Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2014) 22: 1437 – 1452

doi:10.3906/elk-1203-75

Fault tolerant broadcasting analysis in wireless monitoring networks

Akbar Ghaffarpour RAHBAR*
Electrical Engineering Technologies Research Center, Computer Networks Research Lab,
Sahand University of Technology, Tabriz, Iran

Received: 17.03.2012 • **Accepted:** 21.06.2012 • **Published Online:** 07.11.2014 • **Printed:** 28.11.2014

Abstract: Wireless monitoring networks can be used for security applications such as the monitoring of narrow passages and operational fields. These networks can be designed based on sensor networks. In sensor networks, each node can hear a message and broadcast the message to its neighbor nodes. Nevertheless, nodes may fail, so that faulty nodes cannot hear or cannot transmit any message, where the locations of the faulty nodes are unknown and their failures are permanent. In this paper, the nodes are situated on a line or a square grid-based topology in a plane for security/monitoring applications. For each topology, 2 nonadaptive and adaptive broadcasting scheduling algorithms are proposed and analyzed. In addition, the scheduling algorithms take the energy consumption of the sensor nodes into account in order to prolong the network's lifetime. The analysis results show that adaptive algorithms need less time than nonadaptive algorithms to inform the whole network domain.

Key words: Wireless networks, broadcasting, fault tolerance, adaptive scheduling algorithms, nonadaptive scheduling algorithms

1. Introduction

Wireless sensor networks (WSNs) have attracted much attention and interest from both industry and academic researchers due to their low-cost infrastructure and flexibility. A WSN includes many low-cost and low-power sensors that can collect observations in an environment and then send the relevant information to a central sink node, which is responsible for processing the information and making appropriate decisions. Because of their agility, low implementation cost, and robustness, WSNs have found wide applications [1–5], such as military applications, environmental monitoring applications, and target tracking applications.

A straight line vector of sensors (with equidistance between the sensor nodes) can be used for security applications such as the monitoring of a narrow passage (e.g., an indoor corridor, a tunnel, a bridge) [6]. Sensors can also be deployed on a square grid-based topology (SGT), where a sensor field is divided into square grids and the sensor nodes are deployed on grid points [7–10]. Finally, the sensor nodes could be deployed on a polygonal topology [11].

A WSN is a collection of transmitter–receiver devices located in a geographical region, in which a node receives a message and transmits (broadcasts) it. Each node has a range of transmission and reception. When a given node X is located in the range of 2 nodes V and W, and both of them start transmitting, node X will receive a collided message. However, if one of them transmits a message, node X can successfully receive it.

In a WSN, a sensor node may fail due to a lack of power, physical damage, or environmental interferences. However, the failure of a sensor node should not exert a negative impact on the overall operations of the WSN

*Correspondence: ghaffarpour@sut.ac.ir

[12]. Since energy consumption is an important problem in sensor nodes [13–18], its usage must be controlled in order to prolong the WSN's lifetime.

Broadcasting could be the fundamental operation in WSN communications. In this network, a source node may have a message for all of the other nodes in the network. To deliver the message to remote nodes not located within the range of the source node, intermediate nodes have the responsibility of receiving the message and transmitting it until all of the nodes are informed. The total time required to inform all of the nodes in the network is an important performance parameter of broadcasting. An important feature for broadcasting algorithms is their capacity to inform all of the nodes in the presence of unknown-located faulty nodes. In the presence of faulty nodes, the topology of the network is uncontrolled. On the other hand, broadcasting algorithms designed for fault-free networks cannot be used for these networks; i.e. fault-tolerant algorithms should be designed. The works in [19,20] have analyzed fault tolerance broadcasting in heterogeneous WSNs. There are also some works, such as [21,22], that provide fault tolerance broadcasting in homogeneous WSNs. Broadcasting algorithms have been proposed and analyzed in radio networks in [23].

The contributions of this paper are as follows: 1) the algorithms proposed for radio networks in [23] are adapted and modified to be useful for WSNs; 2) in [23], the energy of the nodes has not been considered at all. In this paper, the energy consumption of the sensor nodes is considered in the newly proposed algorithms; and 3) since many details are missing in the analysis of the message broadcasting time (MBT) in the algorithms provided in [23], this paper also aims to provide a simpler analysis with more details compared with [23]. The worst case analysis is considered in computing the MBT.

The organization of this paper is as follows: The network model, assumptions, and definitions are stated in Section 2. In Section 3, the broadcasting scheduling algorithms are discussed when deploying sensor nodes on line. Broadcasting scheduling algorithms for sensor nodes distributed on a SGT topology are described in Section 4. Section 5 provides a brief conclusion.

2. Network model, assumptions, and definitions

In this paper, 2 different topologies for locating sensor nodes are considered. In the first topology, sensor nodes are located on a line topology, where the distance between every pair of adjacent nodes is equal. In the second topology, sensor nodes are deployed on the grid points of a SGT network. As stated in Section 1, these topologies are suitable for security applications.

To model the system, several assumptions and definitions are made:

- All of the nodes have previously assigned distinct identifiers. In other words, the nodes are only determined with their identifiers.
- Each node adjusts its range of transmission to radius R.
- In terms of available energy, the sensor nodes are heterogeneous.
- A fault may happen for any reason. In addition, when a node cannot cover transmission range R due to its low battery, it is called a faulty node as well.
- Maximum number of faulty nodes in the network is denoted by F. This number increases by 1 whenever the energy of a sensor node finishes, i.e. it encounters a permanent fault.
- Faults are assumed to be permanent and their locations are unknown. The worst case distribution of faulty nodes is considered in our analysis. The source node is assumed to be fault-free.

- A node $v$located within the range of more than one simultaneous transmission may receive a collided message (noise), where collision detection is available.
- A broadcast message can only arrive at fault-free connected components of the source node, which is called the *domain*. In other words, a domain is the connected components of the fault-free nodes of graph G, in which the adjacent nodes are in each other's range. For example, in Figure 1, the black nodes are faulty nodes, so that node number 2 cannot deliver the message received from the leftmost nodes to the rightmost nodes. Thus, the domain is restricted to the leftmost graph.
- Parameter D is the network diameter, i.e. maximum of all of the shortest paths between the source node and the domain nodes. In Figure 1, we have $D = 4$.
- The system is time slotted and each node is scheduled to transmit a message at a specific time slot. Assume a message can be completely transmitted within a time slot.
- The MBT is defined as the maximum number of time slots elapsed between the first transmission of a message by a source node until its reception by all of the nodes in the domain.

