

Analytical Study of Collaborative Information Coverage for Object Detection in Sensors Networks

Dec 2007

Guanqun Yang, Vinod Shukla, and Daji Qiao
Iowa State University, Ames, IA 50011
{gqyang, vkshukla, daji}@iastate.edu

Abstract

In this report, we provide the detailed description about our proposed on-demand collaborative framework and some proofs, which are omitted in our submission to the IEEE SECON 2008.

I. THE PROPOSED ON-DEMAND COLLABORATIVE FRAMEWORK FOR OBJECT DETECTION

We call our proposed on-demand collaborative framework for object detection as *DeCODE* for short. Formally, our proposed framework is described as follows. Upon sensing a measurement higher than the decision threshold (T_K) where K is the collaboration degree, a sentry node triggers the neighboring inert nodes within its *fusion range* (a disc centered at the sentry node with radius of R_f) to collaboratively sense the environment. A collaboration degree of K means that, in order to report a detection of the object, a sentry node which initiates the detection process needs $K - 1$ positive alarms from sensors in its fusion range. Decision threshold T_K varies with K , N_K^a (number of sentry nodes), and N_K^i (number of inert nodes). Our framework consists of three phases: *initialization*, *bounded flooding*, and *selective bouncing*.

A. Phase 1: Initialization

Upon sensing a measurement higher than the decision threshold, a sentry node initiates the collaborative detection process. If inert nodes function according to the message-based inert node model, the sentry node will immediately start with *Phase 2* after initialization. Otherwise, with the circuit-based inert node model, the sentry node will wait for an appropriate triggering time for the inert nodes within its fusion range to be triggered and then it enters *Phase 2*. The triggering time varies with the size of the fusion range (R_f) as well as the strength of the triggering signal. For example, with one of the radio-triggered circuits which was described in [1] and designed to work with Mica2 motes, if the triggering signal strength is 10 dBm, it takes about 5 ms [1] to trigger the inert nodes within a fusion range of radius $R_f = 30$ feet. Thus, depending on the inert node model, after a certain time, the sentry node enters *Phase 2* to start the *flooding-and-bouncing* protocol to collect alarms from its neighboring inert nodes.

B. Phase 2: Bounded Flooding

The objective of this phase is formation of a tree rooted at the sentry node, which will be used in *Phase 3* to collect positive alarms from sensors within the sentry node's fusion range. This is accomplished via flooding of special *f_msg* messages and supplemental *ack_msg* messages.

The flooding of *f_msg* messages is bounded within the fusion range of the sentry node by a TTL field carried in the *f_msg* message header. The initial TTL value is determined by the radius of the fusion range (R_f) and the average distance between neighboring nodes in the network. Upon reception of the first *f_msg* with a positive TTL value, an inert node (i) attaches itself

Algorithm 1 Bounded Flooding

For each sentry node $sNode_i$ (with node ID i)

Initialization:

1: Determine TTL based on K , N_K^a and N_K^i

Upon sensing a reading higher than decision threshold T_K :

1: Start $timer_c(i)$ with interval τ /* τ is a system parameter */
 2: $sNode_i.ChildrenList \leftarrow \emptyset$
 3: $sNode_i.sumAlarm \leftarrow 1$ /* to be used during Selective Bouncing */
 4: $sNode_i.ReportList \leftarrow \emptyset$ /* to be used during Selective Bouncing */
 5: Broadcast a flooding message: $f_msg(SID = i, NID = i, TTL, bAlarm = 0)$

Upon receiving an acknowledgment message $ack_msg(SID, PID, NID)$:

1: **if** ($ack_msg.SID == i$) AND ($ack_msg.PID == i$) **then**
 2: $sNode_i.ChildrenList \leftarrow sNode_i.ChildrenList \cup ack_msg.NID$
 3: **end if**

Upon $timer_c(i)$ is fired:

1: Clear $sNode_i$ data structure

For each inert node $iNode_j$ (with node ID j)

Initialization:

1: $iNode_j.SentryList \leftarrow \emptyset$
 2: $iNode_j.Need2Sense = 1$

Upon receiving a flooding message $f_msg(SID, NID, TTL, bAlarm)$:

