Fault-tolerant Communications by reservation-based protocol in *IoT* network

1st Boudaa Abdelghani
Département d'informatique (Faculté de M.I.)
Université Med BOUDIAF (M'sila)
M'sila, Algerié
ghani.boudaa@gmail.com

2nd Belouadah Hocine
Département de Sciences Exactes
Ecole Normale Supérieure (Bou Saada)
Bou Saada, Algerié
r_belouadah74@yahoo.fr

***Abstract*—The Internet of things *(IoT)* consists of a great number of heterogeneous nodes. Things are equipped with data processing and communication capabilities. These things are useful in a wide range of applications Such applications smart energy,smart health etc., so they use the permutation protocol and make our daily life smarter. Among the most important problems related to the *IoT*, there are constraints of energy and fault tolerance. Thus, we designed a protocol that provides fault-tolerant communications by use of reservation-based protocol.**

***Index Terms*—Internet of things, Single-hop networks, Permutation Routing, Parallel broadcasting, reservation-based protocol, Energy-efficiency, fault-tolerant communications**

I. Introduction

Internet of Things is a collection of entities which can be physical devices, animals, people, electronic devices etc. *IoT* is now becoming a vital instrument to interconnect devices. *IoT* Communication over the internet has grown from *user-user interaction* to *devicedevice interactions* these days [5]. As said Lakhlef et al. [13], *IoT* will occupy a place of choice in our everyday life.

The permutation routing problem is a useful abstraction for most routing problems in distributed systems [7]. Each node must send information from its own memory to allow its neighbours to progress, while minimizing the total number of retransmissions.

The nodes are devices running on batteries and the batteries cannot be recharged while on a mission [3]. Indeed, due to the resources limitation, a solution for an application in Internet of things should take into account the restrained capabilities of these heterogeneous devices [10], [9], [4].

CSMA (Carrier Sense Multiple Access) is a simple and robust random access method for wireless networks [**?**]. However, a fraction of the available bandwidth is wasted for resolving random conflicts of messages [8]. Fine et al. [18] proposed DAMA (Demand Assignment Multiple Access) for transmission networks. This protocol provides conflict-free transmission using distributed access protocols with bounded delay. The main idea behind the DAMA scheme is that the nodes that wish broadcast on a given channel are ordered in a logical ring, according to which they are granted broadcast access to the channel [18]. Sivalingam et al. [6] have found that the CSMA protocol requires higher energy consumption than the DAMA protocols. In a DAMA or reservation-based protocol, collisions are avoided by reserving channels. Nakano [14] uses a local clock that synchronizes the time by interfacing with a GPS(*Global Positioning System*). Time is divided into slots and all packets transmissions take place at slot boundaries [1], [2].

In this paper, we are interested in designing a fault-tolerant reservation-based DAMA protocol for permutation routing in the *IoT* where each node is within the transmission range of all other nodes and the mobility duration is weaker than the time taken by a protocol to complete.

II. Related Works

Recently, the permutation routing problem has been explored for wireless networks in several papers [16]. We refer the reader to Fig. 1 for an illustration of the permutation routing problem with $n = 32$ items and $p = 8$ stations. For simplicity, for each item, we only indicate its destination station. As an example, station $S(1)$ initially stores $\frac{n}{p} = 4$ items destined to stations $S(3)$, $S(5)$, $S(6)$ and $S(8)$. As in

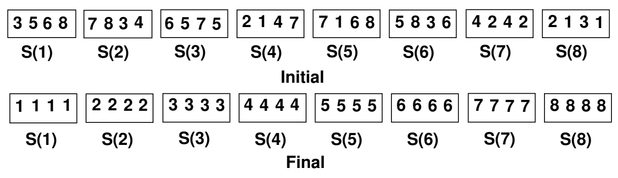

Fig. 1. Permutation routing with $p = 8$ and $n = 32$

the paper by Nakano et al. [16], we are interested in designing a reservation based DAMA protocol [18] for permutation routing on a wireless networks. Nakano et al. [15] have designed a protocol that runs in $\dfrac{2n}{k}+k-1$ time slots subject to k, the number of channels, satisfying $k \leq \sqrt{\dfrac{p}{2}}$. The protocol by Nakano et al. [15] runs extremely fast at the expense of high energy consumption, since each station must be awake for $\dfrac{2n}{k} + k - 1$ time slots. Datta and Zomaya [11] have shown that the permutation routing problem of n packets on a wireless network of p stations and k channels can be solved in $\frac{2n}{k} + (\frac{p}{k})^2 + p + 2k^2$ slots and each station needs to be awake for at most $\frac{6n}{p}+\frac{2p}{k}+8k$ slots. In [11], [16], they assume that each station works correctly during its lifetime. If there are faulty stations, the main problem with the protocols in [11], [16] is that not fault-tolerant. Amitava Datta [7] assumes that the routing protocol works correctly when there are faulty stations. The fault-tolerant permutation routing in a $WN(p,k)$ can be takes $\frac{2n}{k}+(\frac{p}{k})^2+\frac{p}{k}+\frac{3p}{2}+2k^2-k$ slots.

