Dynamic Service Execution in Sensor Networks

Lei Chen[1], Zijian Wang[1], Boleslaw Szymanski[1], Joel Branch[2], Dinesh Verma[2], Raju Damarla[3], and John Ibbotson[4]

[1]Dept. Computer Science, RPI   [2]IBM Watson Research Center   [3] U.S. Army Research Laboratory [4]Emerging Technology Services, IBM

Troy, NY, USA   Hawthorne, NY, USA   Adelphi, MD, USA   Hursley Park, Winchester, UK

Abstract

Sensor networks face a number of challenges when deployed in unpredictable environments under dynamic, quickly changeable demands, and when shared by many partners, which is often the case in military and security applications. To partially address these challenges, we present a novel target tracking algorithm that can be deployed on various sensor nodes and invoked dynamically when needed by the presence of targets. We also demonstrate that an auction-based mechanism can be used to provide efficient and localized wireless sensor network (WSN) congestion management for bursty traffic of abstract services based just on user-assigned priorities to different services and the quality of information provided by the services. We present results from using this auction mechanism to resolve congestion caused by packets from competing target tracking missions.

1. Introduction

Steady advancements in integrated circuitry and wireless communications, fabrication cost reductions, and innovative high-value applications make wireless sensor networks (WSNs) a significant component of pervasive computing. These networks, comprised of small, autonomous, tetherless sensor nodes, have played an essential role in closing the void between computers and the physical world by enabling continuous nonintrusive observation of real-world phenomena, which was impossible in the past.

In the current state of the art, WSNs are typically viewed as a resource-constrained collection of sensing and information fusion assets connected in an ad-hoc manner. The considerations of energy and bandwidth limitations are paramount and these aspects have been studied extensively. However, there has been insufficient investigation into aspects such as services and their composability in sensor processing elements.

Some suggest that a service-oriented approach to facilitating the operations of sensor networks could provide a better approach to making more usable and human friendly sensor networks [1]. In this paper, we investigate the challenges involved in defining dynamic services on a sensor network and introduce our initial solutions to address some of those challenges, namely, defining flexible, parameterizable dynamic services and considering Quality of Information in efficient execution of those services.

A service, in our terminology, is an abstraction provided by a collection of sensors. The service abstraction is associated with a description, which can be stored, advertised and retrieved from a registry. This description provides an interface by which users of the service can invoke and use it. Some examples of services provided by a sensor network include target tracking, intruder detection, and event detection. The basic services form a foundation for dynamically creating higher level services that are composed by taking the capability of sensors and other Information Surveillance and Reconnaissance (ISR) assets, and exposing them as a more abstract, well-defined interface.

Defining the functionality of a sensor network by means of abstractions like services has a number of benefits. Services provided by different sensor networks can be composed easily at the service level and made independent of the specific nature of the sensors that are deployed in an environment. When sensors deployed by two organizations in the same area need to coordinate their operation, a service-oriented approach enables the coordination and integration at a higher level abstraction without interfering with the specific implementation details of the two networks. Specifically, in the context of coalition operations, such an approach enables the sharing of information and functionality between two sensor networks, each implemented according to the policies and best practices of different coalition partners.

In summary, a service is a discoverable software resource, which has an advertised service description. The service description is available in a repository, called a registry that can be searched by a potential consumer of the service. Given the service description, a service consumer can bind to and use the service.

Despite the promise of truly dynamic service, discovery and binding processes, services today tend to be used in a more static way, wherein a palette of available services is made available in a design tool and a developer selects the services that they require to implement a business process. The services are then combined into an application. This composability means that a service can be assembled by combining a number of existing services. In a way, it is a fractal structure in that services can themselves consist of services. While these may be acceptable within a highly managed enterprise IT environment, within a distributed sensor network, the implementation of a Service Oriented Architecture (SOA) must be more agile [1]. First, services must be described in a way that allows algorithmic composition through semantic inferencing about the composed service properties, notably the characterization of sensors to allow them to participate in composed services that reflect mission information gathering requirements. Current research in the semantic Web services community provides a starting point, but it is not yet sufficiently advanced to implement the kind of systems envisaged for future military and security needs. Second, sensor services will exist in unreliable, low resilience network infrastructures, such as mobile ad hoc network (MANETs), and may have limited lifetime. Hence, protocols that support

unreliable and/or temporal service infrastructures, particularly distributed registry support, must be developed. Third, a security infrastructure that allows controlled access to services that exhibit the properties described in this paper must be developed. Services must provide information on a post-before-processing basis to all accredited users and resources based on data rather than application standards. Information must be available to whoever requires it whenever it is required. Fourth, the SOA must not constrain how command and control is implemented. It must also support the creation and management of ad-hoc relationships formed by members of a coalition in order to support their mission and task objectives. At a tactical level, an SOA must support the creation of dynamic communities of interest formed to perform a particular mission or task.

In addition to services offered to users, a sensor network may choose to implement its own internal operations as a service. For example, it may choose to offer congestion management or route management as a service. Exposing an internal operation as a service provides an easy knob to manipulate and change the operation and behavior of the sensor network, and will improve its manageability and effectiveness. For instance, an administrator would be able to easily adjust congestion management to improve the power efficiency of a system, without worrying about how the details of power management work.

In this paper, we describe how a service-oriented approach can be used to build an example service, binary target tracking, and demonstrate the different steps required to implement and offer this service. We provide the same level of detail for an internal operation service, congestion management for sensor networks shared among competing and prioritized target tracking missions. This particular service considers the quality of information (which is a factor of mission priority, reporting delay, and sensing error) of reports generated by the target tracking service during an attempt to manage the bandwidth between competing missions often belonging to different organizations (coalition members). Section 2 of this paper provides a scenario that motivates the need for a service-oriented approach. Section 3 uses a binary target tracking example to demonstrate how it can be done in a service-oriented manner. Section 4 does the same, but for the case of an internal operations service for congestion management. Finally, we present our conclusions in section 5.

2. Motivating Scenario

Many important applications of WSNs result from their increasing deployment in public spaces, where they can sense the surroundings to increase public safety in urban areas, improve emergency response to accidents, or assist in maintaining stability in an area controlled during an asymmetric military operation. In many instances, the problem can be reduced to identifying and tracking suspicious events or entities. WSNs are specially used in areas where it is difficult to maintain a continued physical presence of the appropriate authorities, e.g., tracking suspects in the crowded markets of large cities. Also in many instances, the capabilities of such WSN target tracking systems would benefit more than one party (e.g., different members of a coalition operation), and hence, a resource sharing mechanism would have to be deployed. This situation introduces

important new research challenges in WSN operation, as is illustrated by the following scenario.

Suppose a city hosts a high-profile event requiring the majority of law enforcement agents to be physically present within the event's vicinity, e.g., the head of a NATO state is visiting a function in a city that is controlled by a joint U.S./UK coalition force. A multimodal WSN composed of chemical sensors, video cameras, etc., has been deployed to help monitor other areas of the city. The general goal is to identify and track suspicious and/or threatening behavior (while enforcing privacy protection whenever possible). Different agencies share the WSN to monitor different types of targets with various priorities. The local Iraqi police may be mainly interested in using the network to detect offences such as muggings, curfew violations, etc. However, the coalition officials would be tasked with monitoring for highly organized and high-magnitude threats (e.g., terrorist activity), which may, among other characteristics, be indicated by the presence of large vehicles emitting suspicious chemical fumes and driving in suspicious patterns. Furthermore, due to specific skill backgrounds, different task forces within the agency may be assigned to track different types of threats. Fig. 1 depicts a rendition of this scenario, where three vehicles are being tracked by separate users, represented more generally as applications, and updates describing the vehicles' behavior are being sent to a single aggregation point, or sink. A noticeable problem occurs as the truck and van converge within close proximity to each other and temporarily cause network congestion on nearby paths to the sink. This increased delay and potential loss of packets can easily reduce the quality of information of the targets received at the sink, whose value increases in importance in such an event since target convergence might indicate collusive behavior, which must be monitored closely. The problem is further complicated given the users described above have different and dynamic usage priorities based on both the intent of the user group and the target behavior.
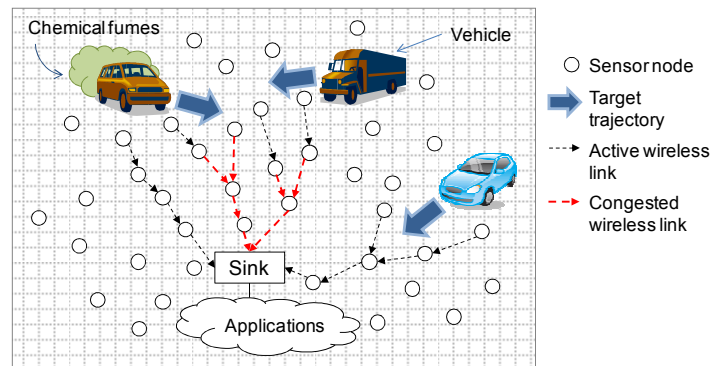

FIGURE 1. WSN target tracking scenario causing network congestion.