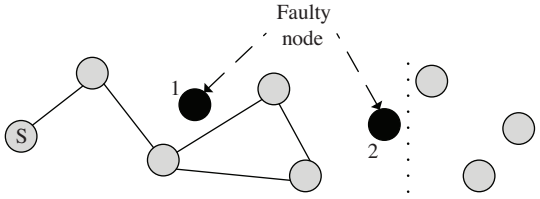

Figure 1. Node 2 cannot receive and transmit, and therefore the rightmost nodes are out of the domain.

Two types of broadcasting algorithms are defined as:

1. **Nonadaptive algorithms** are defined with the following features:
   - All transmissions are scheduled in advance. Each node has a table that specifies the time slots of the transmission. It is possible that 2 nodes that are not located within range of each other transmit simultaneously. Obviously, the scheduling of simultaneous transmissions close to each other should be avoided because this will result in noise/interference for the receiving nodes. Scheduling in sparse areas causes a communication delay.
   - If a node is scheduled to send a message in a time slot, but it has not received the message, it will transmit a default message.
   - Nodes are scheduled to transmit in a distributed manner based on their labels, where the label for a node is defined based on its location in a segment and the segment number.
   - The periodic scheduling of a node only depends on its label, not the length of line or size of the SGT.
2. **Adaptive algorithms** are defined with the following features:
   - Nodes are scheduled for future transmission based on their communication history.

- Nodes learn the faulty status of other nodes from the obtained messages and noises.
- The behavior of a node only depends on its label.
- Nodes are scheduled to transmit in a semidistributed manner, i.e. a group of sensor nodes cooperate with each other to provide the scheduling. The central fault-free node in the group should run the scheduling algorithm. Notice that the network is divided into a number of groups and each group is assigned a number. First, the groups with odd numbers perform the scheduling in parallel, and then the groups with even numbers carry out the scheduling in parallel.

3. Distribution of the sensor nodes on a line

Here, sensor nodes of a WSN are distributed on a line and at integer points of 0, 1,..., $n-1$, where n is the number of total nodes. For $1 < R < n-1$, all of the nodes in set $\{v\pm1, v\pm2,\ldots, \text{v} \pm R\}$ are in the range of node v. All of the nodes in set $\{(i-1)R+ 1,\ldots,i \times R\}$ belong to the ith segment. Each segment includes at most R nodes. Since the worst case analysis is considered in computing the MBT, node 0 is considered to be the source node located at the leftmost part of the line.

Parameter m is defined to be the largest-numbered node in the domain (may be different from $n-1$ due to the presence of faulty nodes in the network). It is clear that parameters D and m depend on the configuration of the faulty nodes in the network. For example, in Figure 2, we have S = source node (numbered 0), $n = 21$ nodes, $R = 5$, $F = 6$ faulty nodes (shown with black nodes), and 4 segments. For node $v = 9$, the transmission range has been drawn covering nodes from $v - R = 4$ to $v + R = 14$. Figure 3 depicts the domain of this network and the nodes that are located within each other's range. The black nodes are faulty nodes that are out of the domain. In this figure, only the arcs between the segments have been drawn. For simplicity, the arcs within a segment have not been displayed. According to the figure, the nodes with a large number in each segment can cover more nodes in the neighbor segments. In this diagram, the largest-numbered node in the domain is node $m = 18$ and $D = \lceil m/R \rceil = 4$.

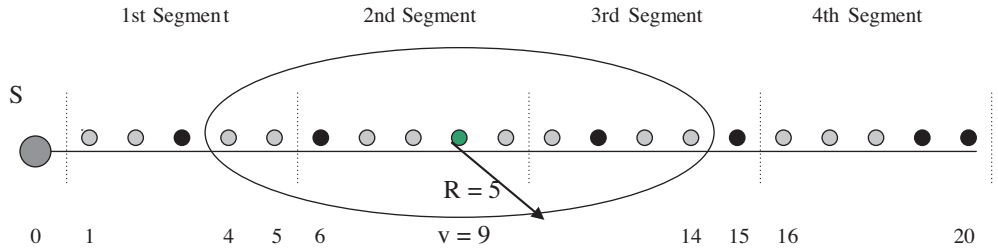

Figure 2. An example of the segment and transmission range for nodes on a line.

3.1. Line nonadaptive (LINE_NA) algorithm

Here, the nonadaptive broadcasting scheduling algorithm in Figure 4 is used for a line topology (*LINE-NA*). The scheduling formula in the LINE_NA algorithm computes the transmission slot time for each node a_{ij}. Here, parameter c is the period number for the scheduling. The choice of tradeoff coefficient β is important since it influences the transmission delay, signal interference probability, and energy consumption of the sensor nodes. Note that the case of $\beta = 1$ is equivalent to the method described in [23]. For the choice of β, the following considerations should be taken into account:

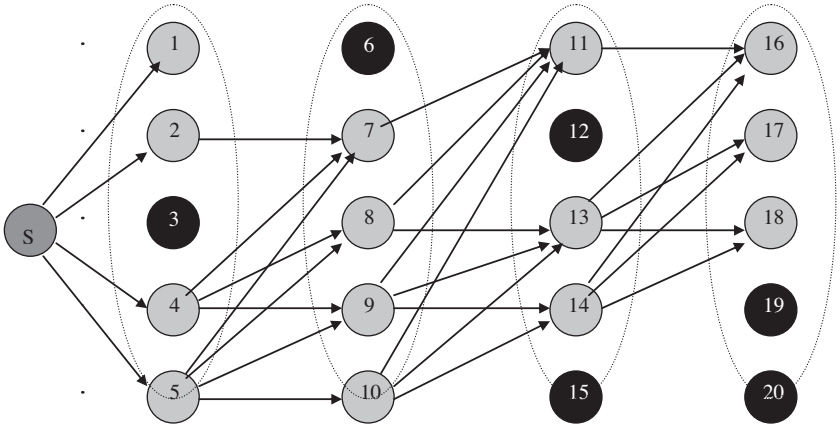

Figure 3. Domain of Figure 2 with intersegment connections.