III. Contributions

The main contribution of this work is to present a protocol for *IoT* which should work well even when some of nodes are faulty and it should be energy efficient. We start by grouping the T thing nodes according to number and memory spaces M_i of the nodes t_i. So, to optimize the diffusion, we use parallel channels. Next, in every group, each thing node broadcasts its items to destination agents. Finally we broadcast the items to their final destination.

Outline of the paper: The rest of the paper is organized as follows: Section 4 presents the preliminaries. We proposed an overview of our routing protocol in Section 5. Next, we present our permutation routing protocol in Section 6. Finally, our conclusions and are given in section 7.

IV. Preliminaries

We consider a single-hop network where each thing node can communicate directly with the others thing nodes. The computation and communication capabilities are different for all thing nodes. The thing nodes communicate with each other using bidirectional links. The thing nodes are dispersed in a thing field and include a large number of static and mobile things. We consider a network of thing with n items and T nodes, *IoT(n,T)* for short. In addition, the things have different memory capacities; each thing t has M_t items in its local memory. We refer the reader to Fig. 2 (a) depicting an *IoT (8, 32)*. In *IoT (T,n)*, each thing t_i has its local memory M_i items. Figure 2 (b) presents the network after the permutation routing, where each thing node has its item and can proceed to execute its task [12].

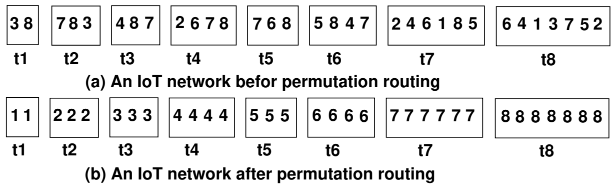

Fig. 2. An example of *IoT* before and after permutation routing

Let $LT=\{t_1,t_2,..,t_T\}$ be the list of T things in *IoT(T,n)* and let $LM=\{M_1,M_2,..,M_T\}$ be the set of the memory spaces that hold the items of the things in the network. With M_i the memory space of thing t_i , for each $0\leq i\leq T$. We note, $MIN(M_i)$ (respectively $MAX(M_i)$) the lowest (respectively the largest) memory space for items that a thing has in the network .

V. An overview of our protocol

In this section, we present our proposed protocol. It uses $O(\frac{n}{k})$ memory on *IoT(T, n, k)*, where k is the number of channels. We assume that the permutation routing on an *IoT(T,n,k)* of T things, n items and k channels, where each thing has a memory space of $O(\frac{n}{k})$, is possible when:

$$k\leq\sqrt{T} \tag{1}$$

we need at least k of agent nodes in each group to receive all items. Our protocol consists of three steps: Grouping thing nodes, Broadcast items to groups and Broadcast items to the final destination.

First step: We partition the T things t_i, $1\leq i\leq T$, into k groups $G(1),G(2),..,G(k)$ with $\frac{T}{k}$ the number of thing nodes in each group and M_j is the number of items for each thing node t_j ,$1\leq j\leq\frac{T}{k}$. Note that the maximum value possible for grouping is $k_{max}=\sqrt{T}$, in each group we select k agent nodes which receive the items that belong to a given group. Unlike the previous work of [12], our selection method depends on on both two parameters at a time, the maximum memory spaces M_i and the number of T things. Then, the nodes are divided into identical groups with same number of nodes. A size, $S(j)$, of a group is defined as the sum of all M_i, $i\in\{1,2,..,T\}$, that a group $G(j)$ contains, with different sizes.

Second step: In this step, each thing node t_i determines all working and faulty thing nodes in its group, and gets all agent thing nodes of all network $IoT(T,n)$. Each group has k agent nodes. The k agent in every group $G(j)$ receive n_m items composed of a set $\{M_i\}$,$i\in\{(m-1)\frac{T}{k}+1,...,m\frac{T}{k}\}$ that have destination addresses in $G(j)$. In each group $G(m)$ and in parallel, the $\frac{T}{k}$ nodes use the channel $C(m)$ to transfer all these items to their respective destination groups. But, at this point the items are only sent to agent nodes not necessarily to the correct destination thing node in the group. Note that each agent $t_{am}\,(G(j))$ in the group $G(j)$ may be the destination of all things in its group, because all the n_m items in group $G(m)$ may have all destinations in the unique group $G(j)$. On this basis, we must have for one agent node t_{am} a memory space capacity M_{am} equal to the largest size group to store all $\{M_i\},(m-1)\frac{T}{k}+1\leq m\leq\frac{T}{k}$ items of a group $G(m)$.

