5, 20$ and 100

utility of n is defined as $\left(\mathrm{C}(S_j) - \mathrm{C}(S_j \setminus \{n\})\right)/(\pi\gamma^2)$ (i.e. the fraction of the sensing circle of n that has no overlap with the sensing circle of the other nodes).

Let node $s_1 \in S_j$ be a distance $R_1 (< \gamma)$ from n. The utility of n (with respect to s_1) is defined as

$$\theta(n) = \left(1 - \frac{L(R_1)}{\pi\gamma^2}\right) \quad (5)$$

Consider the case where all nodes in $S_k \setminus \{n\}$ have their sensing circles overlapping with the sensing circle of n. Let the nodes be $s_1, s_2 \cdots s_j$ and the distance of these j nodes from n be given by random variables $R_1, R_2, \ldots R_j$. Since the placement of the nodes are independent of each other, the utility of n is given by

$$\theta(n) = \left(1 - \frac{L(R_1)}{\pi\gamma^2}\right) \ldots \left(1 - \frac{L(R_j)}{\pi\gamma^2}\right). \quad (6)$$

If we consider the range of values that each random variable $R_1, R_2, \ldots R_j$ can take, then the expected utility of n assuming j overlapping nodes is

$$E_j\left[\theta(n)\right] = \left[\int_0^{2\gamma} \left(1 - \frac{L(r)}{\pi\gamma^2}\right) \frac{f_R(r)}{F_R(2\gamma)}\, \mathrm{d}r\right]^j \quad (7)$$

If we construct S_j such that each node in S_j is at least a distance of h from n, then the expected utility as given in (7) becomes

$$E_j\left[\theta(n)\right] = \left[\int_h^{2\gamma} \left(1 - \frac{L(r)}{\pi\gamma^2}\right) \frac{f_R(r)}{F_R(2\gamma) - F_R(h)}\, \mathrm{d}r\right]^j \quad (8)$$

Fig. 2 shows a graph of the expected utility of n against h over a unit square with $\gamma = 0.12$. As we increase the minimum distance h between n and all other nodes, $E_j\left[\theta(n)\right]$ increases. For sufficiently large h, the nodes are far apart and the utility is 1.0. In addition, when j increases, $E_j\left[\theta(n)\right]$ is small since there is likely to be high overlap or redundancy among nodes.

The result in Fig. 2 provides the motivation for the proposed buffer management and scheduling algorithm. When a packet needs to be dropped, drop the packet that

results in the smallest h value such that its removal will increase the minimum distance among all nodes. Fig. 2 shows that by ensuring that the k packets remaining on the queue are as far away from each other as possible, the expected utility of each packet is higher. Likewise, the scheduling algorithms would do well to select packets with source nodes that are as far away as possible from the source nodes of other packets. In Section IV, we present algorithms that exploit these heuristics.

IV. Buffer Management and Packet Scheduling

A. Virtual Queue

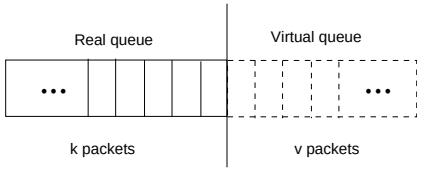


Fig. 3. Virtual queue adds history to the real (data) queue

Before describing the buffer management and scheduling algorithms, we introduce the idea of a *virtual queue*, as shown in Fig. 3. Basically, a virtual queue stores the *headers* of the most recently transmitted packets. Assuming that the size of the packet header is only a small fraction of the total packet size, the virtual queue allows a node to cache its transmission history with minimal memory overhead. For the rest of this paper, we will explicitly state *real queue* for the queue that contains data and *virtual queue* when there is a need to differentiate between them.

B. Most Redundant Drop (MRD)

MRD is based on the idea that for any two sensor nodes, the correlation between the data reported by the two nodes decreases with increasing distance between them. If the nodes are near to each other, there will be a large degree of redundancy in the sensing data reported by the two nodes. During congestion, dropping a packet among a set of nodes far apart is likely to result in the loss of more information compared to dropping a packet from nodes close together.