- *For each segment, nodes are numbered in decreasing order. For the i*th *segment, nodes are numbered as* $a_{iR}, \ldots, a_{i3}, a_{i2}, a_{i1}$.
- *Source transmits at time slot* 1.
- *Fault-free node* a_{ij} *transmits at time slot*

$$i + \beta \times j - \beta + 1 + \beta \times c \times (R + 1), \text{ where } c = 0, 1, 2, \ldots.$$

Figure 4. Nonadaptive scheduling algorithm for the linear distribution of the sensor nodes.

- By choosing β as a large number, the transmission delay increases, while the signal interference probability decreases. In addition, the energy consumption of a sensor node decreases since the node transmits at long intervals (due to the $\beta \times c \times (R+ 1)$ term in the scheduling). For example, at $\beta = 3$ and $R = 5$, node a_{43} broadcasts messages with an 18-time slot difference at time slots 11 (when $c = 0$), 29 (when $c = 1$), 47 (when $c = 2$), etc.

- By choosing β as a small number, the delay decreases, while the signal interference probability increases. Furthermore, the energy consumption for a node goes up. For example, at $\beta = 1$ and $R = 5$, node a_{43} broadcasts messages with a 6-time slot difference at time slots 7 (when $c = 0$), 13 (when $c = 1$), 19 (when $c = 2$), 25 (when $c = 3$), etc.

- As can be observed, node a_{43} broadcasts 4 times until time slot 29 at $\beta = 1$, but only twice until time slot 29 at $\beta = 3$. Therefore, $\beta = 2$ could be a reasonable value as a tradeoff between the transmission delay, energy consumption, and signal interference. Figure 5 illustrates an example for the scheduling time in a network under 3 periods ($c = 0, 1, 2$) at $\beta = 2$ and $R = 5$. According to Figure 5, nodes a_{44} and a_{25} transmit simultaneously, but their transmissions never cause a collision in the network due to their distance of $2 \times R$ from each other.

To compute the complexity of LINE_NA, we should consider the following 2 cases:

Case 1: If node a_{iR} is transmitting and node $a_{(i+1)R}$ is faulty, then segment i must send the message at the next time starting from node a_{i1} and from the next period of the scheduling time. Now, it is shown that retransmitting a message increases the MBT by 4 time units. Using the relation in Figure 4 at $\beta = 2$, the transmission times for nodes a_{iR} and a_{i1} in periods c and $c + 1$ are:

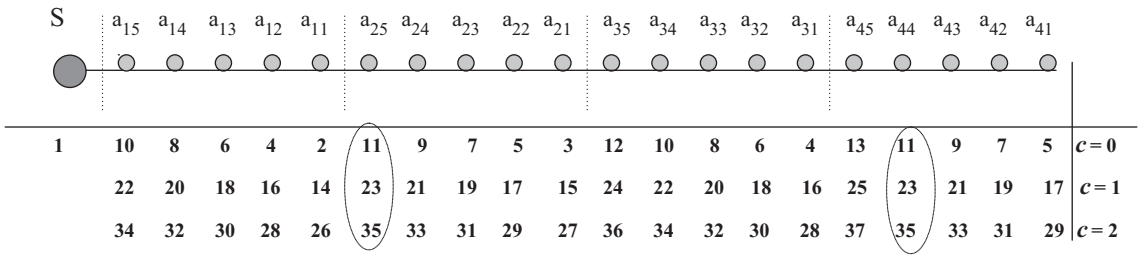

Figure 5. Scheduling algorithm *LINE_NA* at $\beta = 2$ and $R = 5$: the 3 bottom rows show the transmission times under different scheduling periods c.

$T_{a_{iR}} = i + 2 \times R - 1 + 2 \times c(R+1) =$ Time slot assigned to node a_{iR} in period c.

$T_{a_{i1}} = i + 2 \times 1 - 1 + 2(c+1)(R+1) =$ Time slot assigned to node a_{i1} in period $c+1$.

Therefore, we have $T_{a_{i1}} - T_{a_{iR}} = 4$.

Case 2: To achieve the upper bound on the broadcasting time, the following fault configuration should be considered:

- Faults are located on the right side of the first segment starting from a_{11}.
- Consider that a_{ij} is the current transmitting node, where node(s) $a_{(i+1)k}$ (where $k \geq j$) are sequentially faulty. This will increase the broadcasting time by $2 \times (k - j + 1)$ time units.

For example, in Figure 6, one of the possible worst case fault configurations is shown at $R = 5, n = 51$, and $F = 14$, where a node on the top of a column denotes the left side of a segment and the node on the bottom of that column denotes the right side of that segment. The dashed segments should send more than once due to Case 1. Here, node 46 is the last segment informed by the previous segment, where nodes 48, 49, and 50 should be informed by node 46.

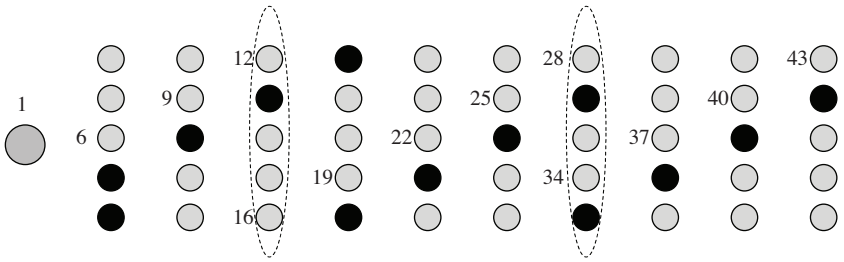

Figure 6. A worst case fault configuration at $n = 51$, $R = 5$, $F = 14$. Numbers beside the nodes present the transmission times.

Thus, the different delay components of the MBT to inform the whole domain are:

- Each fault in a segment causes a 2-time unit delay in the broadcasting. Therefore, in the worst case, $2 \times F$ time units are wasted in the segments until the node that is not faulty starts its transmission.
- There are some cases that increase the broadcasting time by 4 time units. Following the mentioned fault distribution, there are an average of F/N_s (where N_s is the number of segments in the network) faults in each segment. From almost each $z = R/(F/N_s)$ segments, we need 4 additional time units. Since the total number of such cases is $[N_s / z] = [F/R]$, the total delay is equal to $4 \times [F/R]$. Note $[x]$ returns ceil of x.
- We also need at most D time units to pass through all of the segments.

Therefore, at $\beta = 2$, it takes at most MBT $= D+2 \times F+4 \times [F/R] \in O(D+F)$ time slots to inform all of the nodes in the domain.