1: $x \leftarrow f_msg.SID$
 2: **if** ($x \notin iNode_j.SentryList$) **then**
 3: Start $timer_b(x)$ with interval δ /* δ is a system parameter */
 4: Start $timer_c(x)$ with interval τ /* τ is a system parameter */
 5: $iNode_j.SentryList \leftarrow iNode_j.SentryList \cup x$
 6: $iNode_j.x.ReportList \leftarrow \emptyset$ /* to be used during Selective Bouncing */
 7: $iNode_j.x.BounceDone \leftarrow 0$ /* to be used during Selective Bouncing */
 8: $iNode_j.x.PID \leftarrow f_msg.NID$
 9: $iNode_j.x.ChildrenList \leftarrow \emptyset$
 10: $iNode_j.x.LocalAlarm \leftarrow 0$
 11: $iNode_j.x.BranchAlarm \leftarrow f_msg.bAlarm$
 12: **if** ($iNode_j.Need2Sense == 1$) **then**
 13: Sense and store its own reading at $iNode_j.Reading$
 14: $iNode_j.Need2Sense = 0$
 15: **end if**
 16: **if** ($iNode_j.Reading \geq T_k$) **then**
 17: $iNode_j.x.LocalAlarm \leftarrow 1$
 18: $iNode_j.x.BranchAlarm \leftarrow 1$
 19: **end if**
 20: $iNode_j.x.sumAlarm \leftarrow iNode_j.x.LocalAlarm$ /* to be used during Selective Bouncing */
 21: Broadcast an acknowledgment message: $ack_msg(SID = x,$
 $PID = iNode_j.x.PID, NID = j$)
 22: **if** ($f_msg.TTL > 1$) **then**
 23: Broadcast a flooding message: $f_msg(SID = x, NID = j,$
 $TTL = f_msg.TTL - 1, bAlarm = iNode_j.x.BranchAlarm$)
 24: **else**
 25: **Execute Algorithm 2: Selective Bouncing**
 26: **end if**
 27: **end if**

Upon receiving an acknowledgment message $ack_msg(SID, PID, NID)$:

1: $x \leftarrow ack_msg.SID$
 2: **if** ($x \in iNode_j.SentryList$) AND ($ack_msg.PID == j$) **then**
 3: $iNode_j.x.ChildrenList \leftarrow iNode_j.x.ChildrenList \cup ack_msg.NID$
 4: **end if**

Upon $timer_b(x)$ is fired:

1: **Execute Algorithm 2: Selective Bouncing**

Upon $timer_c(x)$ is fired:

1: $iNode_j.SentryList \leftarrow iNode_j.SentryList - x$
 2: $iNode_j.Need2Sense = 1$
 3: Clear $iNode_j.x$ data structure

to the tree by replying with an ack_msg , and (ii) refreshes its sensed reading, if necessary, and records it in a local variable $iNode_j.Reading$.

After Algorithm 1 has been executed for sentry node x , almost all the inert nodes within its fusion range are attached to the tree. Each on-tree node ($iNode_j$) maintains the IDs of its parent

node ($iNode_j.x.PID$) and children nodes ($iNode_j.x.ChildrenList$), as well as a Boolean variable ($iNode_j.x.BranchAlarm$) which indicates whether any of the node between itself and the sentry node has sensed a measurement higher than the decision threshold.

Fig. 1 gives an example of the proposed framework. Fig. 1(a) shows a sentry node and inert nodes within its fusion range before flooding-and-bouncing. The formed tree after execution of Algorithm 1 is shown in Fig. 1(b), where black/white dots represent inert nodes with sensed readings higher/lower than the decision threshold, and cross dots represent inert nodes whose own readings are lower than the decision threshold but lie along the branches on which at least one node has sensed a reading higher than the decision threshold.

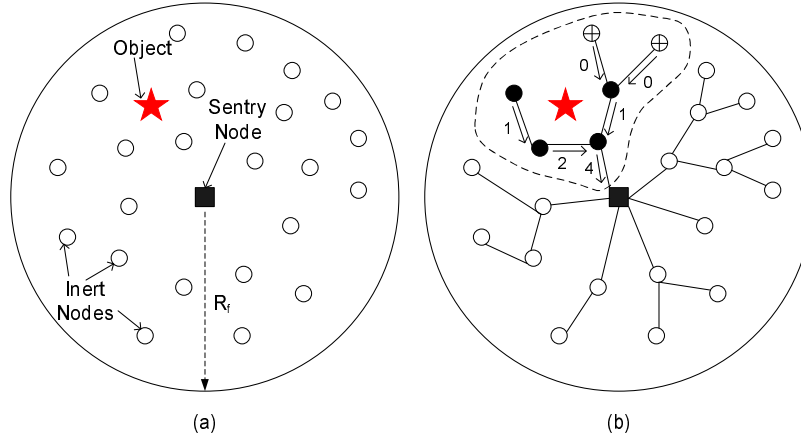


Fig. 1. The proposed DeCODE framework. The object is shown as the star; the sentry node is shown as the black square; circles represent inert nodes. (a) Network topology before flooding-and-bouncing. (b) During *Phase 3: Selective Bouncing*, bouncing messages are initiated by leaf nodes and routed towards the sentry node along the sub-tree inside the dash-curve-bounded region; numbers along the edges are the $iNode_j.x.sumAlarm$ values reported by the corresponding inert nodes.

