

# Chapter 4

## Faulty Detection Algorithm based on SensorRank

### 4.1 Overview of TrustVoting



As mentioned before, given a correlation network, we are able to assign SensorRank to each sensor. Sensors with higher SensorRank will have more confidence in voting process. In light of SensorRank, we propose a faulty reading filtering algorithm (referred to as TrustVoting) to detect and filter out the faulty readings so as to increase data accuracy.

Algorithm TrustVoting consists of two phases: self-diagnosis phase and neighbors diagnosis phase. In self-diagnosis phase, each sensor verifies whether the current reading of a sensor is unusual or not. Once the reading of a sensor goes through the self-diagnosis phase, this sensor can directly report the reading. Otherwise, the sensor has to consult with the neighbors to further verify whether the current reading is faulty or caused by an interesting event. If the readings are determined as faulty readings, these faulty readings will be filtered out. Those sensor generated faulty readings will not get involved in voting since these sensors are likely to bias the voting for other sensors. Note that algorithm TrustVoting is executed in a distributed manner. The execution order of algorithm TrustVoting will have an influence on faulty reading

detection. We will describe this phenomenon later. When a region of sensors are queried, each sensor in the region queried should first broadcast its SensorRank to its neighbors. According to SensorRank received from its neighbors, sensor  $s_i$  should determine when to perform self-diagnosis and neighbor diagnosis in algorithm TrustVoting. When receiving SensorRank of neighbors, sensor  $s_i$  will set the timer to execute self-diagnosis and neighbor diagnosis. With a higher SensorRank, sensor  $s_i$  will have more priority to first execute self-diagnosis and neighboring diagnosis due to that this sensor has more similar neighbors to consult with for verifying whether the reading is faulty or not. If the fault reading is determined, the corresponding sensor will have no chance to participate the voting for other sensors. As such, the dominated problem will be solved accordingly. The detailed descriptions of two phases and the order of executing TrustVoting are described in the following chapters.

---

**Algorithm** *TrustVoting*

**Input:** a sensor  $s_i$ , its SensorRank  $rank_i$  and time interval  $t$

**Output:** justify whether the reading is faulty or not (i.e.,  $faulty = true$  or not)

```

1: set  $faulty = false$ 
2: broadcast  $rank_i$  to the neighbors
3: receive  $\{rank_j | s_j \in nei(i)\}$  from the neighbors
   /*set timer according to the priority sorted by SensorRank*/
4: sort SensorRank values received
5:  $x = rank_i$ 's order in the sorted SensorRank values
6:  $n =$  neighbors of sensor  $s_i$ 
7:  $timer = x \cdot (\frac{t}{n+1})$ 
   /* $t$  is the time interval given*/

8: while  $time == timer$  do
9:    $faulty =$  Procedure Self-Diagnosis
10:  if  $faulty == true$  then
11:     $faulty =$  Procedure Neighbor-Diagnosis
12:    return  $faulty$ 

```

---

## 4.2 Self-diagnosis Phase

When a region of sensors are queried, each sensor performs procedure self-diagnosis to judge the current reading behavior. Once the reading behavior of a sensor is determined as normal, the sensor will not be in the neighbor diagnosis phase. To execute procedure self-diagnosis,

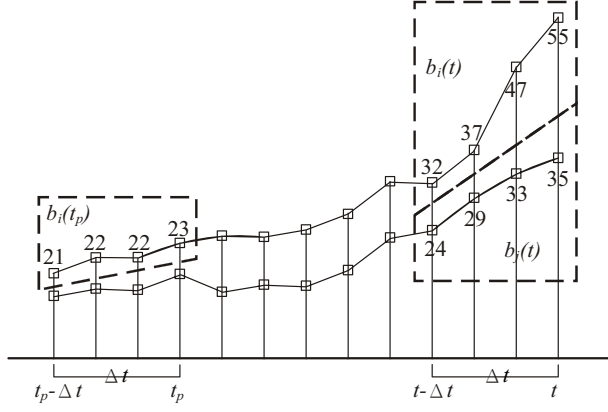


Figure 4.1: The reading behavior of sensors  $s_i$  and  $s_j$ .