Let Q be the queue of a sensor node. Q consists of two parts: a *real* queue of length k packets and a *virtual* queue of length ν packets. We will refer to the real and virtual queues as $\mathrm{Re}(Q)$ and $\mathrm{Vir}(Q)$ respectively. We will also refer to the source node n_i of any packet p_i as $\mathrm{Src}(p_i)$. We use $\mathrm{d}(n_i, n_j)$ to denote the distance between the nodes n_i and n_j, where $\mathrm{d}(\cdot,\cdot)$ is the Euclidean distance function. To simplify notation, we will use $\mathrm{d}(p_i, p_j)$ to denote $\mathrm{d}(\mathrm{Src}(p_i), \mathrm{Src}(p_j))$. Finally, let $D(p_i, S) = \sum_{p_j \in S \setminus \{p_i\}} \mathrm{d}(p_i, p_j)$.

```
RECEIVE-PACKET(p)
1  ▷ Q is a queue of length (k + 1) + ν,
   ▷ where the maximum length of Re(Q) is k + 1
   ▷ and the maximum length of Vir(Q) is ν
2  enqueue(Q, p)
3  if length[Q] = k + 1
4     then BUFFER-MGT(Q)

MOST-REDUNDANT-DROP(Q)
 1  dist ← ∞
 2  p1 ← NULL
 3  p2 ← NULL
 4  for i ← 1 to length[Re(Q)]
 5       do for j ← i + 1 to length[Q]
 6            do if d(Q[i], Q[j]) ≤ dist
 7                 then p1 ← i
 8                      p2 ← j
 9                      dist ← d(Q[i], Q[j])
10  if p2 > k + 1
11     then drop(Q[p1])
12     else if D(p1, Q) < D(p2, Q)
13             then drop(Q[p1])
14             else drop(Q[p2])
```

Fig. 4. Most Redundant Drop (MRD) algorithm

If $\mathrm{Re}(Q)$ is full when an incoming packet p arrived, we find two packets p_1 and p_2 that are *closest* together using the following rules.

1) $p_1 \in \mathrm{Re}(Q) \cup \{p\}$ and $p_2 \in Q \cup \{p\}$, $p_1 \neq p_2$
2) $\forall p_i \in \mathrm{Re}(Q) \cup \{p\}$ and $\forall p_j \in Q \cup \{p\}$, $p_i \neq p_j$, $\mathrm{d}(p_1, p_2) \leq \mathrm{d}(p_i, p_j)$

The drop policy is as follows:

- if $p_2 \in \mathrm{Vir}(Q)$, then drop p_1, or
- if $p_2 \in \mathrm{Re}(Q) \cup \{p\}$, then we drop p_1 if and only if $D(p_1, Q \cup \{p\}) < D(p_2, Q \cup \{p\})$. Otherwise, drop p_2.

This buffer management algorithm has the property that if Q is the queue before MRD executes and Q' is the queue after a packet has been dropped, then $\sum_{p_i, p_j \in Q} \mathrm{d}(p_i, p_j) \leq \sum_{p_m, p_n \in Q'} \mathrm{d}(p_m, p_n)$. Hence, when a sensor node encounters congestion, space in the queue is biased towards packets with higher utility values. The pseudo-code for the algorithm is shown in Fig. 4.

C. Coverage Transmit (CT)

When FIFO scheduling is used, the scheduler always transmits packets from the head of the queue and it may transmit a large number of low utility packets before reaching one with high utility. Furthermore, given the low bandwidth of sensor nodes, the time interval

between the transmission of high utility packets can be substantial. This problem can be solved by selecting, from within the queue, the packets with the highest utility for transmission. In this section, we present a scheduling algorithm called *Coverage Transmit (CT)* that tries to select the packet that maximizes coverage for transmission.

While the basic idea of finding packets with high utility is similar to the buffer management problem, it is actually much more complicated than finding packets with low utility. This is because when two packets are very close together, the overlap or redundancy is high *independent of other packets*. However, a packet has high utility only if it has the least sensing overlap with all other packets. As a result, finding high utility or low redundancy packets requires that all packets be taken into account at the same time. Accurate computation for finding high utility packets is thus too expensive. Instead, an approximation, as outline below, is used.

For each packet $p_i \in \mathrm{Re}(Q)$, let $V_i = \{\overrightarrow{v} | \overrightarrow{v}$ is a vector from $\mathrm{Src}(p_i)$ to $\mathrm{Src}(q)$ where $q \in \mathrm{Vir}(Q)$ and $\mathrm{d}(p_i, q) \leq 2\gamma\}$. Let