C. Phase 3: Selective Bouncing

The objective of this phase is to collect positive alarms from relevant sensor nodes and propagate them to the sentry node. This is accomplished via special b_msg messages.

As shown in Algorithm 2, only leaf nodes who either have sensed a measurement higher than the decision threshold or belong to a branch on which at least one node has sensed a measurement higher than the decision threshold can initiate the bouncing process; for example, the bouncing process in Fig. 1(b) is initiated by the leaf nodes inside the dash-curve-bounded region. An inert node relays the bouncing message after hearing from all of its children, and indicates in the message the total number of positive alarms raised by nodes belonging to the subtree rooted at itself: $b_msg.nAlarm = iNode_j.x.sumAlarm$. Such bouncing process is expedited when an inert node has collected adequate (i.e., $\geq K - 1$) positive alarms for the sentry node to report a detection. Each inert node also maintains the latest $sumAlarm$ values reported by its children. This enables the node to update its $sumAlarm$ values in case one of its children nodes reports multiple times. This situation may occur because, if the tree is unbalanced, it is possible (though not likely) that the leaf nodes along the shorter branches have already started bouncing while the formation of the longer branches has not yet completed.

Not shown in Algorithms 1 and 2 are how a sentry node responds to the f_msg messages from other sentry nodes and how it participates in selective bouncing. In such situations, the sentry node acts exactly like an inert node except that it uses its most recent sensed reading to participate in the decision process, instead of performing an additional sensing upon reception of the f_msg message.

Algorithm 2 Selective Bouncing

For each inert node $iNode_j$ (with node ID j)

Initialization:

```

1: for (Each sentry node  $x \in iNode_j.SentryList$ ) do
2:   if ( $iNode_j.x.ChildrenList == \emptyset$ ) AND ( $iNode_j.x.BranchAlarm == 1$ ) then
3:     Broadcast a bouncing message:  $b\_msg\langle SID = x, PID = iNode_j.x.PID,$ 
 $NID = j, nAlarm = iNode_j.x.sumAlarm\rangle$ 
4:     if ( $iNode_j.x.sumAlarm \geq K$ ) then
5:        $iNode_j.x.BounceDone \leftarrow 1$ 
6:     end if
7:   end if
8: end for

```

Upon receiving a bouncing message $b_msg\langle SID, PID, NID, nAlarm\rangle$:

```

1:  $x \leftarrow b\_msg.SID$ 
2:  $y \leftarrow b\_msg.NID$ 
3: if ( $x \in iNode_j.SentryList$ ) AND ( $iNode_j.x.BounceDone == 0$ )
   AND ( $b\_msg.PID == j$ ) then
4:   if ( $y \notin iNode_j.x.ReportList$ ) then
5:      $iNode_j.x.ReportList \leftarrow iNode_j.x.ReportList \cup y$ 
6:      $iNode_j.x.sumAlarm \leftarrow iNode_j.x.sumAlarm + b\_msg.nAlarm$ 
7:      $iNode_j.x.nAlarm_y \leftarrow b\_msg.nAlarm$ 
8:   else
9:     if ( $iNode_j.x.nAlarm_y < b\_msg.nAlarm$ ) then
10:       $iNode_j.x.sumAlarm \leftarrow iNode_j.x.sumAlarm - iNode_j.x.nAlarm_y$ 
 $+b\_msg.nAlarm$ 
11:       $iNode_j.x.nAlarm_y \leftarrow b\_msg.nAlarm$ 
12:    end if
13:   end if
14:   if ( $iNode_j.x.ReportList == iNode_j.x.ChildrenList$ )
   OR ( $iNode_j.x.sumAlarm \geq K$ ) then
15:     Broadcast a bouncing message:  $b\_msg\langle SID = x,$ 
 $PID = iNode_j.x.PID, NID = j, nAlarm = iNode_j.x.sumAlarm\rangle$ 
16:     if ( $iNode_j.x.sumAlarm \geq K$ ) then
17:        $iNode_j.x.BounceDone \leftarrow 1$ 
18:     end if
19:   end if
20: end if

```

For each sentry node $sNode_i$ (with node ID i)