A similar WSN usage scenario may be found in other military applications. For example, a set of acoustic arrays deployed in an area of interest can perform multiple tasks for different command units. Acoustic arrays can detect both mortar launchings and their explosions. Each array capable of detecting the direction of arrival (DOA) of acoustic event reports the DOA, using WSN, to a central processing unit where multiple DOAs can be used to triangulate the location of the mortar launching and explosion. Mortar launching location is useful to the command unit that eliminates the mortar to prevent further launchings. Mortar

explosion sight is needed for another command unit that is sending troops to assess the damage and send the required humanitarian support. The same array system can be used for sniper detection if the sampling rate of the data is increased to detect the shockwave and muzzle blast from a supersonic gunfire. The soldiers in the vicinity would use the sniper location to take appropriate action. If there are other sensors, such as infrared cameras, in the vicinity, the flash from the rifle can be recorded and reported to determine the smoking gun.

Another example of multiple sensors using WSN is the case where an airborne acoustic array determines the location of a hostile fire on the ground and cues a visible camera to the location, while at the same time sends the hostile activity information and video to the ground protection units so they can take appropriate action. Such a system can be deployed to watch an area of interest in an urban area both to locate the hostile activity and also to track vehicles of interest approaching a check-post serving multiple missions.

Next, we address the problems brought forth by the previous scenarios. Our technical contributions are a novel target tracking algorithm deployable on various sensors, described in section 3, and an auction-based mechanism for providing efficient and localized WSN congestion management based on applications' priorities and the utility of sensed information, which, together with its application to target tracking scenarios, are discussed in section 4.

3. Target Tracking

One of the most common and important applications of WSNs is target tracking. We study it in its most basic form, assuming the binary sensing model, in which each sensor can return only information regarding a target's presence or absence within its sensing range. However, unlike most traditional approaches to binary sensing, we allow the sensors to recognize not only target's range but also a sector within the circular range around it. This assumption is justified by increasing importance of sensors with a directionally limited sensing range caused by a directional antenna or limited measurement capabilities. Examples of such sensors include cameras, infrared sensors, and directional acoustic sensors. For simplicity, we assume that either a group of sensors are collocated in a single spot providing 360° coverage or a sensor has multiple antennas or camera providing such coverage. Our goal is to have a tracking algorithm deployable for different kinds of sensors with different sensing modes, so it could be used in dynamic composition of higher level tracking services. Hence, the sensing ranges, the precision and frequency of sensing, the number, and the angular range of each sector can be set on each sensor differently to account for varying properties of sensor nodes that are at proximity to the target.

With these assumptions in mind, we introduce a novel, real-time, distributed target tracking algorithm with directional binary sensor networks. It is an extension of our previous work on omnidirectional binary sensor networks. Using simulations, we demonstrate that this new algorithm achieves high performance and outperforms other algorithms by yielding accurate estimates of the target's location. In addition, we discuss the improvement of the tracking performance resulting from providing a

directional range in addition to a distance range for the algorithm.

A number of approaches using binary sensor networks for target tracking have been proposed in recent years. The algorithms presented in [2, 3] first route the binary information to a central node and then the central node applies particle filters on information gathered from all sensors to update the target's track. In Mechitov et al. [4], each point on the target's path is computed using the weighted average of the detecting sensors' locations. Then, a line that best fits the newly estimated location and the points on the trajectory established in the recent past are used as the target trajectory. Kim et al. [5] improve the weight calculation for each sensor node that detected the target and use the estimated velocity to get the estimated target location. In Shrivastava et al. [6], both the presence and absence of the target within the node's sensing range are used to form local regions that the target has to pass. In our previous work [7], we proposed a distributed target tracking algorithm for the ideal binary sensing model. In it, each active node computes the target's location locally but uses cooperation to collect the sensing bits of its neighbors. In [8], we presented an extension of this algorithm that made it applicable to imperfect binary sensing model while keeping all the other properties of its predecessor.

All the algorithms mentioned above used omnidirectional binary sensor networks, in which each sensor can only detect the target presence or absence within its sensing range but cannot get any directional information about the target. To address this lack, we propose a novel distributed target tracking algorithm using directional binary sensor networks. Under the directional binary sensing model, each sensor node's sensing region is divided into sectors and each node can identify in which sector the target is present or absent, which gives rough directional information about the target. To the best of our knowledge, this directional binary sensor network model has not been used by any previously published algorithms for target tracking. To establish the fundamental limits of the directional binary sensing, we consider the ideal case of error-free sensing, leaving more complex analysis of the impact of errors on the tracking algorithms for the future work.

3.1. Network Model and Assumptions

The sensor network comprises N nodes placed uniformly randomly over a finite, two-dimensional planar region to be monitored. Each node has a unique identifier and its sensing region forms a disk centered at the node and bounded by a circle defined by the sensing range R. The union of sensing regions of all network nodes guarantees redundant coverage of the monitored region. Each node's sensing region is divided into sectors. Examples of four-sector directional binary sensor nodes are shown in Fig. 2 with four equal size sectors, numbered from 0 to 3 in clockwise sequential order. The initial angle of a clockwise lower radius of sector 0 with positive x-axis is selected randomly for each sensor. s denotes the number of sectors of each node and $b=log_2(s+1)$ is the number of bits needed to represent the presence of target in one of the sectors or its absence from the node's sensing range. If the b bits information changes from "0" to "1" for a specific sector of the node for the first time while it is still "0" for other sectors, the node will know the target has entered its sensing range; if the b bits information

changes from "1" to "0" for a specific sector of the node and the target does not enter any other sectors, the node will know that the target has exited its sensing range.

At the moment at which the target enters or exits the sensing range of a sensor node for the first time, that node will generate b bits of information, indicating in which sector the target is present or that it is absent from the sensing range. This b-bit status information is also updated at the moment at which the target exits a currently visited sector and enters another sector of the same sensor node. If there is no change in the b-bit status information, the node remains silent to save energy and bandwidth and avoid collisions with transmissions from other nodes. Each time a new b-bit of information is generated, the node communicates it to its neighbors, defined as nodes whose sensing range intersects its sensing range (depending on the relation between the sensing and communication radii, this may require a multi-hop transmission). Henceforward, we use the term neighbor in this specific sense.

We assume a node knows its location and the locations of its neighbors. It should be noted that each sensors estimates the position of the target relative to its own location and sensing range. Hence, to report those results, a sensor may not be aware of its geographical position and computation of the speed requires only the knowledge of its neighbors' positions relative to each node, which can be established via triangulation. Thus, establishing the geographical location of each node, although helpful, is not necessary for the introduced algorithm to work correctly. It is needed, though, to create a central trajectory of the sensed target. Establishing the location of neighbors by triangulation and then finding each node's geographical position based on some of those neighbors having GPS is a fairly standard procedure [9] that can be done at the network deployment, so it is not discussed here.

For simplicity, we assume the sensing range of each node is identical across the network and each node's sensing region is divided into the same number of equal size sectors, s. We also assume that for each sensor, the initial angle of radius "oA" is selected randomly. However, our algorithm also applies when sensing range, sector number, and sector size vary from node to node. Additionally, we assume the target moves with a velocity that is low relative to the node's ratio of sensing range to the sensing frequency. Consequently, time of discovery of a change in the target's presence or absence within the node's sensing range differs negligibly from the time the target moves within or outside this range. This assumption is reasonable as the sensing frequency for ultrasonic sensor is usually around 10^{-2}–10^{-3} seconds, while the range is in meters or tens of meters. The sensing frequency for infrared sensor is usually 10^{-4} seconds or higher, while their range is tens or hundreds of meters. Over such a short time period, most real-world target will move a very short distance, below a meter for land vehicles and a few meters for airborne vehicles. Hence, our assumptions are easily satisfied by today's sensors.