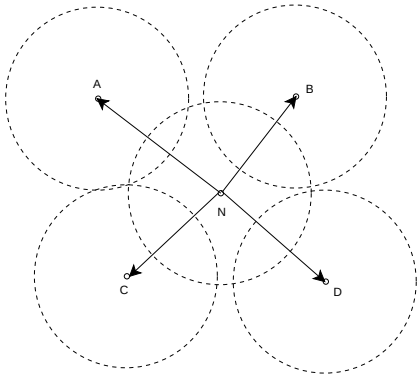


Fig. 5. Computing V_i for packet p_i where $\mathrm{Src}(p_i) = \mathrm{N}$

As an example, consider Fig. 5, where node N is equal to $\mathrm{Src}(p_i)$, and nodes A, B, C and D are source nodes for four packets in the virtual queue. $V_i = \{\overrightarrow{NA}, \overrightarrow{NB}, \overrightarrow{NC}, \overrightarrow{ND}\}$. The resultant vector V_i^{sum} is then equal to the sum of the vectors in V_i, i.e. $V_i^{sum} = \overrightarrow{NA} + \overrightarrow{NB} + \overrightarrow{NC} + \overrightarrow{ND}$.

There are a few important points to note in the computation. First, instead of measuring overlap, we used *spread* as an approximation. $|V_i^{sum}|$ is a measure of how source nodes of other packets are distributed around $\mathrm{Src}(p_i)$. When nodes are "evenly distributed" around N, we assume that the overlap is large. Correspondingly, the value of $|V_i^{sum}|$ is small. When nodes are clustered on one side of N, we assume that overlap is small and the $|V_i^{sum}|$ value is large. $|V_i^{sum}|$ is not unique for a specific node placement. For example, when two nodes are equal distance from and on opposite sides of N, $|V_i^{sum}| = 0$ independent of how far they are from N. However, assuming that nodes are randomly placed, such symmetry is unlikely. In addition, due to the aggregate nature of the vector sum, a small number of elements in V_i is sufficient. A large number of elements do not necessary make the approximation more accurate because the nodes are randomly placed. Second, only packets in the virtual queue is used for computing $|V_i^{sum}|$. These are packets that have already been transmitted and the decision cannot be undone. On the other hand, the packets in $\mathrm{Re}(Q)$ can change due to buffer management.

When transmitting a packet, CT selects a packet p_i such that $\forall p_j \in \mathrm{Re}(Q) \setminus \{p_i\}, |V_i^{sum}| \geq |V_j^{sum}|$.

D. Packet Timeout

As we are interested in the coverage offered by the sensor network in the last T_c time period, packets that are older then T_c will not contribute to the coverage. We drop packets that are older then T_c from the real queue. For the virtual queue, we set the timeout period to $1.5T_c$.

V. Results with exact coordinates

In this section, we study the performance of our buffer management and scheduling algorithms under the assumption that the sensor nodes know their exact geographical coordinates.

A. Simulation Setup

The simulation was conducted with a version of Glomosim[1] modified to include our buffer management and scheduling protocols.

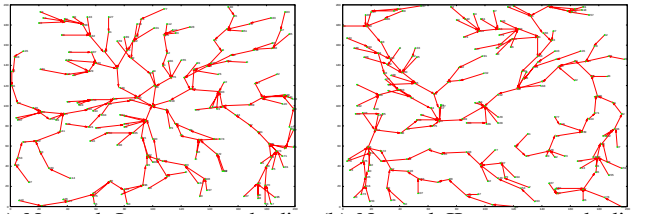
Number of nodes	200
Simulation area	$200\times200m^2$
Location of sink	(100, 100)
Transmission range	30 m
Sensing range, γ	20 m
Event generation rate	1 packet every 10s
Packet timeout, T_c	30s
Virtual queue length, ν	24 packets

TABLE I
Parameters for Glomosim simulation

Table I shows the parameters that are used in the Glomosim simulation. With the parameters chosen, the network is dense and the sensing density is 31.8. Therefore, on the average, every point is covered by 31.8 nodes. There is only one sink node and it is located at the center of the square. The packet or event generation rate

per node is 1 packet every 10s and is constant throughout the simulation period of 600 seconds. For each sensor node, the starting time of event generation is randomly distributed between 0 and 30 seconds.

We fix the size of the virtual queue to be 24. From our simulation results, this virtual queue length is sufficiently large and achieves the best possible performance. Due to lack of space, we will not include results on how coverage varies with different virtual queue sizes.



(a) Network I, average node distance = 5.73 (b) Network II, average node distance = 8.23

Fig. 6. Simulated sensor networks

Two topologies of 200 sensor nodes are used in the simulations and they are shown in Fig. 6. Static routes are computed and the average hop count of Fig. 6(a) and Fig. 6(b) are 5.73 and 8.23 respectively. Each of these networks represent possible layouts of nodes that are scattered randomly within an area. We expect Fig. 6(b) to be the typical network that occurs in a cluttered environment, where the large number of obstacles block off radio signals and interfere the the line-of-sight communication of nodes. Fig. 6(a) would be typical in a more open area, with minimal radio interference. In general, packets in Network II have to travel further to reach the sink.