each sensor  $s_i$  only maintains two reading behaviors: one is the current reading behavior at the current time  $t$  (denoted as  $b_i(t)$ ) and the other is the previous correct reading behavior at a previous time  $t_p$  (expressed by  $b_i(t_p)$ ).  $b_i(t_p)$  records a series of readings occurred in the previous time and is used for checking whether the current reading behavior is faulty or not. If these two reading behaviors are not similar,  $b_i(t)$  is viewed as an unusual reading behavior. Once a sensor is detected an unusual reading behavior, this sensor will go through neighbor-diagnosis phase to further justify whether the unusual reading behavior is faulty or is caused by an interesting event. Note that when  $b_i(t)$  is identified as a normal behavior through the neighbor diagnosis,  $b_i(t_p)$  is updated so as to reflect the current environmental status. For example shown in Figure 4.1, the previous reading behavior  $b_i(t_p)$  is  $\{21, 22, 22, 23\}$ , and the current reading behavior  $b_i(t)$  is  $\{32, 37, 47, 55\}$ . If  $c_{rand} = 0.5$  and  $\sigma = 0.7$ ,  $sim(b_i(t), b_i(t_p)) = 0.3 < \sigma$ . Assume that  $b_i(t)$  is identified as a normal reading behavior in the neighbor diagnosis phase,  $b_i(t_p)$  will be updated as  $\{32, 37, 47, 55\}$ .

---

#### Procedure *Self-Diagnosis*

**Input:** a sensor  $s_i$ , its previous reading behavior  $b_i(t_p)$ , its current reading behavior  $b_i(t)$ , and a threshold  $\sigma$ .

**Output:** the variable *faulty*.

- 1: **if**  $sim(b_i(t_p), b_i(t)) \geq \sigma$  **then**
  - 2:   return *false*
  - 3: **else**
  - 4:   return *true*
-

### 4.3 Neighbor diagnosis Phase

As mentioned above, if a sensor  $s_i$  sends  $b_i(t)$  to a neighbor  $s_j$ ,  $s_j$  will compare  $b_i(t)$  with its own current reading behavior  $b_j(t)$  and then give its judgment for  $b_i(t)$ . When  $s_i$  receives all the judgments from the neighbors,  $s_i$  has to determine whether  $b_i(t)$  is faulty or not according to the judgments. However, some of the judgments are more authoritative and some are not, which depends on SensorRank values of these neighbors. A sensor with higher value of SensorRank has more similar neighbors to consult with. Therefore, the judgments from the neighbors with higher values of SensorRank are more authoritative, whereas the judgments from the neighbors with lower values of SensorRank are less authoritative. Consider the above example in Figure 3.1,  $s_1$  senses unusual readings and asks its neighbors. Assume some of the neighbors think that  $s_1$  generates faulty readings, and the other sensors claim that  $s_1$  has correct readings. Now,  $s_1$  should make a decision among these judgments. Since  $s_5$  has the highest SensorRank (1.543) and has the most similar behavior ( $tr_{1,5} = 0.7$ ) with  $s_1$ ,  $s_5$  will have more authority than other neighbors. On the other hand,  $s_2$  has the lowest SensorRank (0.243) and is not similar to  $s_1$  ( $tr_{1,2} = 0.3$ ). Thus,  $s_2$  has fewer influence on the judgment.

When sensor  $s_i$  sends  $b_i(t)$  to all its neighbors for the neighbors' diagnosis, each neighbor should determine whether  $b_i(t)$  is faulty or not. A vote is returned to show the judgment. If a neighbor  $s_j$  considers  $b_i(t)$  is not faulty by comparing the similarity of the two reading behaviors (i.e.,  $sim(b_i(t), b_j(i)) \geq \sigma$ ),  $s_j$  will send a positive vote, denoted  $vote_j(i)$ , to  $s_i$ . Otherwise, the vote will be negative. In addition, the vote from  $s_j$  will be weighted by its SensorRank.

$$vote_j(i) = \begin{cases} rank_j, & sim(b_j(t), b_i(t)) \geq \sigma \\ -rank_j, & \text{otherwise.} \end{cases}, \quad (4.1)$$

After collecting all the votes from the neighbors,  $s_i$  has two classes of votes: one is positive class (consider  $b_i(t)$  as normal) and the other is negative class (consider  $b_i(t)$  as faulty). If the weight of the former is larger than the weight of the later, the most neighbors will view

---

**Procedure** *Neighbor-Diagnosis*

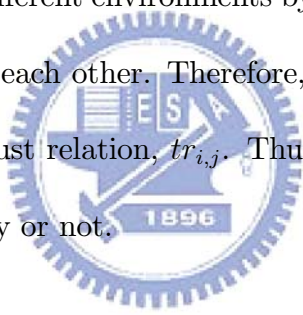
**Input:** a sensor  $s_i$ , its current reading behavior  $b_i(t)$ , and a threshold  $\sigma$ .