According to [23], the MBT at $\beta = 1$ is at most $D+F+[F/R]$. When there is no fault, the MBT values for both $\beta = 1$ and $\beta = 2$ are the same. However, when $F > 0$, it seems that the MBT at $\beta = 2$ is always very high compared to the MBT at $\beta = 1$. Nevertheless, notice that the energy consumption of the nodes at $\beta = 1$ is high, and therefore the number of faulty nodes F will increase faster at $\beta = 1$ compared to $\beta = 2$. In this case, the MBT will increase faster at $\beta = 1$ compared to $\beta = 2$.

Example For an energy consumption illustration, let E_{ij} denote the number of messages that node a_{ij} can broadcast before failing. This parameter is directly relevant to the energy of node a_{ij}, i.e. higher E_{ij} means more energy. In other words, E_{ij} is the capacity of the message broadcasting for node a_{ij}. Since only one message is broadcast within each period, the last broadcasting of node a_{ij} occurs at time slot $t_{ij} = i + \beta \times j - \beta + 1 + \beta \times E_{ij} \times (R+1)$. In other words, at time slot $T > t_{ij}$, node a_{ij} is a faulty node since its energy has been depleted. Consider $R = 5$ and $n = 21$. According to this rule, at different time slots, Figure 7a shows the number of faulty nodes F under $\beta = 1$ and $\beta = 2$, and Figure 7b depicts the MBT under $\beta = 1$, and $\beta = 2$. Here, the capacity E_{ij} for sensor node a_{ij} is uniformly distributed between 1000 and 2000 messages. As can be observed, under $\beta = 1$, the number of faulty nodes reaches $F = 8$ at time slot 9000, and 21 (i.e. all of the nodes become faulty) at time slot 12,000. In other words, all of the nodes fail until time slot 12,000 under $\beta = 1$. Hence, there is no MBT shown after time 12,000 in Figure 7b. On the other hand, under $\beta = 2$, we have $F = 8$ at time slot 17,500. In addition, the MBT is always smaller under $\beta = 2$ compared with $\beta = 1$. These diagrams show the superiority of $\beta = 2$ over $\beta = 1$ in the energy consumption of the nodes.

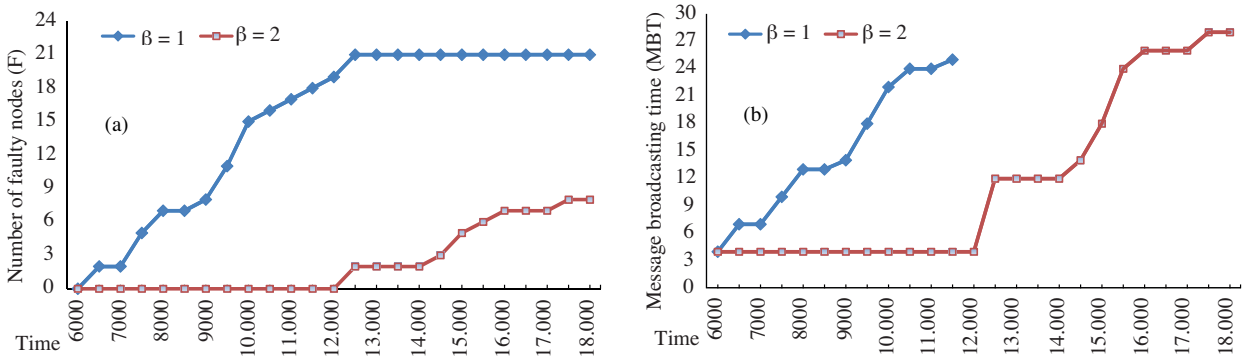

Figure 7. a) Number of faulty nodes F at $\beta = 1$ and $\beta = 2$, and b) the MBT at $\beta = 1$ and $\beta = 2$.

3.2. Line adaptive (LINE_A) algorithm

Most of the time spent in nonadaptive broadcasting is due to faulty nodes. In an adaptive algorithm, to speed up the broadcasting time, faulty nodes are not given the chance for transmission. Thus, preprocessing should be first performed in order to detect the fault-free nodes in each segment in a binary manner, as in [23]. Fault-free nodes could be detected using other techniques such as those in [24–26]. Here, 3 procedures are used to detect the fault-free nodes. The *BinaryElectSmallest* (see Figure 8) chooses the smallest-numbered fault-free node in segment N, and the *BinaryElectLargest* (see Figure 8) selects the largest-numbered fault-free node in segment N. In addition, the *BinaryElectMiddle* (see Figure 9) is used to select 2 fault-free nodes from the middle of

segment N. Notice the nodes on the left side of a segment are large-numbered nodes, and the nodes on the right side of a segment are small-numbered nodes.

```
Procedure BinaryElectLargest (N)        ; to choose the largest-numbered fault-free node in segment N
 U = min(R, F + 1)                      ; if F < R, then U = F + 1; otherwise, U = R
 ACT = Set of U largest-numbered nodes in N   ; set of nodes to be investigated as active
 I = 1
 While (I ≤ log2(U))                    ; until the ACT set includes one item
     S = Smaller half of ACT
     L = Larger half of ACT
     Invoke all nodes in L to start transmission   ; fault-free nodes transmit simultaneously
     If message or noise is heard, then ACT = L    ; there is at least one fault-free node in L
                                else ACT = S       ; L is removed when all nodes in L are faulty
    I = I + 1
 End while
 Remaining node in ACT transmits its identity      ; to the fault-free nodes in the segment
 Return identity                        ; to inform the segment active node identity
End


Procedure BinaryElectSmallest (N)       ; to choose the smallest-numbered fault-free node in segment N
 U = min(R, F + 1)                      ; if F < R, then U = F + 1; otherwise, U = R
 ACT = Set of U smallest-numbered nodes in N   ; set of nodes to be investigated as active
 I = 1
 While (I ≤ log2(U))                    ; until the ACT set includes one item
     S = Smaller half of ACT
     L = Larger half of ACT
     Invoke all nodes in S to start transmission   ; fault-free nodes transmit simultaneously
     If message or noise is heard, then ACT = S    ; there is at least one fault-free node in S
                                else ACT = L       ; S is removed when all nodes in S are faulty
    I = I + 1
 End while
 Remaining node in ACT transmits its identity      ; to the fault-free nodes in the segment
 Return identity                        ; to inform the segment active node identity
End
```

Figure 8. The BinaryElectLargest and BinaryElectSmallest algorithms.