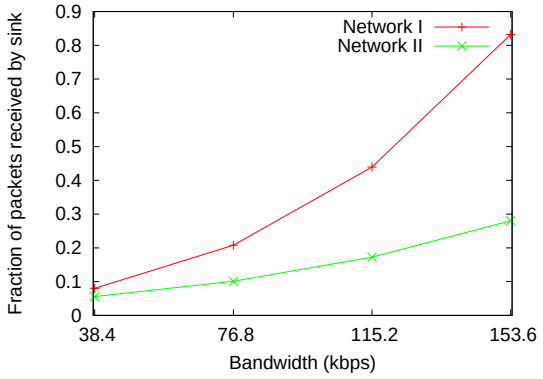


Fig. 7. Fraction of packets received by the sink

Generally, for the same network bandwidth, Network I experiences lower packet loss than Network II. Figure Fig. 7 shows the packet delivery ratio for different bandwidth. For both networks, with 38.4kbps, the packet delivery ratio is below 10%. When the bandwidth is 153.6kbps, Network II still delivers only 30% of the packets, while Network I delivers more than 80% of the packets. Note that these results are independent of the buffer management and scheduling algorithms used, as long as there is no early drop and scheduling is work-conserving. Nevertheless, buffer management and scheduling algorithms will make substantial difference, especially in Network II where the loss is high, when the objective is to maximize network coverage.

In the simulation, we compare the following 4 combinations of buffer management and scheduling algorithms, namely:

C1: drop-tail and FIFO
C2: MRD and FIFO (labeled MRD in the figures)
C3: drop-tail and coverage transmit (labeled CT in the figures) and
C4: MRD and CT (labeled MRD+CT in the figures).

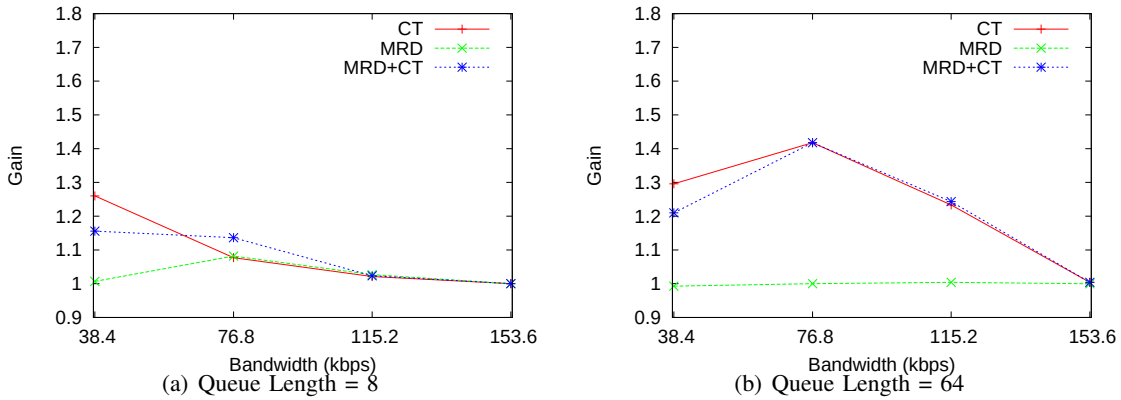
B. Performance Metric

Before presenting the results, we would like to highlight the performance metric used. The entire simulation interval is divided into disjoint intervals of T_c seconds. We define *timed coverage* for any T_c interval as the percentage of the sensor field that is covered by packets received by the sink in the interval and that are generated less than T_c seconds ago. This metric measures the network's ability to provide up-to-date data for the area under surveillance. *Mean timed coverage* is the average of all the timed coverage values measured and is the key performance measure used. For ease of comparison, we focus on the *coverage gain*, defined as the ratio of the mean timed coverage achieved by the buffer management and scheduling algorithms relative to that achieved by drop-tail and FIFO. (i.e. We plot the ratio of the coverage achieved by combinations C2, C3 and C4 to that achieved by combination C1).

C. Simulation Results

In the simulation, we vary the *data queue length* D from 2 to 64 packets and the bandwidth (β) from 38.4kbps to 153.6kbps. Due to lack of space, only results for queue size of 8 and 64 packets will be shown.