Upon receiving a bouncing message $b_msg\langle SID, PID, NID, nAlarm\rangle$:

```

1:  $y \leftarrow b\_msg.NID$ 
2: if ( $b\_msg.SID == i$ ) AND ( $b\_msg.PID == i$ ) then
3:   if ( $y \notin sNode_i.ReportList$ ) then
4:      $sNode_i.ReportList \leftarrow sNode_i.ReportList \cup y$ 
5:      $sNode_i.sumAlarm \leftarrow sNode_i.sumAlarm + b\_msg.nAlarm$ 
6:      $sNode_i.nAlarm_y \leftarrow b\_msg.nAlarm$ 
7:   else
8:     if ( $sNode_i.nAlarm_y < b\_msg.nAlarm$ ) then
9:        $sNode_i.sumAlarm \leftarrow sNode_i.sumAlarm - sNode_i.nAlarm_y + b\_msg.nAlarm$ 
10:       $sNode_i.nAlarm_y \leftarrow b\_msg.nAlarm$ 
11:    end if
12:   end if
13:   if ( $sNode_i.sumAlarm \geq K$ ) then
14:     Report a detection
15:     return true
16:   end if
17: end if

```

II. DISCUSSION ABOUT USING MULTIPLE DECISION THRESHOLDS IN DECODE

In this section, we provide the omitted proof about using multiple detection thresholds in our proposed DeCODE. Some notations may refer to our formal submission to the IEEE SECON 2008. We assume that the noise value follows a standard normal distribution.

Before we proceed to that problem, we introduce the concepts of detection zone (D.Z.) and fusion range R_f as follows. We define detection zone (D.Z.) to be a disc centered at the object and with a radius R_d such that, at distance R_d from the object, the probability of a sensor's measurement exceeding the decision threshold is less than a very small number (we use 1% in this paper). Fig. 2(a) illustrates the variation of probability of a sensor's measurement being higher than the decision threshold with respect to the distance between sensor and object. It can

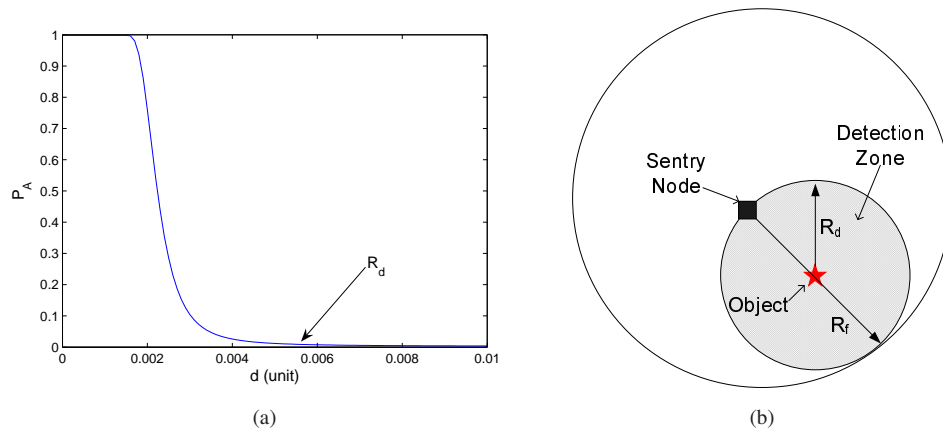


Fig. 2. (a) Illustration of the relation between the sensor-to-object distance and the probability that sensor's measurement is higher than its decision threshold; and (b) Illustration for detection zone and fusion range.

be seen that the probability is almost one at short distances and then sharply falls. Thus, the probability for a sensor's measurement being above the decision threshold is quite high within the detection zone and is almost negligible outside the detection zone.

We define fusion range to be a disc centered at the sentry node and with radius $R_f = 2R_d$. Such definition of fusion range guarantees that, whenever a sentry node is within detection zone of the object and senses a measurement above the decision threshold T_K (which itself is a high-probability event), all inert nodes within the detection zone will be triggered, as illustrated in Fig. 2(b).

In our proposed DeCODE, there is only one decision threshold for all sentry nodes and inert nodes. Naturally, there could be multiple decision thresholds in an 'OR' collaborative detection scheme. However, we will show that such a compounded detection scheme can not help improve the \bar{P}_D for a fixed P_{FD} .