**Output:** the variable *faulty*.

```
1: set  $dec_i = 0$ 
2: broadcast  $b_i(t)$  to the neighbors
3: for all  $s_j \in nei(i)$  do
4:   if  $sim(b_i(t), b_j(t)) \geq \sigma$  then
5:      $vote_j(i) = rank_j$ 
6:   else
7:      $vote_j(i) = -rank_j$ 
8:      $dec_i = dec_i + tr_{ij} \cdot vote_j(i)$ 
9:   if  $dec_i \geq 0$  then
10:    return false
11: else
12:    return true
```

---

$b_i(t)$  as normal. Note that the weight of a vote represents how much authority the judgment has. It is possible that a neighbor  $s_j$  of  $s_i$  has higher SensorRank with a low trust relation. In this case, these two sensors sense different environments by nature, and thus these two sensors cannot provide good judgments to each other. Therefore, each vote (i.e.,  $vote_j(i)$ ) has to be multiplied by the corresponding trust relation,  $tr_{i,j}$ . Thus, we have the following formula to justify whether the reading is faulty or not.


$$dec_i = \sum_{s_j \in nei(i)} tr_{ij} \cdot vote_j(i)$$

If the weight of the positive votes are more than the weight of the negative votes,  $dec_i$  will be positive which means that  $s_i$ 's reading behavior is normal and the current reading can be aggregated. Otherwise,  $dec_i$  is negative, implying that the current reading of  $s_i$  is a faulty reading. For example in Figure 4.2, a region of sensors is queried ( $s_1, s_2, s_3, s_4$  and  $s_5$ ) and four faulty sensors (gray nodes) exist. SensorRanks of sensors are shown in square brackets in nodes and the values of trust relations between sensors are shown on edges. To facilitate the presentation of this example, the plus sign (minus sign) shows that two sensors have the similar (dissimilar) current reading behaviors, and they are going to give the positive (negative) votes to each other when executing the neighbors' diagnosis. Consider sensor  $s_5$  as an example,

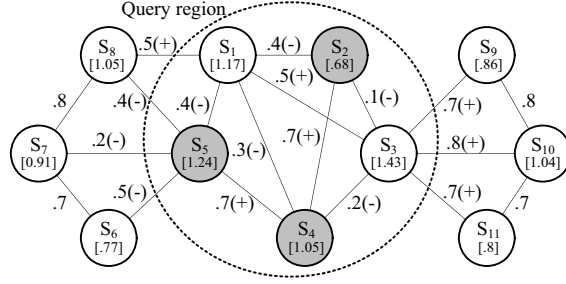


Figure 4.2: An example of casting votes.

where sensor  $s_5$  will receive the votes from its neighbors (i.e.,  $s_1, s_4, s_6, s_7$  and  $s_8$ ). It can be verified that  $dec_5 = (-1.08) \cdot 0.4 + 1.05 \cdot 0.7 + (-0.81) \cdot 0.5 + (-0.99) \cdot 0.2 + (-0.97) \cdot 0.4 = -0.688$ . Therefore, the reading reported by sensor  $s_5$  is a faulty reading and will be filtered out.

## 4.4 Execution Order of TrustVoting

Since algorithm TrustVoting is a distributed algorithm, sensors queried will perform procedures of self-diagnosis and neighbor diagnosis individually. It is possible that different execution orders will have different results for faulty detection. For example, consider two execution orders  $\{s_1, s_2, s_3, s_4, s_5\}$  and  $\{s_5, s_1, s_2, s_3, s_4\}$  in Figure 4.2, and assume that all the queried sensors have to perform the neighbors' diagnosis. In the order of  $\{s_1, s_2, s_3, s_4, s_5\}$ , when  $s_1$  executes TrustVoting,  $s_2, s_4$  and  $s_5$  will vote negative votes of  $-1.088$ , while  $s_3$  and  $s_8$  will vote positive votes of  $1.17$ . As such,  $dec_1$  will be  $0.082$  and  $s_1$  will be identified as normal. For  $s_2$ , since  $dec_2 = (-1.1) \cdot 0.4 + (-1.36) \cdot 0.2 + 1.04 \cdot 0.7 = 0.016$ ,  $s_2$  is identified as normal. In the similar way, we can find that  $s_4$  is also identified as normal. However, in the order of  $\{s_1, s_2, s_3, s_4, s_5\}$ , the result will be different. When  $s_5$  executes TrustVoting, it will be identified as faulty obviously because almost all neighbors give it negative votes. Therefore,  $s_5$  cannot vote for any other sensors. Without  $s_5$ 's vote  $s_4$  will be identified as faulty since  $(-1.1) \cdot 0.3 + 0.78 \cdot 0.7 + (-1.36) \cdot 0.2 = -0.056$ . As a result, we can see that different orders derive different results.