Fig. 8 shows how coverage gain varies with bandwidth for Network I and queue sizes of 8 and 64. For smaller queue size and bandwidth, the impact of buffer management is the most significant. For the smallest buffer size simulated ($D = 2$) and at the lowest bandwidth of 38.4kbps, MRD achieves the largest gain of 1.18 (18% better than drop-tail and FIFO). For $D = 8$, the improvement varies from 0% to 5%. There is no gain for MRD when Q=64. On other hand, CT has the



(a) Queue Length = 8 (b) Queue Length = 64

Fig. 8. Coverage gain at various queue lengths in Network I

largest gain (1.25) when bandwidth is lowest at 38.4kbps and $D = 8$. Overall, the combination of MRD and CT performs the best across the entire range of buffer size and bandwidth simulated. In general, scheduling plays a more important role because the packet loss is lower in Network I. In addition, scheduling is required for the transmission of every packet while MRD is only executed when the buffer is full.

Fig. 9 shows the coverage gain achieved in Network II. Network II experiences a higher level of packet loss due to its longer average node distance to sink and the gain is larger in general. For small buffer size and lower bandwidth, buffer management is more important and for large buffer and higher bandwidth, scheduling is very useful. Hence, in the presence of higher loss, the use of coverage-aware buffer management and scheduling algorithms can improve the performance by up to 75%. Again, the combination of MRD and CT performs the best across the entire range of buffer size and bandwidth simulated.

These results show that by using only approximate spatial information, it is possible to achieve substantial coverage gains over the drop-tail and FIFO queuing schemes.

VI. Modified DV-Hop

In the previous sections, we assume that exact location is available. However, given the current state of technology, such an assumption may not be realistic. In this section, we modified the DV-Hop [7] algorithm to provide a coordinate system that is suitable for our buffer management and scheduling algorithms. The proposed modified DV-Hop algorithm assumes that exact location is not available, *even for the anchor nodes*. However, we will show that this approximate coordinate system is sufficient for the purpose of improving coverage. Before detailing our algorithm, we give a brief overview of DV-Hop localization.

A. DV-Hop

Distance Vector (DV) Hop localization relies on the existence of anchor nodes within the sensor network whose coordinates are known to all the nodes. The location of the other nodes in the network is then determined in relation to these anchor points. DV-Hop localization employs three main stages. In the first stage, each non-anchor node needs to determine its distance to each anchor node in hops. This is achieved by propagating distance vector information from each anchor using controlled flooding. At the end of this stage, each anchor node will also have the hop count to each of the other anchor nodes.

In the second stage, the anchor node uses this hop count information, along with the known euclidean distance to the other anchor nodes, to estimate the length of a single hop. Let A_{dv} be the set of all anchors. The length of a hop computed by an anchor, $a_i \in A_{dv}$ is

$$c_i = \frac{\sum \mathrm{d}(a_i, a_j)}{\sum h_j} \qquad a_j \in A_{dv}, a_j \neq a_i \tag{9}$$

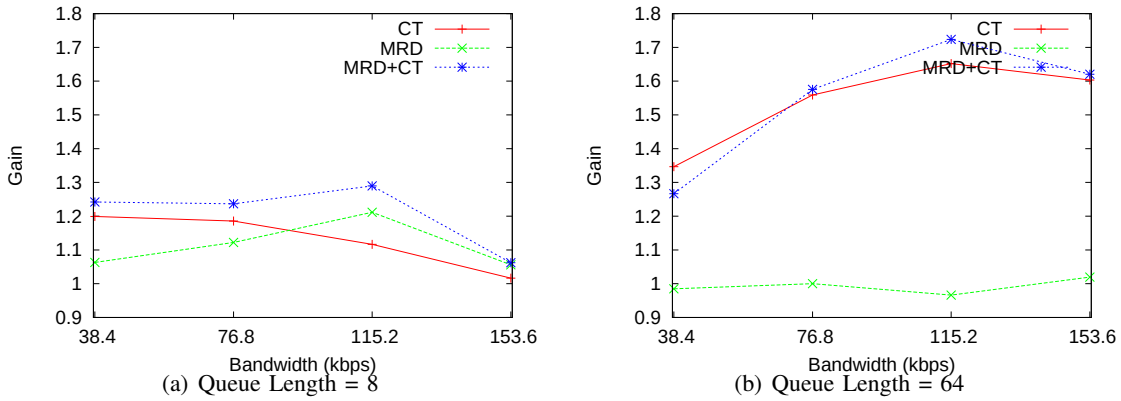
where h_j is the number of hops from a_i to a_j.