The uniqueness of DeCODE is to utilize inert nodes around sentry nodes to help sense the environment in an on-demand fashion. In the numerical study, we observe that the density of inert nodes is usually much higher than that of sentry nodes as shown in Fig. 3. Thus, after the

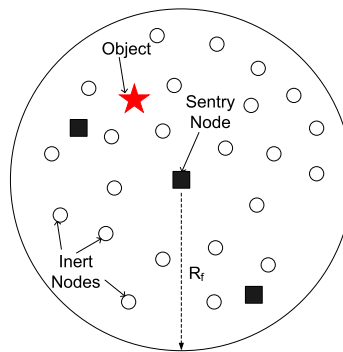


Fig. 3. Illustration of sentry nodes and inert nodes in the detection zone. The object is shown as the red star.

inert nodes in the D.Z. get triggered by the sentry node, they can generate enough alarms to raise a positive detection with a very high probability (almost equal to 1). From the above observation,

we may get an approximate expression of \bar{P}_D as below:

$$\begin{aligned} \bar{P}_D &\approx P(\text{at least 1 sentry node within D.Z. has a measurement higher than } T_K) \\ &= \sum_{n=1}^{N_K^a} P(n \text{ sentry nodes in D.Z.}) \cdot P(\text{among those } n \text{ sentry nodes, at least 1 sentry node raise an alarm}) \\ &= \sum_{n=1}^{N_K^a} \frac{(\lambda_a \cdot \|D.Z.\|)^n \cdot e^{(-\lambda_a \cdot \|D.Z.\|)}}{n!} \cdot (1 - (1 - P_{sn}(K))^n), \end{aligned} \quad (1)$$

where $\lambda_a = N_K^a$ is the density of sentry nodes and

$$\begin{aligned} P_{sn}(K) &= P(\text{a sensor within D.Z. has a measurement higher than } T_K) \\ &= \int_0^{\frac{R_f}{2}} \frac{1}{\left(\frac{R_f}{2}\right)^2 \pi} \cdot 2\pi r \cdot \text{erfc}\left(T_K - \frac{\Omega d_0^2}{r^2}\right) dr. \end{aligned} \quad (2)$$

Next, we will compare the decision threshold of different detection schemes (using a single threshold or multiple thresholds), and find the reason why using multiple thresholds does not help improve the \bar{P}_D . Suppose every sensor has a set of decision thresholds, say, $T'_K, T'_{K-1}, \dots, T'_m$, among which the lowest threshold is T'_K as shown in Fig. 4 (b). Note that m could be equal

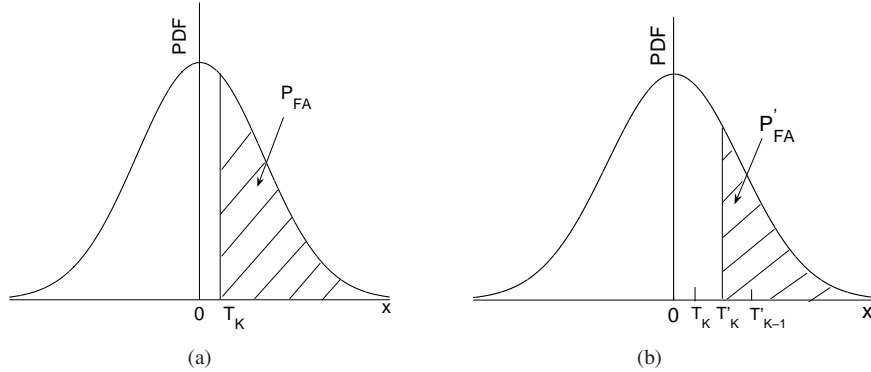


Fig. 4. (a) Decision threshold T_K in DeCODE; and (b) Decision thresholds T'_K and T'_{K-1} in the compounded detection scheme.

to or larger than 1. If a sentry node gets a sensed reading higher than T'_n but lower than T'_{n-1} ($n - 1 \geq m$), then at least $n - 1$ nearby sensors should get sensed readings higher than T'_n in order to raise a positive detection. Since such compounded detection scheme is based on an ‘OR’ rule, i.e., a positive detection can be raised if enough positive alarms have been collected with respect to any decision threshold, the P_{FD} of the above compounded detection scheme (using thresholds $T'_K, T'_{K-1}, \dots, T'_m$) is higher than that of the detection scheme using T'_K only. Also, because a lower decision threshold usually yields a higher P_{FD} , we can expect that T_K in the detection scheme with one single threshold is even lower than T'_K in the compounded detection scheme for a fixed P_{FD} . Recall that \bar{P}_D is approximately determined by the probability that at least one sentry node in D.Z. raises an alarm (see Eq. (1)), and a sentry node with a lower decision threshold is always more likely to raise a genuine alarm than that with a higher decision threshold. Therefore, we find that, given a fixed P_{FD} , \bar{P}_D of the detection scheme using one single threshold T_K ($T_K < T'_K$) is higher than that of the compounded detection scheme.

REFERENCES

- [1] L. Gu and J. Stankovic, “Radio-Triggered Wake-Up Capability for Sensor Networks,” in *Proc. IEEE RTAS*, Apr 2004.