Two sorts of adaptive algorithms can be used here:

1. **LINE_A1:** Figure 10 shows the LINE_A1 algorithm adapted from [23]. At the preprocessing step (i.e. Step I) in each segment, this algorithm uses the aforementioned *BinaryElectLargest*, *BinaryElectSmallest*, and *BinaryElectMiddle* procedures in a parallel manner to specify 2 fault-free nodes as the head of the segment for the intra- and intersegment transmissions. To avoid a collision, the segments should not be adjacent. Whenever the number of faults F increases due to the failure of a node, Step I should be reexecuted. In addition, Step I should be reexecuted at some periodic intervals. Through this mechanism, the location of nodes L_i and S_i, selected as the transmitting head nodes in each segment, will be changed in each period. This can balance the energy consumption of the nodes.

   After executing Step I and transmitting identities, nodes L_i and S_i, and nodes S_i and L_{i+1} will know each other. Figure 11 illustrates an example for the transmission step (i.e. Step II) at $R = 10$, in which notations R1 and R2 represent the conditions stated in Step II of Figure 10. Here, Segment i has run the *BinaryElectMiddle* procedure to determine L_i and S_i. On the other hand, the *BinaryElectLargest* and *BinaryElectSmallest* procedures are executed in Segment $i+1$ to determine L_{i+1} and S_{i+1}.

```
Procedure BinaryElectMiddle (N)          ; choose 2 mid-numbered fault-free nodes in segment N
 U = min(R / 2, F + 1)                   ; if F < R, then U = F + 1; otherwise, U = R

;choose the largest fault-free node number in right half of segment N
 ACT = {}
 For J = R / 2 to (R / 2 – U + 1) Step –1
   Append (ACT, ID of node a_NJ)         ; append from the middle toward the right of the segment to ACT
 End for

 I = 1
 While (I ≤ log2(U) )                    ; choose the largest-numbered node
      S = Smaller half of ACT
      L = Larger half of ACT
      Invoke all nodes in L to start transmission   ; fault-free nodes transmit simultaneously
      If message or noise is heard, then ACT = L    ; there is at least one fault-free node in L
                                 else ACT = S       ; L is removed when all nodes in L are faulty
      I = I + 1
 End while
 Remaining node in ACT transmits its identity      ; to the fault-free nodes in the segment
 Y = identity                                      ; to keep the segment active node identity

; choose the smallest fault-free node number in left half of segment N
 ACT ={}
 For J = R/2+1 to R/2+U
   Append (ACT, ID of node a_NJ)         ; append from the beginning of segment to ACT investigated
 End for

 I = 1
 While (I ≤ log2(U))                     ; choose the smallest-numbered node
      S = Smaller half of ACT
      L = Larger half of ACT
      Invoke all nodes in S to start transmission   ; fault-free nodes transmit simultaneously
      If message or noise is heard, then ACT = S    ; there is at least one fault-free node in S
                                 else ACT = L       ; S is removed when all nodes in S are faulty
     I = I + 1
 End while

 Remaining node in ACT transmits its identity      ; to the fault-free nodes in the segment
 X = identity                                      ; to keep the segment active node identity
 Return X, Y
End
```

Figure 9. The BinaryElectMiddle algorithm.

Obviously, in Step II of Figure 10, 2 transmissions occur in each segment. Hence, informing all of the domain nodes needs at most $2 \times D$ time units. In Step I, the BinaryElect procedures are invoked in parallel and we need to calculate only 4 times of invoking. Note that BinaryElectMiddle includes 2 binary search operations, where each binary search operations has the complexity of $a_1 \times \log_2(U)$. Since each BinaryElect is run in $c_1 + a_1 \times \log_2(U)$, Step I encounters a total complexity of $4c_1 + 4 \times a_1 \times \log_2(U) \in \mathrm{O}(\log_2(U))$. Notice that $\log_2$ notation denotes the logarithm in base 2. Here, c_1 and a_1 are 2 coefficient values for considering, respectively, the running time of the instructions outside of the loop and inside of the loop. Therefore, LINE_A1 can inform all of the nodes within the domain at most in $2 \times D + \mathrm{O}(\log_2(U))$ times, where in the worst case we have $U = \min(R, F + 1)$.

Initialize: *SchedulingType* = 0

Step I

```
SchedulingType = 1- SchedulingType
If (SchedulingType = 0), then
    For i = 2 to N_s step 2                ; do for all even segments in parallel
        L_i = BinaryElectLargest (i)       ; largest fault-free node in segment i
        S_i = BinaryElectSmallest (i)      ; smallest fault-free node in segment i
    End
    For i = 1 to N_s step 2                ; do for all odd segments in parallel
        L_i, S_i = BinaryElectMiddle (i)   ; 2 nodes from the middle of segment i
    End
Else
    For i = 2 to N_s step 2                ; do for all even segments in parallel
        L_i, S_i = BinaryElectMiddle (i)   ; 2 nodes from the middle of segment i
    End
    For i = 1 to N_s step 2                ; do for all odd segments in parallel
        L_i = BinaryElectLargest (i)       ; largest fault-free node in segment i
        S_i = BinaryElectSmallest (i)      ; smallest fault-free node in segment i
    End
End if
```

Step II

Source transmits a message at time r
Segment 1 *receives the message at time r and* L_1 *transmits it at time r* + 1
In each segment i:
R1: ***If*** *segment i receives the message from* L_i *at time u,* ***then*** S_i *transmits it at time u* + 1
R2: ***If*** *segment i* + 1 *receives the message from* S_i *at time y,* ***then*** L_{i+1} *transmits it at time y* + 1

Figure 10. The LINE_A1 scheduling algorithm.

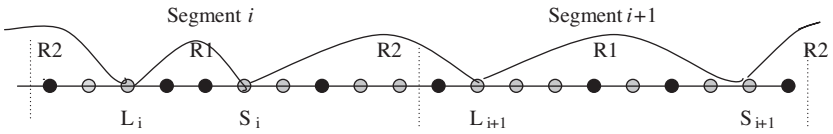

Figure 11. Representing Step II conditions in the LINE_A1 algorithm.