For example, in Fig. 10 the anchors are A, B and C. Anchor A computes the hop length as $(70 + 45)/(2 + 4) = 19$. Likewise, the hop length computed by B is $(45+80)/(3+3) = 20.83$ and C is $(70+80)/(2+3) = 30$.

This hop distance is then propagated to the non-anchor nodes via controlled flooding, and used by the non-anchor nodes for trilateration to estimate their positions.

B. Modified DV-Hop

From the results of previous sections, we observed that the buffer management and scheduling algorithms can operate with only *proximity information* instead of



(a) Queue Length = 8 (b) Queue Length = 64

Fig. 9. Coverage gain at various queue lengths in Network II

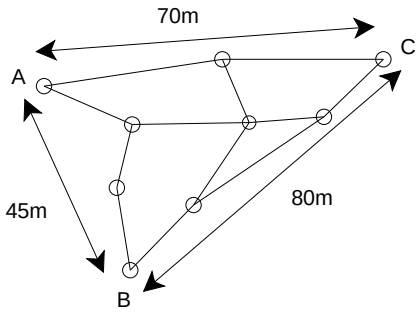


Fig. 10. DV-Hop example. Nodes A, B and C are the anchors used to localize the other nodes.

the absolute location of nodes. All we need is a way to determine if two nodes are close enough such that the correlation between their sensor data is high. In the modified DV-Hop described in this section, we assume that even the anchor nodes have no coordinate information. This is the most flexible scenario and would provide the worst case scenario. If the location of the anchor points are known, then we can use a localization algorithm like DV-Hop to obtain the approximate location of the nodes.

Our modified DV-Hop algorithm consists of an anchor-selection algorithm as well as only the first stage of the original DV-Hop as described in Section VI-A. Two messages are required for anchor selection and localization: ANCHOR and LOCATE. Anchor selection (using the ANCHOR message) requires an existing route from sink to each source node. The proposed localization relies on controlled flooding to disseminate LOCATE messages throughout the network, and is independent of the route.

The ANCHOR message contains two pieces of information: the current hop count h_c, which is updated every time the packet is forwarded by a non-sink node, and the threshold hop count h_m. h_m is used to influence anchor placement: by using a larger value for h_m, the anchor nodes will be further away from the sink.

The sink initially unicasts η ANCHOR messages to the nearest nodes, where η is the number of anchors to be selected. The messages are uniformly distributed over all the neighbors.

At each non-sink node, h_c is first incremented by one. If the non-sink node is a leaf, then it assumes the role of an anchor node immediately. For non-leaf nodes, if $h_c < h_m$, the node then forwards the ANCHOR message to only one of its children. If $h_c \geq h_m$, the non-leaf node forwards it to one of its children if it has previously received a LOCATE message indicating that it is less then or equal to 2 hops away from some other anchor node and it is not itself an anchor. Otherwise, this non-leaf node assumes the role of an anchor.

Once an anchor is determined, it floods the network with LOCATE messages to construct a shortest path tree with itself as the root. All the other nodes in the network will know their distance, in hops, to this anchor. This process is performed for each anchor. Since η messages are sent by the sink, η anchors will be elected.

At the end of the modified DV-Hop localization, each node will have the hop distance to each anchor. Each anchor is assigned its own unique *anchor id*, from 1 to η. The hop-based coordinates assigned to each node is of the form $(x_1, x_2, \cdots x_\eta)$ where x_1 is the hop distance to the anchor with ID 1, x_2 is the hop distance to the anchor with ID 2 and so on. The new coordinate system based on hop-count is used in the buffer management and coverage transmit scheduling algorithms by treating these coordinates as higher dimension Euclidean space.

C. Sensing Distance in Hops

In both MRD and CT, it is necessary to consider only nodes that are within sensing coverage of each other. However, by using the new hop-count based coordinate

system, the maximum separation between two overlapping nodes is unlikely to be 2γ. Since node coordinates are now specified in hops, there is a need to determine the value of γ_{dv}, the sensing range of the node, in number of hops as well. The coordinate system imposed on the sensor network depends on the number of anchors used and this in turn affects the Euclidean distance, in hops, between any two nodes. We evaluate γ_{dv} via simulations.