3.2. Cooperative Tracking Algorithm

To illustrate our basic idea, we use an example from Fig. 2, which shows a target moving through an area covered by two nodes whose sensing ranges are divided into four sectors. Initially, the target is outside of the sensing ranges of two nodes.
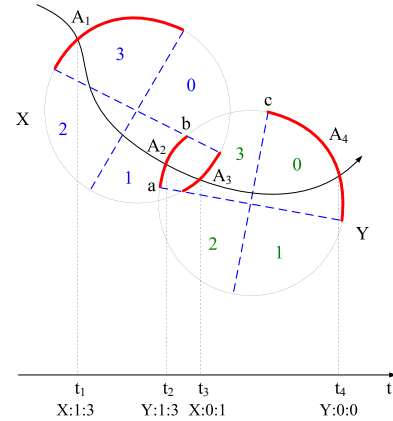

FIGURE 2. An illustration of the basic idea behind the algorithm.

Later, it moves into sector 3 of node X at the system time t_1, crosses sector 2 of node X, and enters sector 3 of node Y at time t_2. Then, it leaves sector 1 of nodes X and sector 0 of node Y, in that sequence, at times t_3 and t_4, respectively. According to the model described in section 2, each node will generate information at the time it first senses the target's presence and later at the time when it first no longer senses its presence, which corresponds to the times at which the target enters and exits sensing range of the node. Besides those times, each node will also generate information when the target leaves one sector and enters another sector of the same node.

Consequently, at the transition time t_j, the target must be on arc A_j which is a part of the arc of the corresponding sector of the node reporting the information. Hence, arc A_j can be determined cooperatively from b-bit information reported by the neighbors of that node. Let's consider arc A_2 defined at time t_2, as an example. Before time t_2, node Y will receive three messages from node X, which we can mark as "X13", "X3-2", and "X2-1". "X13" means that node X first senses the target ("1" stands for presence, "0" stands for absence) in sector 3. "X3-2" means that the target within sensing range of node X leaves sector 3 and enters sector 2 of node X. "X2-1" means that the target within sensing range of node X leaves sector 2 and enters sector 1 of node X. At time t2, node Y senses the target presence within its sector 3 for the first time, so the target must be on arc "abc", which corresponds to sector 3 of node Y. At that time, node Y also knows that the target is within sector 1 of node X. Thus, node Y concludes that the target must be on arc A2. It is important to observe that, by using this method, the two-dimensional uncertainty of the target's location on the plane is reduced to a one-dimensional uncertainty within the circle section. The shorter this circle section is, the smaller the uncertainty becomes.

At the network deployment stage, each node initializes status lists of its neighbors for its four sectors. Each time a node receives information from a neighbor, it updates the status list. At the moment at which the node discovers the change in the target's presence within its sensing range (no matter which sector the target enters or exits), it identifies the arc that the target is crossing. The target location is estimated as the middle point of the corresponding arc and is broadcasted to its neighbors.
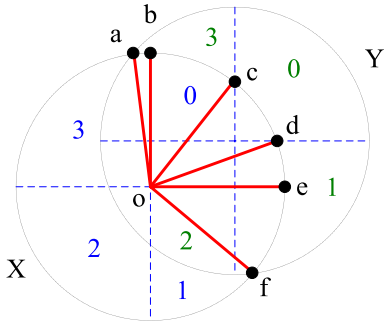
FIGURE. 3. An illustration of neighbor list initiation.

In the neighbor sector match procedure, each node finds out neighbor relations from all the sectors of a neighbor node for each of its sectors. At first, each node calculates three types of intersection points that include (1) intersection points of the sensing circles of the node and its neighbor, e.g., points "a" and "f" in Fig. 3; (2) intersection points of the sensing circles of the node and its neighbor's sector boundaries, e.g., points "c" and "d" in Fig. 3; and (3) intersection points of the sensing circles of the node and its sector boundaries that fall into the sensing area of its neighbor, e.g., points "b" and "e" in Fig. 3.

Then each of these intersections is sorted by the angle formed by one of the intersection points of type (1) and the intersection point itself, e.g., $\angle aob$, $\angle aoc$, $\angle aod$. After sorting, the sequence of intersection points in Fig. 2(right) will be "abcdef". These sorted points form a number of angles defined by each pair of points in sequence, e.g., $\angle aob$, $\angle boc$, $\angle cod$. Then, the center point of each arc corresponding to each angle is calculated and checked to see into which sector it falls. For example, the center point of the arc corresponding to $\angle boc$ falls into sectors 0 of node X and 3 of node Y, which means that sector 3 of node Y is a neighbor sector of node X's sector 0.

Next comes initialization and information update. Each node first establishes neighbor lists for each of its sectors. Each element of such list stores the neighbor node identifier, neighbor sector identifier, intersection points related to this neighbor sector, an angle corresponding to the arc defined by these intersection points, and *b*-bit status information generated by the neighbor, initialized to "0". Upon receiving information from a neighbor, the node updates the corresponding entry in the list. Sector 0 of node X in Fig. 3, for example, has three neighbor sectors: sector 0, 1 and 3 of node Y. The intersection points "c", "d" and $\angle cod$ are related to neighbor sector 0. The intersection points "d", "e" and $\angle doe$ are related to neighbor sector 1. The intersection points "b", "c" and $\angle boc$ are related to neighbor sector 3.

The procedure for location estimate proceeds as follows. When the node senses change in the status of the presence or absence of the target from one of its sector, it combines all angles in the corresponding neighbor list to determine the arc that the target is crossing. The four instances of this process are shown in Fig. 4. If sectors of both neighbors generate bits indicating the target's presence, the corresponding central angles are combined by an "&" operation that returns the intersection of the two angles. As shown in Fig. 4(a), the common angle of $\angle lo3$ and $\angle 2o4$ is $\angle 2o3$, so the node Y estimates the target location as the middle point of arc "23" when it senses that the target just moved within its sector. One special instance is shown in Fig. 4(b), where the

common angle is just one of the two angles. If one neighbor sector status indicates presence while the other indicates absence, the corresponding central angles are combined with an "–" operation that returns the angle formed by excluding the second angle from the first. For example, in Fig. 4(c) $\angle lo3 - \angle 2o4$ is equal to $\angle lo2$. In another special case shown in Fig. 4(d), the result may consist of two angles, $\angle lo2$ and $\angle 3o4$. The correct angle in this case is chosen by considering the recent estimate of the target location.
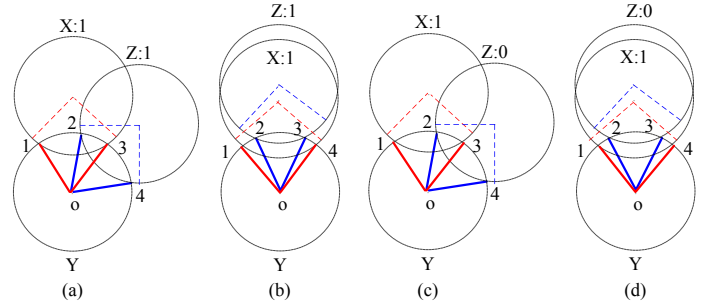

FIGURE. 4. Four instances of angle combinations.

Let *FA* be the sought arc's central angle initialized to *2π/s* (the sensing border of a sector that the target is crossing). Let *IN* be the set of neighbor sectors indicating target's presence while *OUT* be the set of neighbor sectors indicating target's absence. Then, the final angle whose corresponding arc is the one that the target is crossing can be expressed as

$$FA = FA \underset{i \in IN}{\&} angle_i \underset{j \in OUT}{-} angle_j, \qquad (1)$$

where $angle_i$ is the central angle of the neighbor's sector *i*.

It should be noted that regardless of the relative position between the two sensor nodes, the intersection of their sensing ranges must fall into one of the four cases depicted in Fig. 4, because the nodes are assumed to be Unattended Ground Sensors (UGS). Hence, our tracking algorithm will always find the correct arc that the target is crossing. For the same reason, the computational complexity of the angle combination is independent of the number of sectors or the network density. However, the space and time complexity of the algorithm initialization is linear in the number of intersection points between the reporting node and neighbor nodes, which is proportional to the network density (measured as the average number of nodes in the sensing range) and also the number of sectors of each node.