2. **LINE_A2:** The algorithm in Figure 12 attempts to speed up Step II of the LINE_A1 algorithm [23], provided that $F > 0$. Whenever F increases, Step I of Figure 12 should be rerun. Figure 13 depicts an example for the choice of L_j^i and Step II of the algorithm at R =6, $F = 5$, and $k = 2$ for 3 consecutive segments ($i - 1$, $i, i+1$). Node L_1^{i-1} transmits from segment $i - 1$. Since condition $v + R \geq L_3^i$ is true, the smallest p for $v + R \geq L_p^i$ is searched and the result is $p = 3$. Node L_3^i transmits from segment i. For segment $i+1$, condition $v + R \geq L_3^{i+1}$ is false, and therefore L_1^i transmits from segment i again.

The complexity of Step I in LINE_A2 is $2(k+1)(a_1 \times \log_2(U) + c_1) \in O(k \times \log_2(U))$, where $U = \min(R, F+1)$. In Step II, each segment transmits at least once and some segments transmit twice. To compute the upper bound for the MBT, we should consider the worst case scenario that happens when the fault configuration is as follows:

Step I

$U = \min(R, F + 1)$; if $F < R$, then $U = F + 1$; otherwise, $U = R$

$k = \left\lceil \sqrt{\frac{F}{\log_2(U)}} \right\rceil$; it will be proved why this value has been taken for k

For $i = 2$ ***to*** N_s ***step*** 2 ; do for all even segments in parallel

  For j=1 ***to*** k

    L_j^i = *the* jth *smallest-numbered fault-free node in segment* i

  End

  $L_{k+1}^{\ i}$ = *BinaryElectLargest (i)* ; largest fault-free node in segment i

End

For $i = 1$ ***to*** N_s ***step*** 2 ; do for all odd segments in parallel

  For j=1 ***to*** k

    L_j^i = *the* jth *smallest-numbered fault-free node in segment* i

  End

  $L_{k+1}^{\ i}$ = *BinaryElectLargest (i)* ; largest fault-free node in segment i

End

Step II

Source sends the message at time r, and node $L_1^{\ 1}$ (*in segment* 1) *transmits it at time* r+1

If *a message arrives at segment i from node v in segment* i–1 *at time r,* ***then***

  If $v+R \geq L_{k+1}^{\ i}$, ***then*** $L_p^{\ i}$ *transmits it at time* r+1; p is the smallest index for $v + R \geq L_p^{\ i}$,

  else $L_1^{\ i-1}$ *transmits at time* r+1; this is second transmission from segment i–1

Figure 12. The LINE_A2 scheduling algorithm.

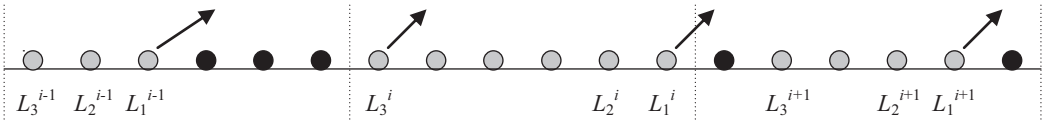

Figure 13. Representing Step II conditions in the LINE_A2 algorithm.

- In segment $i-1$, node L_1^{i-1} is forced to transmit the message (for the second time).
- In segment i, the Rth, and 1st to kth nodes are faulty. Hence, the $L_1^i = (k+1)$th node transmits the message.
- In segment $i+1$, the 1st to kth nodes are fault-free, and therefore, the smallest-numbered node in segment $i+1$ transmits.
- The fault configuration of segment $i+2$ is the same as that of segment i. Therefore, node L_1^{i+1} is forced to transmit the message for the second time.

According to the mentioned configuration, the number of faults between the segments that are forced to transmit for the second time is $k+1$. Thus, the number of segments that send twice is $[F/(k+1)] < [F/k]$. Therefore, the complexity will be $O(k \times \log_2(U)) + (D + [F/k])$. One can find a value for k to minimize $O(k \times \log_2(U)) + [F/k]$ by:

$$G(k) = k \times \log_2(U) + F/k \rightarrow G'(k) = \log_2(U) - \frac{F}{k^2} = 0 \rightarrow k = \sqrt{\frac{F}{\log_2(U)}}.$$

This is the value used in the first line of the LINE_A2 algorithm (see Figure 12). Using this value of k, the upper bound time on the MBT is computed by:

$$D + O(\sqrt{F \times \log_2(U)}) + ceil\left(\sqrt{F \times \log_2(U)}\right) = D + O(\sqrt{F \times \log_2(U)}), \text{ where } U = \min(R, F+ 1).$$

In the LINE_A2 scheduling algorithm, at least 1 node must transmit in each segment. In some cases, however, 2 nodes may be forced to broadcast within a segment. On the other hand, in the LINE_A1 scheduling algorithm, 2 nodes always broadcast in each segment. Therefore, Step II of LINE_A2 consumes less energy than Step II of LINE_A1. Nevertheless, Step I of LINE_A2 consumes a little more energy than Step I of LINE_A1. Thus, the LINE_A2 algorithm could be superior to the LINE_A1 algorithm.

4. Distribution of the sensor nodes on the SGT

Under the SGT distribution, a sensor field is divided into square grids and the sensor nodes are deployed on grid points in an $n \times n$ grid network. The location of a node is known with (i, j), where $0 < i, j < n+1$. Each square $\{(i-1)R+1, \ldots, i \times R\} \times \{(j-1)R+1, \ldots, j \times R\}, 0 < i, j \leq n/R$ is called a *cell*, similar to that in [23]. If a cell is divided into 4 squares of side $R/2$, each square is called a *tile*, similar to that in [23]. Obviously, the range of each node equals the square of side $2 \times R$. Two cells/tiles are neighbors if they are located at the side or corner of each other (see Figure 14). Since the worst case analysis is considered in computing the MBT, the source node is considered to be at location (1,1).