For each anchor count $\eta = 2, 4, 6$ and 8, we randomly created 40 different sensor networks, each with the parameters shown in Table I. For every network, we ran the anchor selection and localization algorithm as described in Section VI-B 20 times for each $h_m = 4, 8, 12$ and 16. This results in a total of 800 iterations of the anchor selection and localization algorithm for each value of h_m. We record the γ_{dv} values of all pairs of nodes that are within 2γ of each other, and compute the expectation $E(2\gamma_{dv})$ for each value of h_m.

	h_m			
η	1	4	8	12
2	1.12	1.17	1.17	1.19
4	1.72	1.96	1.92	1.94
6	2.23	2.66	2.69	2.68
8	2.71	3.25	3.32	3.27

TABLE II

EXPECTED DISTANCE IN HOPS, $E(2\gamma_{dv})$ BETWEEN NODES LESS THAN 2γ APART.

The results of the simulation are shown in Table II where each entry gives the expectation of $2\gamma_{dv}$. For example, with $h_m = 1$ and $\eta = 2$, for all pairs of nodes that are within sensing range of each other, the average distance measured in hop count for all these nodes is 1.12 hops.

In the algorithm, $E(2\gamma_{dv})$ is used as the threshold to determine if two nodes are within sensing range. For example, if $h_m = 8$ and $\eta = 8$, we assume that two nodes with a distance of 3 hops apart are within sensing range of each other and two nodes with a distance of 4 hops apart is beyond sensing range of each other. Next, we try to determine the accuracy of this classification and the results are shown in Table III. In each entry, the first value is the accuracy of the classification and the second value is the false negative percentage. For example, when the threshold of 1.96 for $h_m = 4$ and $\eta = 4$ is used, 60% of the pairs classified are actually within sensing range and among those classified as out of sensing range, 5% of the pairs are actually within sensing range.

We see that for any particular value of η, $E(2\gamma_{dv})$ is the smallest for $h_m = 1$, while $E(2\gamma_{dv})$ at higher values of h_m are largely similar. This behaviour can also be observed in Table III, where for the same η, the accuracy at $h_m = 1$ is the lowest. In addition, for the same value of η, $E(2\gamma_{dv})$ is fairly constant for values of h_m from 1 to 12. Increasing η increases $E(2\gamma_{dv})$ as well as its accuracy. This suggests that the *number of anchors* in a sensor network is important in determining the accuracy of $2\gamma_{dv}$ while the placement of these anchor nodes has a much smaller influence. The best accuracy (76%) is attained at $\eta = 8$ and $h_m = 8$ or 12.

	h_m			
η	1	4	8	12
2	24(7)	33(6)	34(6)	36(6)
4	46(6)	60(5)	61(5)	63(5)
6	58(6)	72(5)	73(5)	72(5)
8	69(5)	75(5)	76(4)	76(4)

TABLE III

PERCENTAGE ACCURACY OF $E(2\gamma_{dv})$. EACH ENTRY IS GIVEN AS %ACCURACY(% FALSE NEGATIVE)

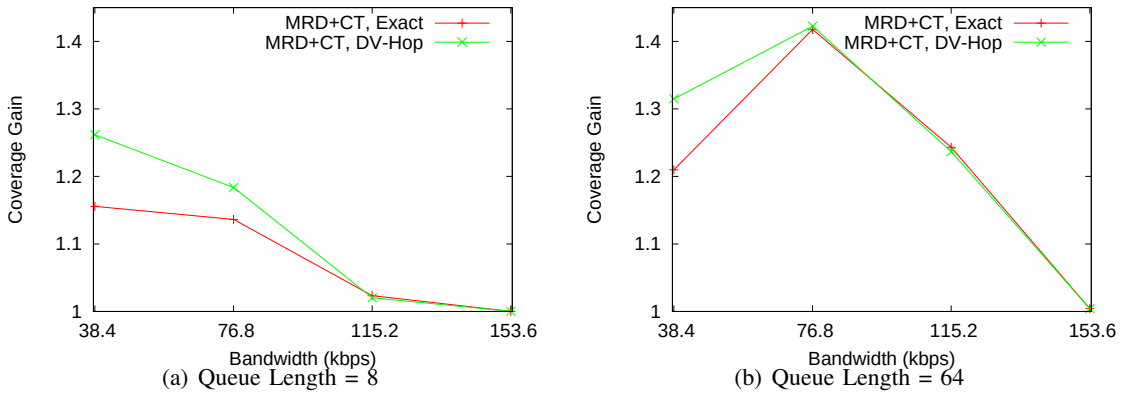
It is important to note that the use of such a threshold relies on the assumptions that (1) the sensor nodes are randomly placed (2) the node density is known and (3) the number of nodes deployed is known. With these information, $E(2\gamma_{dv})$ can be computed off-line given η and h_m. Since no localization information is required, these assumptions are fairly reasonable.