3.3. Simulations

We have designed a QT (a cross-platform application framework) based simulator that uses data exchange between multi threads to simulate the wireless communication between sensor nodes. In the simulator, we assume there are some Media Access Control (MAC) protocols providing ideal wireless communication, which means no collisions or data drops. We chose the location estimation error, measured as the ratio of the distance between the estimated and real target locations to the sensing range *R*, as the basic metric of target tracking. This metric reflects the final angle corresponding to the ring section that the target is crossing in our method, and therefore, is independent of sensing range, but should decrease with an increase of network density.

When evaluating the impact of network density on the location estimation accuracy, we kept the number of nodes fixed at 300 and varied the sensing range R from 50 to 150 units with an increment of 25 units. The velocity of the target was adjusted proportionally to the sensing range, making it constant if measured in sensing range units.

Several types of trajectories have been considered, including linear, circular, and a piecewise linear trajectory with random turns. To exclude the boundary effect, all the measured trajectories are confined within the square with sides of 800-R_{max} (R_{max}=150 is the maximum sensing range) in the center of the simulation area. For the random trajectory, the length of the trajectory is proportional to the sensing range R.

Target tracking with binary sensors was considered in [2–6]. However, papers [2], [3] and [6] all rely on a central node to gather the target information and estimate the target location, while our approach estimates the target information in a distributed way. So, we compare performance of our algorithm with the following four other distributed algorithms introduced in [4] and [5]:

(1) *Equal weight*: The target's position is estimated as the average of the detecting sensors' positions.

(2) *Distance weight*: The target's position is estimated as the weighted average of the detecting sensors' positions, where the weight for each node is set at $1/\sqrt{R_{out}^2 - 0.25(v \cdot t)^2}$ and v is the target velocity while t is the time expired since the target has been detected.

(3) *Duration weight*: The target's position is estimated as the weighted average of the detecting sensors' positions, where given the time t that expired since the node has detected the target, and the weight for each node is ln(1+t).

(4) *Line fit*: The initial estimate of target position is made as in the algorithm (2), then a line that fits the previous target positions is found and the current target position is refined using this line and the target velocity. Because algorithms (2) (3) and (4) are designed for a linear trajectory with constant velocity, our comparisons in their case are restricted to the linear trajectory. We also compare this new algorithm with the original target tracking algorithm using traditional omnidirectional binary sensor networks [7]. In comparisons, we use either 4-sector or 12-sector sensor nodes.

Fig. 5 shows the typical example of estimated location points for three kinds of trajectories. The sample network is composed of 300 nodes with sensing range of 150 units and four-sector directional binary sensors.
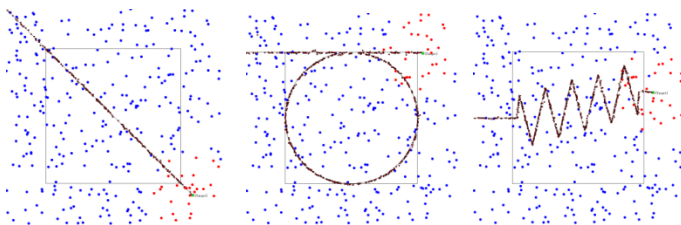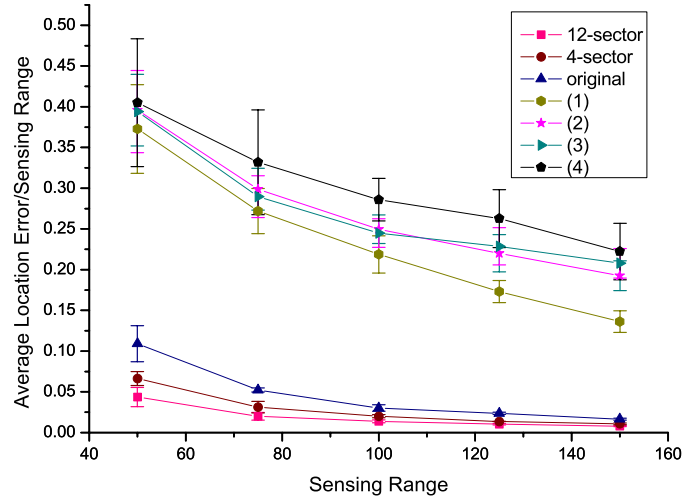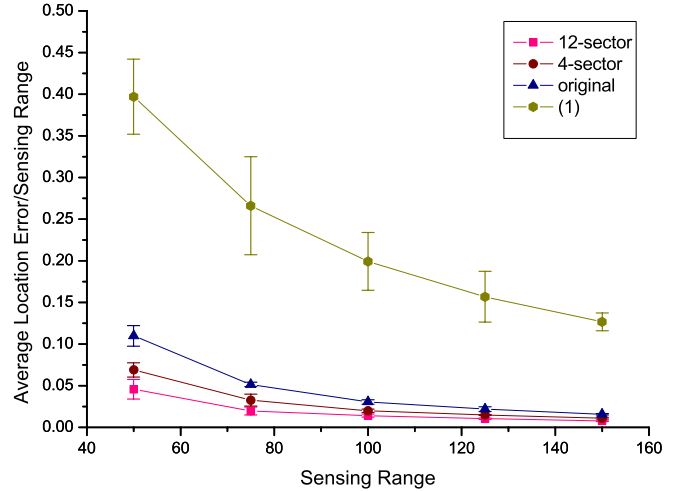

FIGURE. 5. Examples of location estimation: linear (left), circular (middle), and random (right) trajectories.
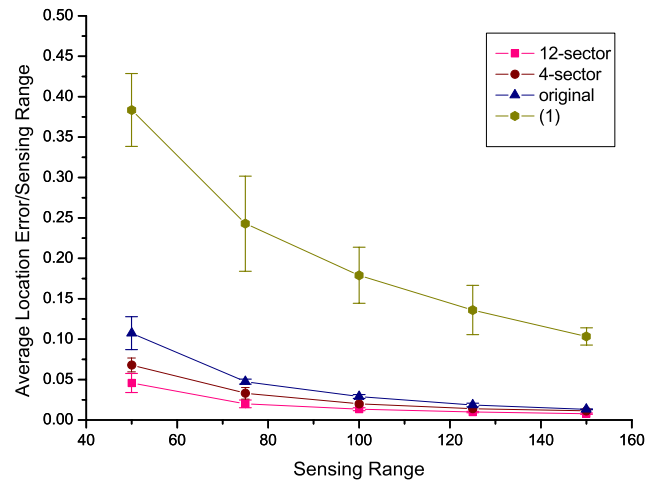
We ran each simulation setup 10 times and presented the averages of those runs and their confidence interval under a confidence level of 95%. Fig. 6 shows the location estimate accuracy results under each of the three trajectories.

(a) linear trajectory

(b) circular trajectory

(c) random trajectory

FIGURE 6. The location estimate accuracy.

In all presented cases, the 12-sector directional binary sensing produced the best results and the 4-sector directional binary sensing outperformed our original binary sensing method, which was already the best compared to any of the (1) to (4) algorithms. The ratio of accuracy of the 12-sector directional binary sensing algorithm to our original binary sensing method is nearly 2. The ratio of accuracy of the 12-sector directional binary sensing algorithm, compared to the best results of among (1) to (4) algorithms, grows from nearly 8 for an important case of network with medium density (sensing range of 50 units) to around 16 for dense networks. Additionally, the location estimate accuracies of all three trajectories of our algorithm are close to each other, demonstrating that our algorithm works well for all kinds of trajectories. Even for a sparse network with sensing range R = 50, which means that there are only three or four neighbor nodes within each sensing range, the sector directional binary sensing algorithm performs well. Our tracking algorithm is very austere in sending messages. A node generates a single message each time there is a change in the target's presence within the node's sensing range or when the sector within which the target is located changes. Hence, the communication incurred creates little burden for the MAC layer and consumes little energy.

Notably, when the number of sectors of each node is increased, only the complexity of the neighbor initialization procedure will grow because additional intersection points will appear. The complexity of angle combination remains the same, because all possible angle relations are the four instances shown in Fig. 4. The message overhead increases sublinearly with the number of sectors. For example, when the number of sectors jumps three times from 4 to 12, the average message overhead grows only by 60%.