Order	Faulty	Not Faulty
$s_1, s_2, s_3, s_4, s_5$	$s_5$	$s_1, s_2, s_3, s_4$
$s_5, s_1, s_2, s_3, s_4$	$s_4, s_5$	$s_1, s_2, s_3$
$s_3, s_5, s_1, s_4, s_2$	$s_2, s_4, s_5$	$s_1, s_3$

Table 4.1: Faulty detection results under different orders.

As such, how to determine an appropriate order to perform procedures of self-diagnosis and neighbor diagnosis in algorithm TrustVoting will have an influence on the final result. Since algorithm TrustVoting is executed in a distributed manner, we could explore timer to control the execution order of procedures self-diagnosis and neighbor diagnosis. Those sensors having smaller values of timer will perform first. By exploring SensorRank, we could allow those sensors having higher SensorRank to perform procedures self-diagnosis and neighbor diagnosis as soon as possible. As pointed out early, sensors with higher SensorRank are likely to have more similar neighbors, thereby these sensors could be correctly identified whether readings are faulty or not. Once sensors reporting faulty readings are detected, these sensors have no right to get involved in voting for other sensor nodes. Therefore, the dominated problem can be solved under the scenario that those faulty sensors with higher weights could be determined as early as possible. Intuitively, we could determine the order of executing procedures of self-diagnosis and neighbor diagnosis according to SensorRank. Sensors with higher SensorRank should have a smaller timer. However, some sensors with higher SensorRank possibly result from those neighbors having higher SensorRank and have few neighbors. Therefore, we should not let such sensors execute procedures due to that with few neighbor sensors, these sensors are likely not to accurately identified as a faulty or not. One should also take the number of neighbors into considerations. In algorithm TrustVoting, timers are set for each sensor in accordance to SensorRank and number of neighbors. Specifically, assume that a time interval will be given in algorithm TrustVoting. In algorithm TrustVoting, each sensor should first broadcast SensorRank to neighbors. Once receiving SensorRank values from its neighbors, each sensor should sort SensorRank values in a decreased order. Then, each sensor should

determine the order of its SensorRank in such sorted list. Furthermore, a sensor will have information related to the number of neighbors from SensorRank values received. Therefore, we could set timer to be  $x \cdot \frac{t}{(n+1)}$ , where  $x$  is the order of this sensor in a sorted list,  $n$  is the number of neighbors and  $t$  is the time interval given. With a smaller value of timer, procedures of self-diagnosis and neighbor-diagnosis will be executed first. Consider an illustrative example in 4.2. The timer value for sensor  $s_3$  should be  $1 \cdot \frac{t}{7}$  since sensor  $s_3$  has 6 neighbors and its SensorRank is the highest among SensorRank values collected (i.e., 6 neighbors and sensor  $s_3$ ). Following the same operation, we could have the timer values  $\frac{2t}{6}$ ,  $\frac{3t}{3}$ ,  $\frac{3t}{3}$ , and  $\frac{2t}{6}$  for  $s_1$ ,  $s_2$ ,  $s_4$  and  $s_5$ , respectively. Assume that each sensor does not go through self-diagnosis and will execute procedure of neighbor diagnosis. According to the timers derived,  $s_3$  will perform first and the reading of  $s_3$  is identified as a normal reading through neighbor diagnosis. Then, both  $s_1$  and  $s_5$  will execute neighbor diagnosis at the same time. The reading reported by  $s_1$  (respectively,  $s_5$ ) will be determined as a normal reading (respectively, faulty). Follow the executions of  $s_2$  and  $s_4$ . In particular, since  $s_5$  is viewed as faulty,  $s_5$  could not participate in voting process of  $s_4$ . As a result, through the execution order derived, we could accurately detect faulty readings reported by  $s_2$ ,  $s_4$  and  $s_5$ .