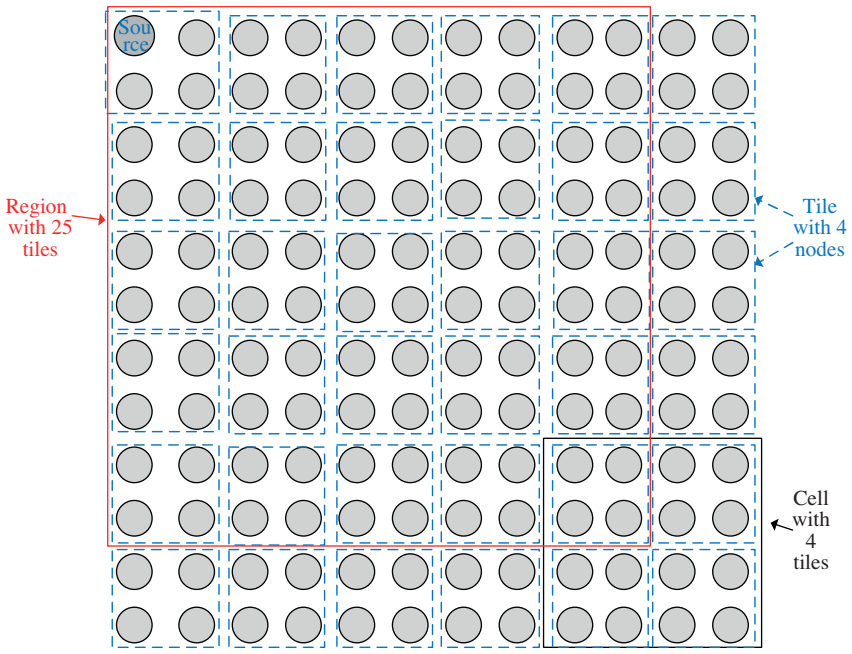

Figure 14. Grid distribution with $R = 4$ and a region with 5×5 tiles.

4.1. SGT nonadaptive (SGT_NA) algorithm

Here, a grid is visualized as regions of 5×5 tiles (as depicted in Figure 14), where the tiles are colored with 25 various colors in each region. Figure 15 depicts the SGT_NA scheduling algorithm. According to this scheduling, the lth nodes in the kth tiles of all of the regions transmit at the same time. For example, the first nodes in

the 10th tiles of all of the regions are scheduled to transmit at the same time. Note that the distance between these nodes should be enough to avoid collision.

Source node transmits a message at time 1
Node a_{ki} *transmits the message at time* $25\times(i-1) + C_k + 25\times S\times r$
where:

$r = 0, 1, 2,\ldots,$
$S = R^2 / 4,$
C_k = color of tile *k in a region*,
$i = 1, 2,.., S$
a_{ki} = *node i* (*with any order*) *in the k*th *tile*

Figure 15. Nonadaptive scheduling algorithm for the SGT topology.

When there is no faulty node in the network, each region of 5×5 is informed in 25 time units and it takes $25 \times \frac{n}{5\times(R/2)} = 10 \times \frac{n}{R} = 10 \times D$ time units to inform the whole domain. To consider the role of faults, the worst case of the fault configurations is assumed. Figure 16 shows this configuration for a cell with $R = 4$ and 4 tiles. Tile 1 has received a message and each tile includes $S - 1 = 3$ faults. After $25 \times (S - 1)$ time units, the fault-free node in tile 1 is scheduled to transmit. However, the last nodes of the neighbor tiles are faulty, and therefore the fault-free nodes in these tiles will be scheduled to transmit after at most $25 \times (S - 1)$ time units. If this structure is repeated, the worst case fault configuration is obtained. Thus, 2 faulty neighbor tiles with the worst case fault configuration cause $50 \times (S - 1)$ time units delay for $2 \times (S - 1)$ faults, i.e. an average of 25 time units delay for each fault. Having a total of F faults, the time required to inform the domain is at most $10 \times D + 25 \times F \in \mathrm{O}(D + F)$.

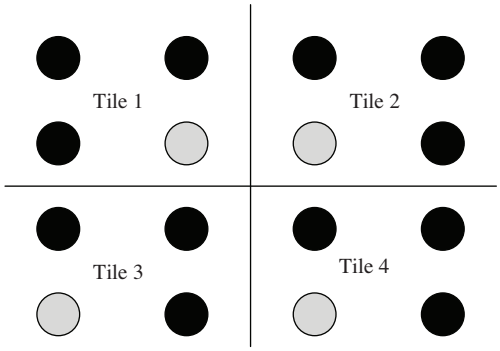

Figure 16. The worst case fault configuration.

It should be noted that a region of 5×5 is chosen in this paper to reduce the energy consumption of the sensor nodes for the same reason mentioned in Section 3.1. One should choose these regions at least in such a way that no collision happens when the corresponding nodes in the corresponding tiles of 2 neighboring regions transmit at the same time. Hence, the distance between any 2 regions must be at least 3. In this case, the scheduling time for 3×3 regions will be $9 \times (i - 1) + C_k + 9 \times S \times r$, and the MBT will be at most $6 \times D + 9 \times F \in \mathrm{O}(D + F)$ [23].

One may think that choosing 3×3 regions results in a smaller MBT of $6 \times D + 9 \times F$ compared with a MBT of $10 \times D + 25 \times F$ in 5×5 regions. Note that the case of 3×3 regions is equivalent to the method

stated in [23]. However, the energy consumption of the nodes increases in 3 × 3 regions, for the same reason discussed in Section 3.1. Therefore, the number of faulty nodes in 3 × 3 regions grows faster than 5 × 5 regions. Therefore, after a while, most of the nodes in 3 × 3 regions fail and the MBT goes up compared with 5 × 5 regions.

4.2. SGT adaptive algorithm

As in Section 3.2, a preprocessor procedure is executed to select the representative fault-free nodes in each cell. This procedure takes 2 sets of nodes (A, B), which are in the range of each other and selects in a binary manner 2 nodes as their representative nodes. The complexity of this procedure is $O(\log_2(\max(|A|, |B|)))$. For each pair of cells, this procedure is invoked and representative nodes are elected. In the broadcasting time, these nodes are responsible to transmit a received message. Therefore, by removing faulty nodes and introducing representative nodes, the MBT reduces to $O(D+ \log_2(U)$ for any configuration of at most F faults [23], where $U = \min(R, F+ 1)$.