We also calculated the ratio between the increase in the number of messages produced and the increase in accuracy when we increased the sector number from 1 (the omnidirectional binary sensor) to 4, 4 to 12 and 1 to 12. These ratios reflect the tradeoff between the gain in accuracy and the cost of sending more messages. The smaller the ratio, the more benefit we gain.

As shown in Fig. 7, the benefit is not constant. For example, if we use 4-sector binary sensor instead of omnidirectional binary sensor with sensing range of 125, the benefit is higher than for other sensing ranges. However, using more sectors to get better accuracy increases the chance of message collisions, which, by delaying message transmission, decreases the location estimate accuracy. Overall, the comparison of performance of target tracking with and without directional binary sensing can guide the sensor node designers in choosing a rational number of sectors to use. The primary factors in selecting this number are engineering (placement on the sensor node, energy supply, interference of one sensor with others, etc.) and economics (increased costs). By quantifying the benefits of directional sensing in an important application, target tracking, our results enable the designer to compute the cost benefits of this solution. Those benefits include (1) increased precision of location estimations, (2) decreased density of network deployment needed for the given accuracy of location estimation or (3) increased network lifetime with the same density and accuracy, thanks to an increased number of sleeping sensors. These benefits allow the designer to compute the change of the cost of deployment and operation of the sensor network with the given number of sectors. Then, choosing the optimal number of sectors for the application becomes a simple comparison of these costs with the number of sectors permitted by engineering considerations.
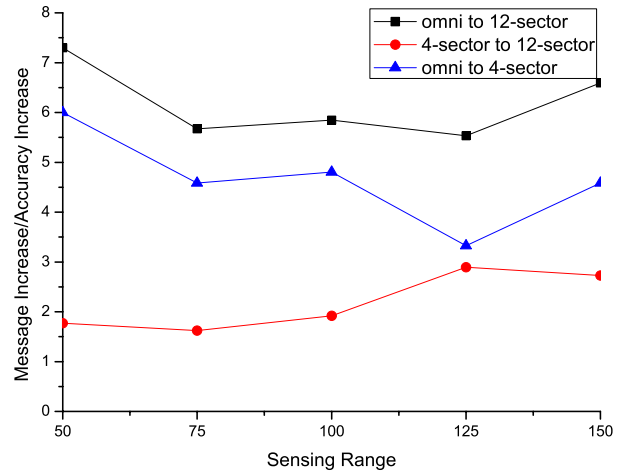

FIGURE 7. Ratio of message count increase to accuracy increase.

In summary, we present a real-time distributed target tracking algorithm for directional binary sensing networks. Extensive simulations of this algorithm performed under different configurations are reported. We observe that our new algorithm yields good performance, better than the performance achieved by our previous and other algorithms in terms of accuracy of target location estimation. Our target tracking method is directly applicable to multi-target tracking when targets are sufficiently separated from each other. As discussed previously, even when targets get close, we should be able to distinguish each target by predicting its position from past location estimates. Hence, the directional sensor helps in distinguishing multiple targets even when they get close, but before their trajectories converge.

4. Auction-based Congestion Control for Bursty Traffic

One of the important challenges arising from supporting generalized services composed of parameterizable basic services is controlling the global properties of the service. It cannot be done from within the service, because the environment in which the service is deployed is not known in advance. This challenge is particularly difficult if several similar services are competing for the limited resources of the sensor network.

Several congestion control protocols have been proposed for WSNs, including congestion avoidance [10], rate-limiting congestion control [11], congestion control and avoidance (CODA) [12] and rate control with multiple classes of flow (CoBRA) [13]. In general, these protocols are organized into three stages—congestion detection, congestion notification and traffic rate adjustment—that together create a feedback loop. The papers cited above either provide different techniques for each of the stages or consider different packet classes and/or cluster sensors differently [13]. A more general problem of fair allocation of bandwidth in computer networks is addressed in [14] in which fairness is achieved by maximizing the minimum benefit. However, all those approaches use the feedback loop-based solution, which fails when the flow paths change before the feedback loop can be formed. Such rapid change in flow paths arises in such sensor network applications as mobile target tracking or dynamic phenomena monitoring, in which the set of

nodes actively engaged in sensing changes quickly because the targets or the monitored phenomena move over time. Hence, traditional feedback loop-based congestion control approaches are inadequate for resolving the problems described in the scenario in section 2. Furthermore, the published literature mainly focuses on the static fairness of resource allocation during congestion without considering dynamic changes of priorities of applications, in particular, their changing sensitivity to packet delays. We address these issues in our approach.

More specifically, our approach to addressing these challenges is based on a distinction between the Quality of Information (QoI) [15] the service provides (and that could be measured by the user or network in an objective way) and the value of that information to the user, which, of course, is subjective (relative the user or the application that the service enables). To make these notions more concrete, in case of target tracking, the QoI can be measured as imprecision of the target location at the user site, while the value of information could be measured by the importance (priority) that the user assigns to this imprecision. Hence, this section is devoted to demonstrating how distributed auctions organized at congested nodes can, based only on QoI measurements and target priorities, arbitrate between packets carrying information about different targets to achieve the overall global goal of providing the best possible service to the sensor network in the given circumstances. What is important is that the auctions are not involved in setting either the QoI measurements or the priorities, so the mechanism proposed is general and can be used for different services and different global arbitration between abstract services.

In support of this approach, we define an information utility metric that jointly represents the QoI and the priority of each application. For simplicity, we use the product of the two, but other functions are possible. At the point of congestion, a Second Price Auction is held with bids represented by target update packets, which essentially compete for transmission slots at the congested node using utility loss as a form of currency. Hence, the problem is addressed locally at the point of congestion, minimizing the overhead of the solution. This befits our purposes, since we focus on allocating bandwidth in response to relatively transient congestion that should not require an overly complex solution.

We demonstrate the applicability of our solution by using it to implement two congestion management techniques: (1) equalizing utility loss for all applications to provide fair bandwidth sharing among all applications and (2) minimizing the sum of all utility loss values to optimize global tracking quality. We further demonstrate the solution's strength by using a simulation to compare its performance in implementing the two techniques above against a purely analytical solution and a simple bandwidth equalization solution.

We assume that for a target tracking application, the QoI with which a target is tracked is a function of the uncertainty of a target's location, which when multiplied by the priority of the application provides a suitable metric expressing the utility loss of information provided by sensors for a tracking application. To reflect such a measure, we conceptually define the information utility loss, u, of an application as $u(t) = pr(t)$, where p is an application's priority (value of p increases with priority) and $r(t)$ is the radius of a circle (or sphere in 3-D) around the *predicted*

location of a target in which the target actually resides at time t. In other words, $d(loc_p(t),loc_a(t)) \leq r(t)$, where $d()$ is a physical distance function, and $loc_p(t)$ and $loc_a(t)$ are the target's predicted and actual locations, respectively, at time t.

We assume that target report packets contain $loc_a(t_m)$ and $v(t_m)$, where v is the target's velocity and t_m is the time of measurement (assuming that an ideal tracking algorithm is used). Considering that a delay, Δt, exists between the time a target report is generated at the source sensor and the time t at which a report reaches the tracking application or an intermediate node, we *formally* need to calculate $r(t_m+\Delta t)$. However, the lack of observation of the target in the period Δt must be considered in this calculation.