D. Simulation Results

In the simulation, we compare the results achieved by using the modified DV-Hop localization and actual location. Our simulation is conducted using the parameters in Table I with some modifications: $\gamma_{dv} = 2$ ($2\gamma_{dv} = 4$) is used instead of γ and η is set to 8. Therefore, we conduct the simulation with the most accurate $2\gamma_{dv}$ among all the combinations of η and h_m studied. For brevity, we only run our simulation over Network I.

The coverage gain when γ_{dv} is used is shown in Fig. 11. Note that the average timed coverages of using drop-tail and FIFO at all bandwidths are the same, since their operations do not depend on either γ or γ_{dv}.

Fig. 11 shows that for the same β, the differences in coverage gain between the simulation using exact coordinate and the one using DV-based localization is small. As the queue size D increases, the gap in coverage gain decreases. At $D = 64$, the coverage gains are almost identical. In fact, in some cases, use of DV-hop actually improves the performance slightly. We can explain the result in the following way. Recall that



(a) Queue Length = 8 (b) Queue Length = 64

Fig. 11. Coverage gain at various queue lengths in Network I with $\gamma_{dv} = 4$ and $\eta = 8$

CT does not use exact location information. Instead, a notion of spread or node distribution is used to reduce computation complexity. As a result, a coordinate system that is based on hop count may perform better if it is able to better correlate spread with the vector V_i^{sum}.

In conclusion, in spite of the fact that the modified DV-Hop algorithm does not make use of any pre-configured anchors and provide only approximate localization, the performance of the buffer management and scheduling algorithms are comparable to the case when exact locations are available.

VII. Conclusion

In the paper, we present the algorithms *Most Redundant Drop* (MRD) and *Coverage Transmit* (CT). The use of MRD and CT achieves the best coverage gain under most circumstances. We also demonstrated that approximated localization loosely based on the DV-Hop algorithm is sufficient for our buffer management and scheduling algorithms. With 8 anchors and $2\gamma_{dv} = 4$, the performance of scheduling algorithms under DV-based localization is very close to the cases when the exact coordinates are known. This shows that perfect localization is not required for an improvement in average timed coverage.

References

[1] "Glomosim: Global mobile information systems library." [Online]. Available: http://pcl.cs.ucla.edu/projects/glomosim/

[2] C. T. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 148–161.

[3] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2003, pp. 81–95.

[4] L. Hu and D. Evans, "Localization for mobile sensor networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2004, pp. 45–57.

[5] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks." in *SenSys*, J. A. Stankovic, A. Arora, and R. Govindan, Eds. ACM, 2004, pp. 134–147.

[6] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks." in *INFOCOM*, 2001, pp. 1380–1387.

[7] D. Niculescu and B. Nath, "Dv based positioning in ad hoc networks." *Telecommunication Systems*, vol. 22, no. 1-4, pp. 267–280, 2003.

[8] S. Pollin, B. Bougard, R. Mangharam, F. Catthoor, I. Moerman, R. Rajkumar, and L. V. der Perre, "Optimizing transmission and shutdown for energy-efficient real-time packet scheduling in clustered ad hoc networks," *EURASIP J. Wirel. Commun. Netw.*, vol. 5, no. 5, pp. 698–711, 2005.

[9] Y. Sankarasubramaniam, Ö. B. Akan, and I. F. Akyildiz, "Esrt: event-to-sink reliable transport in wireless sensor networks." in *MobiHoc*. ACM, 2003, pp. 177–188.

[10] E. Uysal-Biyikoglu, B. Prabhakar, and A. E. Gamal, "Energy-efficient packet transmission over a wireless link," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 487–499, 2002.

[11] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "Coda: congestion detection and avoidance in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 266–279.

[12] E. W. Weisstein, "Square line picking. from mathworld–a wolfram web resource." [Online]. Available: http://mathworld.wolfram.com/SquareLinePicking.html

[13] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Trans. Sen. Netw.*, vol. 1, no. 1, pp. 36–72, 2005.

[14] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Peas: A robust energy conserving protocol for long-lived sensor networks." in *ICNP*. IEEE Computer Society, 2002, pp. 200–201.

[15] H. Zhang, A. Arora, Y. ri Choi, and M. G. Gouda, "Reliable bursty convergecast in wireless sensor networks," in *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM Press, 2005, pp. 266–276.