5. Conclusion

In this paper, some techniques have been proposed for monitoring networks based on WSNs, where the locations of the faulty sensor monitoring nodes are unknown. The application of the proposed techniques is for the monitoring of narrow passages (such as indoor corridors, tunnels, or bridges) or 2-dimensional fields. Two topologies as line and SGT have been considered for locating the sensor nodes. The scheduling for the broadcasting time in wireless monitoring networks has been provided and analyzed by proposing 2 nonadaptive and adaptive broadcasting scheduling algorithms for each topology. In addition, the energy consumption of the nodes has been considered in the proposed algorithms. An analytical computation for the complexity of the MBT has been presented for the proposed algorithms. In short, the scheduling in nonadaptive algorithms is simpler than in adaptive algorithms. The analysis results show that adaptive algorithms need smaller MBTs than nonadaptive algorithms to inform the whole network domain. The Table compares the complexity of different algorithms for the MBT, where $U = \min(R, F+ 1)$.

Table. Comparison of the upper bound complexities.

Broadcasting algorithm	Complexity
Line nonadaptive algorithm	$O(D + F)$
Line adaptive algorithm 1	$2\times D+ O(\log_2(U))$
Line adaptive algorithm 2	$D + O(\sqrt{F \times \log_2(U)})$
SGT nonadaptive algorithm	$O(D + F)$
SGT adaptive algorithm	$O(D+ \log_2(U))$

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey", Computer Networks, Vol. 38, pp. 393–422, 2002.

[2] S. Kamel, A.G. Rahbar, "Heuristic surveillance of targets in sensor networks", IEEE 5th International Symposium on High Capacity Optical Networks and Enabling Technologies, pp. 46–51, 2008.

[3] L. Song, D. Hatzinakos, "A cross-layer architecture of wireless sensor networks for target tracking", IEEE/ACM Transactions on Networking, Vol. 15, pp. 145–158, 2007.

[4] K. Ren, K. Zeng, W. Lou, "Secure and fault-tolerant event boundary detection in wireless sensor networks", IEEE Transactions on Wireless Communications, Vol. 7, pp. 354–363, 2008.

[5] P. Sridhar, A.M. Madni, M. Jamshidi, "Hierarchical aggregation and intelligent monitoring and control in fault-tolerant wireless sensor networks", IEEE Systems Journal, Vol. 1, pp. 38–54, 2007.

[6] S.C. Geyik, B.K. Szymanski, "Multi-target tracking and identification by a vector of sensors", IEEE Military Communications Conference, 2008.

[7] S. Panichpapiboon, G. Ferrari, O.K. Tonguz, "Sensor networks with random versus uniform topology: MAC and interference considerations", IEEE Vehicular Technology Conference, pp. 2111–2115, 2004.

[8] R. Iyengar, K. Kar, S. Banerjee, "Low coordination wakeup algorithms for multiple connected-covered topologies in sensornets", International Journal of Sensor Networks, Vol. 5, pp. 33–47, 2009.

[9] N.A. Ab. Aziz, K.Ab. Aziz, W.Z.W. Ismail, "Coverage strategies for wireless sensor networks", World Academy of Science, Engineering and Technology, Vol. 50, pp. 145–150, 2009.

[10] W.C. Ke, B.H. Liu, M.J. Tsai, "The critical-square-grid coverage problem in wireless sensor networks is NP-complete", Computer Networks, Vol. 55, 11, pp. 2209–2220, 2011.

[11] S. Dolev, T. Herman, L. Lahiani, "Polygonal broadcast for sensor networks", 2nd IEEE Upstate New York Workshop Sensor Networks, 2003.

[12] J.N. Al-Karaki, A.E. Kamal, "Routing techniques in wireless sensor networks: a survey", IEEE Wireless Communications, Vol. 11, pp. 6–28, 2004.

[13] X. Mao, H. Chen, P. Qiu, Z. Zhang, "Energy-efficient scheduling for multiple access in wireless sensor networks: a job scheduling method", Computer Networks, Vol. 54, pp. 2137–2146, 2010.

[14] A.F. Liu, X.Y. Wu, Z.G. Chen, W.H. Gui, "Research on the energy hole problem based on unequal cluster-radius for wireless sensor networks", Computer Communications, Vol. 33, pp. 302–321, 2010.

[15] I. Bekmezci, F. Alagöz, "Delay sensitive, energy efficient and fault tolerant distributed slot assignment algorithm for wireless sensor network under converge cast data traffic", International Journal of Distributed Sensor Networks, Vol. 5, pp. 557–575, 2009.

[16] A. Jawahar, S. Radha, R. Sharath Kumar, "Capacity-preserved, energy-enhanced hybrid topology management scheme in wireless sensor networks for hazardous applications", International Journal of Distributed Sensor Networks, Vol. 2012, 2012.

[17] C.H. Lung, C. Zhou, "Using hierarchical agglomerative clustering in wireless sensor networks: An energy-efficient and flexible approach", Ad Hoc Networks, Vol. 8, pp. 328–344, 2010.

[18] M.K. Watfa, O. Mirza, J. Kawtharani, "BARC: a battery aware reliable clustering algorithm for sensor networks", Journal of Network and Computer Applications, Vol. 32, pp. 1183–1193, 2009.

[19] M. Cardei, S. Yang, J. Wu, "Algorithms for fault-tolerant topology in heterogeneous wireless sensor networks", IEEE Transactions on Parallel and Distributed Systems, Vol. 19, 545–558, 2008.

[20] A. Boukerche, A. Martirosyan, R. Pazzi, "An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks", Mobile Networks and Applications, Vol. 13, pp. 614–626, 2008.

[21] N. Li, J.C. Hou, "FLSS: a fault-tolerant topology control algorithm for wireless networks", ACM International Conference on Mobile Computing and Networking, 2004.

[22] M. Bahramgiri, M.T. Hajiaghayi, V.S. Mirrokni, "Fault-tolerant and 3-dimensional distributed topology control algorithms in wireless multihop networks", IEEE International Conference on Computer Communications and Networks, 2002.

[23] E. Kranakis, D. Krizanc, A. Pelc, "Fault-tolerant broadcasting in radio networks", Journal of Algorithms, Vol. 39, pp. 47–67, 2001.

[24] S. Guo, Z. Zhong, T. He, "FIND: faulty node detection for wireless sensor networks", ACM Conference on Embedded Networked Sensor Systems, 2009.

[25] F. Koushanfar, M. Potkonjak, A. Sangiovanni-Vincentelli, "On-line fault detection of sensor measurements", Proceedings of IEEE Sensors, Vol. 2, pp. 974–979, 2003.

[26] X. Luo, M. Dong, Y. Huang, "On distributed fault-tolerant detection in wireless sensor networks", IEEE Transactions on Computers, Vol. 55, pp. 58–70, 2006.