We use the model of *constant speed precision prediction* to calculate how $r(t_m+\Delta t)$ changes over time. Hence, we assert that the speed computed at time t_m differs from the target's actual speed no more than $\Delta v = \alpha v(t_m)$, where α represents the speed precision prediction factor. More specifically, α represents any (combination of) feature(s) of the target that may affect its ability to change speed. For instance, this may correspond to a reasonable assumption that there is a maximum acceleration that the target can sustain. Fig. 8 can be used to illustrate how $r(t_m+\Delta t)$ is derived using the concept of constant speed precision prediction. Here, a target's predicted position, $loc_a(t_m)$ and corresponding distance traveled given the location and speed measured at t_m is shown. The target's real position, $loc_a(t_m)$, and corresponding distance (not directly observed by the application) given Δv is shown as well. The figure demonstrates that $r(t_m+\Delta t)$ can be no larger than the difference in distances and hence,

$$r(t_m + \Delta t) \leq | v(t_m)\Delta t - (v(t_m) + \Delta v)\Delta t | = \alpha v(t_m)\Delta t .$$
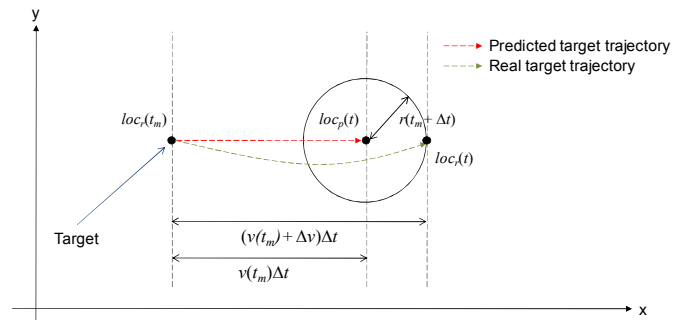
FIGURE. 8. An example relating target velocity to $r(t_m+\Delta t)$.

Hence, $r(t_m+\Delta t)$ is linearly proportional to both $v(t_m)$ and Δt. We note that in the most general case, the stated precision prediction factor, α, should include a factor of both the imprecision of the employed tracking algorithm and some measure of the unpredictability of the target's behavior.

In summary, we consider a model of quality of target tracking that assumes constant speed precision prediction and yields a utility loss metric for sensor information for a target tracking application as follows:

$$u(t) = \alpha p v(t_m) \Delta t \qquad (2)$$

Now, utility loss can be measured using a target's velocity and the time delay between when the velocity was measured at the sensor and received at the application. For simplicity, we set α=1,

which can easily be achieved by properly scaling the missions' priorities.

4.1. Auction-Based Congestion Management

We use an auction-based mechanism to prioritize the forwarding of target report packets in the event of network congestion. When there is no congestion, applications accumulate different utilities since they have different priorities and track targets with likely different speeds, and their target report packets experience different (and largely independent) end-to-end delays (see (2)). Specifically, we employ auctions to provide two different utility loss management solutions: (1) equalizing utility loss across all tracking applications and (2) minimizing the total utility loss of all applications.

TCP congestion detection methods are used in traditional networks, where congestion is observed or inferred at the end nodes based on a timeout or redundant acknowledgments. However, for efficiency, proactive methods are preferred in WSNs. For simplicity, in our setting, we assume that congestion is detected based on a node's outgoing transmission buffer occupancy. Specifically, congestion is detected locally when the packet queue is fully occupied.

We assume that a Spatial TDMA scheme [16] has been deployed in the sensor network to prevent transmission interference between nodes. Here, each node uses one of n slots for transmission over the period t_c; hence, a packet's transmission time is t_c/n. The value of n depends on the type of TDMA scheme (i.e., global or local) and on the topology of the network. The upper bound of n for a given maximum number of neighbors of a node has been previously established [16]; hence, we do not pursue defining a value of n in this work.

Under our scheme, we assume that target reports packets for a application i carry several items of essential data beyond that required by the tracking application: (1) the time at which the target report was generated, $t_{m,i}$; (2) the target's speed, $v_{m,i}$; and (3) the priority of the application tracking the target, p_i. We assume that when a target is first detected, the base station associates it with a relevant application and the application's priority is forwarded back to the node that detected the target. As previously mentioned, the application priority is also propagated among nearby tracking sensors each time a report is generated.

Any time a given node has multiple target report packets for different applications to be forwarded for its current transmission slot, the node conducts an auction to assign the slot to the packet with the highest bid. Thus, the bidders are represented as packets awaiting transmission and their bids are defined by the predicted information utility loss of the applications with which they are associated. . To simplify bidding, we used the Second Price Auctions [17], in which bidding the true value of the traded goods is the optimal strategy for each bidder. The auctions that we have designed are recurrent [18] since they are conducted repeatedly with bidders represented by different reports of each application. Unlike a single auction in which a winner acquires the entirety of a resource indefinitely (the transmission time slot in our case), in a recurrent auction with a participation incentive mechanism [19], it is possible to share resources over time. This, in turn, prevents resource starvation of any auction participant. The use of a Second Price Auction with a participation incentive also helps simplify bidding strategies.

We also note that if a target report packet losses an auction for the currently associated transmission slot and a new packet for the same application arrives at the congested node, the new packet (with the most recent value of $t_{m,i}$) replaces the old one in the queue of messages awaiting transmission. This is because the delivery of the less recent packet does not reduce the application's utility loss of information, even if both packets are delivered together. As a result, in each auction, at most only one packet, i.e., the most recent report packet, associated with a given application will participate.

Fig. 9 presents a more illustrative description of the auction-based congestion management framework. As shown, the auction happens recurrently at the congested nodes that receive combined upstream traffic carrying multiple application target report packets. The auction is repeated for each transmission slot assigned to the node to select the single target report packet that should be allocated to the slot. This packet choice is made is made in such a way as to fulfill the overall system goal. In subsection 4.2, we describe how the auction is defined and show how easily the presented approach can be adjusted for a specific goal.
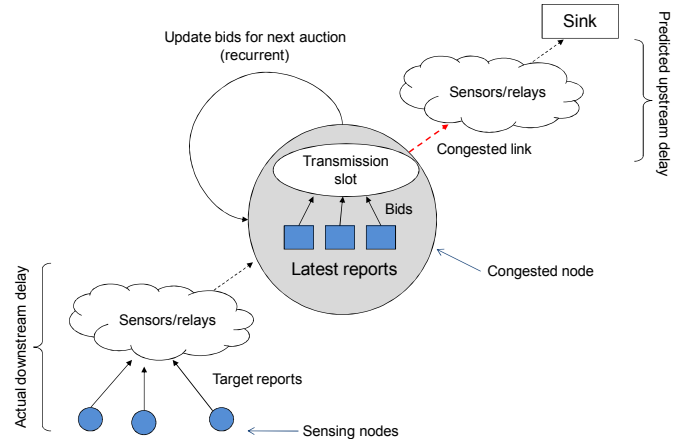

FIGURE 9. Auction-based bandwidth management framework.

4.2 Auction Mechanism

The design goal of the auction mechanism is to either equalize or minimize the actual information utility loss, $u(t)$, for competing applications at the sink. According to (2), knowledge of the delay between which a packet is generated and received at the sink should always be maintained in order to calculate $u(t)$ as observed by the sink. However, auctions are conducted at the nodes experiencing congestion, not at the sink. Hence, the *predicted* average information utility loss must be computed at these nodes over the time of congestion to approximate the observation of $u(t)$ at the sink. To enable the congested node to compute the predicted $u(t)$ (hereafter referred to simple as $u(t)$), we use the assumption that each node knows its distance in relay hops to the sink as well as the average delay (without congestion) at each hop. Under this assumed *routing protocol* with global TDMA slot allocation, a one-hop delay is simply $(n/2+1(t_c/n)$, because in addition to the transmission delay, the report packet will wait, on average, half of the period t_c for the transmission slot. Therefore, each node can compute the expected delay of its packets transmitted upstream to the sink without congestion; we denote this value as t_{up}. Under the assumed routing protocol with global TDMA, t_{up} should be constant over time. Under our scheme, the cumulative product of congestion time and $u(t)$ for each application is maintained at each auction node. The remainder of

this section explains how the values stated above are used to calculate the predicted average $u(t)$ at the auction node, as well as support the definitions of bids and auctions strategies.

We start by considering the first report packet that needs to compete for a transmission slot at the congested node and assume that congestion is observed at the sink after the first such packet arrives there. We denote t_{start} as the time at which congestion at a given node starts, so $t_{start} = t_{first\_slot\_i} + t_{up}$, where $t_{first\_slot\_i}$ is the end of the transmission for the first slot for which packets associated with application i compete. For simplicity, the cumulative products of congestion time and information utility loss for all applications participating in this first auction are set to 0.

We now consider all subsequent auctions. We first define the updated delay at the sink as caused by the previous auction, t_f, as $t_f = t_{up} + t_{slot} + t_{mp,i}$, where t_{slot} is the end of the current auction time slot and $t_{mp,i}$ is the measurement time of the previous target associated with application i. If a report packet loses the current auction, then at t_{slot}, the cumulative product of time and predicted information utility loss for the corresponding application will be increased by the following value:

$$\Delta u_i^l = \int_{t_f}^{t_f+t_c} u_i(t)dt = \left(t_{up} + t_{slot} + \frac{t_c}{2} - t_{mp,i}\right)p_i v_i t_c \ . \quad (3)$$

Otherwise, if a packet wins the auction, its increase of the product in each auction after the win, regardless of whether there is a packet for this application waiting or not and only until the first loss, is

$$\Delta u_i^w = \int_{t'_f}^{t'_f+t_c} u_i(t)dt = \left(t_{up} + t_{slot} + \frac{t_c}{2} - t_{m,i}\right)p_i v_i t_c \ , \quad (4)$$

where, $t'_f = t_{up} + t_{slot} + t_{mp,i}$. The cumulative product of congestion time and predicted information utility loss is used to compute the packet's bid for the current transmission time slot. The bids for the auctions made on behalf of an application i are defined as follows:

$$b_i = \frac{u_i + \Delta u_i^l}{t_{slot} + t_{up} - t_{start}} \quad (5a)$$

$$b_i = \Delta_i^l - \Delta_i^w = (t_{m,i} - t_{mp,i})p_i v_i t_c \quad (5b)$$

where equation (5a) is used under the system configuration chosen to equalize tracking applications' information utility loss and equation (5b) is chosen to minimize the total utility loss for all applications. Both bid values include a participation incentive mechanism. After the auction is completed, all applications adjust their utility loss values by equations (3) or (4) according to whether they won or lost.

The bid in equation (5a) represents the current predicted average utility loss for an application after losing the current auction. Here, the auctioneer (i.e., congestion node) attempts to equalize the utility losses of applications by selecting the target report packet with the highest bid as the winner. As a result, the winner's utility loss increases more slowly after winning than it would after losing considering equations (3) and (4) and the inequality $t_{m,i} > t_{mp,i}$. Hence, utility losses tend to equalize over time. The bid in equation (5b) represents the drop of the predicted average utility loss for the winning mission. Here, the auctioneer attempts to minimize the total value of utility loss of all applications by selecting the target report with the highest predicted utility loss drop as the winner. This choice ensures the smallest change of the total value of utility loss for all applications.

It should be noted that u_i is predicted, not actual. For the first target report generated for an application, this is because t_{up} is predicted. For the subsequent reports, the computation of u_i assumes that the delay from the current node to the sink will be the same for the current report packet as it was for the previous one. Thus, the achieved average utility loss metrics in congestion may not be exactly the same for all applications (as will be shown in the evaluation section). However, the difference is small in practical cases since the congestion from the tracking report packets usually arises only in a single node between the sources of the reports and the sink.

4.3 Evaluation

We conducted simulations using the ns-2 framework [20] to compare the performance of the auction-based congestion management mechanism with that of the analytical solution. Specifically, we compared the following four approaches: (1) auction-based information utility loss equalization, (2) auction-based total information utility loss minimization, (3) an analytical approach of computing transmission frequencies for equalizing information utility loss, and (4) the division of the communication bandwidth equally among all applications, or the equal bandwidth approach. In this section, we compare the solutions' performance in several scenarios and demonstrate that our auction mechanism achieves its design goals.

Following the previous description of the auction-based mechanism, the simulation used the Spatial TDMA (STDMA) protocol [16], which assigns the same transmission time slots to nodes that cannot interfere with each other's wireless communication. In this way, collision-free communication was provided, albeit at the cost of limited bandwidth for each sensor node. The two most frequently used methods for slot scheduling in STDMA are node assignment and link assignment; the latter configuration assigns actual links, not nodes, to time slots. We implemented a node assignment STDMA version [21], in which the transmission slots were assigned to nodes in a centralized manner.

We used the following simulation configuration for this evaluation. The test bed consisted of 80 sensor nodes distributed uniformly distributed over a 500 x 500 m terrain. Ten time slots were allocated in the STDMA mechanism—each node had 1/10th of the total bandwidth available for transmitting packets in non-colliding slots. A sink node hosted three tracking applications (labeled missions in the plots) that each tracked a different target. The applications' priorities were set to 5, 2 and 1 for applications 1, 2 and 3, respectively (where an application's priority increased along with number representing it). The simulation was configured such that constant traffic was generated as a result of tracking the three moving objects in the sensor network and the three paths of the targets converged near a single point in the network so as to induce congestion at nearby nodes. Initially, all targets were set to travel at the same speed of 10 m/s. All simulations were run for 400 seconds. Sensor nodes reported measurements 5 times per second sending packets of length 625 bytes. At this setting, the packet transmission time was the same as the STDMA slot length. The radio transmission rate for all nodes was set to 250 kb/s, which is the same rate as

that of the MICAz platform motes [22]. Following the previous settings, each node was allocated 25 kb/s of bandwidth for transmission, which was also the same rate of traffic generated by the reporting target information. Thus, network congestion easily occurred when two or more targets' traffic converged at a single node along their paths to the sink.

The plot shown in Fig. 10 (top) presents the actual average information utility loss for the four compared congestion-management approaches. As the plot shows, the average utility loss among different applications is almost equal using either the analytical solution or our auction-based utility loss equalization approach, while they are widely different for the other two approaches. The difference in utility losses is smallest under the auction-based equalization approach, because the analytical approach approximates the frequency computation. Beyond this benefit, our auction-based equalization approach also induces less information utility loss on average across all the tracking applications than the analytical solution. Fig. 10 (bottom) presents the forwarding frequency and average delay for the three applications under the four compared approaches at the congested node. The three discriminating frequency approaches (excluding equal bandwidth approach) all favor application 1 (which has the highest priority) at the cost of application 2 and

especially 3. Still the differences in frequency and delays are smaller for the utility loss minimization approach as compared to being more pronounced in the other two approaches (besides equal bandwidth). These results show the dominating impact of application priority on resource allocation for those approaches.

Overall, while the equal bandwidth approach balances resource usage among applications, it does nothing to balance utility loss among them.

To demonstrate the responsiveness of our proposed approach to dynamic changes in the simulation scenario, the speed of the target tracked by application 2 was changed from 10 m/s to 40 m/s in the middle of the simulation. As shown in Fig. 11, applying the equal bandwidth approach, the actual average utility loss of application 2 increased approximately four times after the speed change. Also gathered from Fig. 11, the utility loss of application 2 increased more than three times after the speed variation, while utility losses for applications 1 and 3 changed only slightly using the utility loss minimization approach. Moreover, Fig. 11 demonstrates that our proposed auction-based utility loss equalization approach and analytical approach kept the average utility loss in congestion nearly equal even after the change of speed.
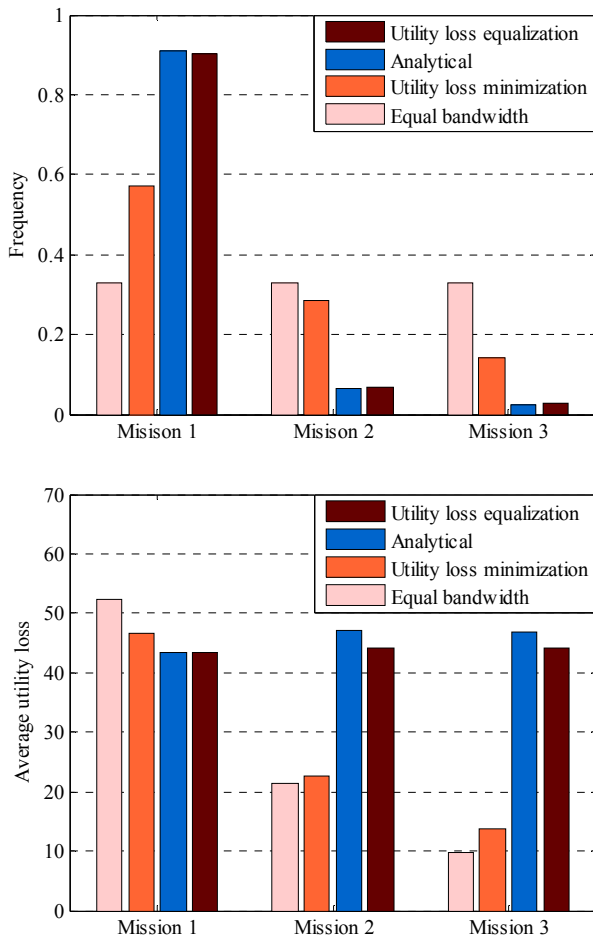
FIGURE 10. The average information utility loss (top) and frequency (f_i) of winning a slot (bottom) under the compared approaches.
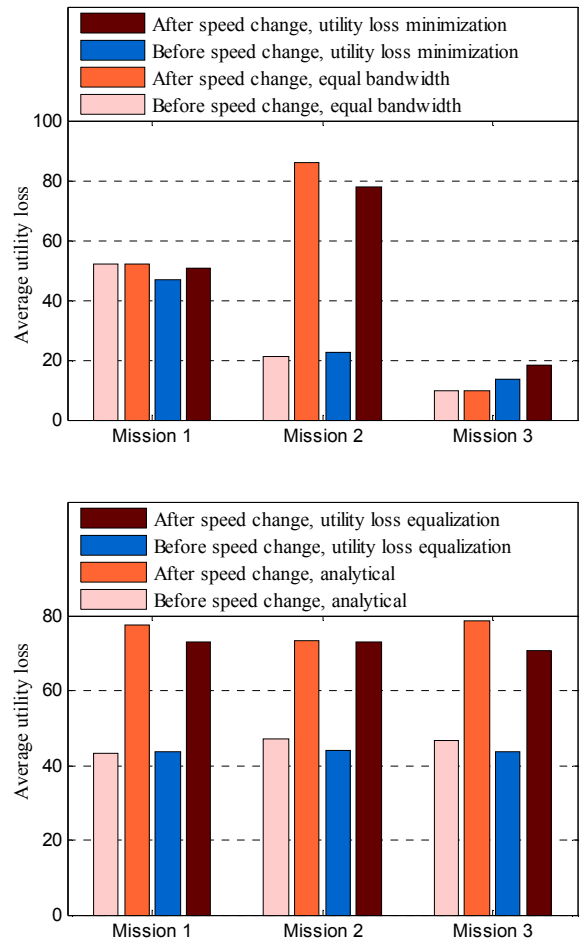
FIGURE 11. Impact of speed change on average utility loss under the compared approaches.

We finalize this section by discussing the computational cost of the auction mechanism. The overhead is rather small, since it consists of just computing the bids at the auctioneer node for each auction as well as disseminating the application priority values whenever they change. Moreover, increasing the number of applications requires simply adding more bidders into the auction process. In addition to good scalability, our approach is also easy to implement and deploy even in dynamically changing environments.

5. Conclusions

In this paper, we addressed the problem of providing congestion-management for WSN-based target tracking systems that are shared among applications that have different priorities and are sensitive to the utility of information of the tracking data that they receive. We used simulations to highlight some of the benefits of our approach in (1) equalizing information utility loss across different target tracking applications and (2) minimizing the total utility loss of all tracking applications in the system.

We have identified several directions for extending this research. One, we plan to explore a novel path selection algorithm, which will use winning bids values at next hop nodes to select the next node on the path with the smallest predicted utility loss value. Two, we plan to relax the assumption of using a homogeneous WSN and further define bidding strategies and utility loss metrics based on the varying capabilities (e.g., detection range, sensing modality) of different sensors. Three, we plan to explore the effects of using different types of auctions.

References

1. Ibbotson, J., Chapman, S. and Szymanski, B. (2007) The Case for an Agile SOA, *Proceedings of the first Annual Conference of the International Technology Alliance (ACITA)*, Adelphi, MD, 25-27 September, ITA, Adelphi, MD.
2. Djuric, P. M., Vemula, M. and Bugallo, M. F. (2004) Signal processing by particle filtering for binary sensor networks, *Proceedings of the 2004 IEEE 11th Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop*, Taos Ski Valley, NM, August 1-4, pp. 263-267, IEEE Computer Society Press, Los Alamitos, CA.
3. Jing, T., Hichem, S. and Cedric, R. (2007) Binary variational filtering for target tracking in sensor networks, *Proceedings of the IEEE/SP 14th Workshop on Statistical Signal Processing*, Madison, WI, August 26-29, pp. 685-689, IEEE Computer Society Press, Piscataway, NJ.
4. Mechitov, K., Sundresh, S., Kwon, Y. and Agha, G. UIUCDCS-R-2003-2379 (2003) *Cooperative tracking with binary-detection sensor networks*. University of Illinois at Urbana-Champaign, IL.
5. Kim, W., Mechitov, K., Choi, J.-Y. and Ham, S. (2005) On target tracking with binary proximity sensors, *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, April 25-27, pp. 301-308, IEEE Computer Society Press, Piscataway, NJ.
6. Shrivastava, N., Mudumbai, R., Madhow, U. and Suri, S. (2006) Target tracking with binary proximity sensors: Fundamental limits, minimal descriptions, and algorithms, *Proceedings of the ACM International Conference on Embedded Networks Sensor Systems (SenSys)* Boulder, CO, October 31 – November 03, pp. 251-264, ACM Press, New York, NY.
7. Wang, Z., Bulut, E. and Szymanski, B. K. (2008) A distributed cooperative target tracking with binary sensor networks, *Proceedings of IEEE International Conference on Communication Workshops*, Beijing, China, May 23, pp. 306-310, IEEE Computer Society Press, Washington, DC.
8. Wang, Z., Bulut, E. and Szymanski, B. K. (2008) Distributed Target Tracking with Imperfect Binary Sensor Networks, *Proceedings of IEEE Global Communications Conference (IEEE GLOBECOM)*, New Orleans, LA, November 30 - December 4, pp. 1-5, IEEE Computer Society Press, Washington, DC.
9. Hu, L. and Evans, D. (2004) Localization for mobile sensor networks, *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, Philadelphia, PA, September 26 – October 1, pp. 45-57, ACM Press, New York, NY.
10. Ee, C. T. and Bajcsy, R. (2004) Congestion control and fairness for many-to-one routing in sensor networks, *Proceedings if the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, November 3-5, pp. 148-161, ACM Press, New York, NY.
11. Hull, B., Jamieson, K. and Balakrishnan, H. (2004) Mitigating congestion in wireless sensor networks," *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, November, 3-5, pp. 134-147, ACM Press, New York, NY.
12. Wan, C.-Y., Eisenman, S. B. and Campbell, A. T. (2003) CODA: Congestion detection and avoidance in sensor networks, *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, November 3-5, pp. 266-279, ACM Press, New York, NY.
13. Karenos, K., Kalogeraki, V. and Krishnamurthy, S. V. (2005) A rate control framework for supporting multiple classes of traffic in sensor networks, *Proceedings of the IEEE International Real-Time Systems Symposium (RTSS)*, Miami, FL, December 6-8, pp. 287-297, IEEE Computer Society Press, Washington, DC.
14. Salles, R.M. and Barriam J,A, (2008) Lexicographic maximin optimisation for fair bandwidth allocation in computer networks, *European Journal of Operational Research,***185**, 2, 778-794.
15. Bisdikian, C. (2007) On Sensor Sampling and Quality of Information: A Starting Point, *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, White Plains, NY, March, pp. 279-284, IEEE Computer Society Press, Washington, DC.
16. Nelson R. and Kleinrock, L. (September 1985) Spatial TDMA: a collision-free multihop channel access protocol. *IEEE Transactions on Communications*, **33**, 9, 934-944.
17. Krishna, V (2002) *Auction Theory*. Academic Press, San Diego, CA.
18. Lee, J.S. and Szymanski, B. K. (2005) A Novel Auction Mechanism for Selling Time-Sensitive E-Services, *Proceedings of the 7th International IEEE Conference on E-Commerce Technology (CEC)*, Munich, Germany, July 19-22, pp. 75-82, IEEE Computer Society Press, Washington, DC.
19. Lee J.–S. and Szymanski, B. K. (2006) Auctions as a dynamic pricing for e-services, In C. Hsu (ed.), *Service Enterprise Integration: An Enterprise Engineering Approach*, Kluwer, New York, NY.
20. The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/.
21. Ramanathan S. and Lloyd, E. L. (April 1993) Scheduling algorithms for multi-hop radio networks. *IEEE/ACM Transactions on Networking*, **1**, 2, 166-177.
22. Crossbow Technology, http://www.xbow.com/.