Lemma 5.1:
Let M_i, $i\in\{1,2,..,T\}$, memory space values of a group with the condition $k=\sqrt{T}$; then the largest size $S(lg)$ of a group, will be with a size of:

$$S(lg)\leq\frac{T}{k}MAX(M_i) \tag{2}$$

proof:
The best grouping for the largest group is when a group contains only $MAX(M_i)$ for example, and the rest of the groups contain the other $Mi,\frac{T}{k}+1\leq i\leq T$, so we can write:

$$n=\sum_{i=1}^{T/k}(M_i)+\sum_{i=(T/k)+1}^{T}(M_i)$$

$$S(G(1)) = n - \sum_{i=(T/k)+1}^{T} (M_i)$$

$$S(lg) \leq \left(\sum_{i=1}^{T/k} MAX(M_i) + \sum_{i=(T/k)+1}^{T} (M_i) \right) - \sum_{i=(T/k)+1}^{T} (M_i)$$

$$\sum_{i=1}^{T/k} MAX(M_i) = \frac{T}{k} MAX(M_i)$$

so the largest group $S(lg)$, is: $S(lg) \leq \frac{T}{k} MAX(M_i)$

With this condition an agent node will use in the worst case $O(\frac{n}{k})$ of memory space because in each group the overall number of items will be $\frac{T}{k} MAX(M_i)$ items.

Lemma 5.2:
Let M_i, $i \in \{1, 2, .., T\}$, memory space values of a group with the condition $k = \sqrt{T}$, then the smallest size $S(ls)$ of a group, will be with a size of:

$$S(ls) \geq \frac{T}{k} MIN(M_i) \tag{3}$$

proof:
The worst grouping for the smaller group is for a group that contains only $MIN(M_i)$ for example, and the rest of the groups contain the other M_i, so we can write:

$$n = \sum_{i=1}^{T/k} (M_i) + \sum_{i=(T/k)+1}^{T} (M_i)$$

$$S(G(1)) = n - \sum_{i=(T/k)+1}^{T} (M_i)$$

$$S(ls) \geq \left(\sum_{i=1}^{T/k} MIN(M_i) + \sum_{i=(T/k)+1}^{T} (M_i) \right) - \sum_{i=(T/k)+1}^{T} (M_i)$$

$$\sum_{i=1}^{T/k} MIN(M_i) = \frac{T}{k} MIN(M_i)$$

Therefore the least group $S(ls)$, is: $S(ls) \geq \frac{T}{k} MIN(M_i)$

With this condition an agent will use in the best case $O(\frac{n}{k})$ of memory space because in each group the overall number of items will be $T/kMIN(M_i)$ items.

At the end of the second step, each group $G(j)$, $1 \leq j \leq k$, has received all the n_m items whose destinations are thing nodes in group $G(j)$. However, we have to ensure that any items destined for $G(j)$ are not sent to faulty agent nodes.

Third Step: In the third step, we again assign channel $C(i)$ to $G(i)$ and route at most $\frac{T}{k} MAX(M_i)$ items to their correct destination nodes within $G(i)$. This routing is done in parallel in all the groups using channel $C(i)$ for group $G(i)$. All the items destined for the faulty nodes can be dropped before they are sent to their final destinations.

VI. Permutation routing protocol on *IoT (T,n)*

Our protocol is composed of three steps: *grouping thing nodes*, *broadcast items to its group* and *broadcast to the correct thing node*.

A. Grouping thing nodes

The nodes in the network are divided into k groups, $G(1), G(2), .., G(k)$ each group contains $\frac{T}{k}$ thing nodes. Each group must have at least k thing nodes, otherwise the condition, $k \leq \sqrt{T}$, is not verified. We use the grouping algorithm presented in Figure 3 to achieve the grouping in the network. This algorithm takes in input a list of memory spaces $Lm = \{M_1, M_2, .., M_T\}$; and an integer k which represents the number of channels which does not exceed $\sqrt{T}$. It outputs the list LG of groups $G(1), G(2), .., G(k)$.

Firstly we put in group $G(1)$, the thing node which has $MAX(M_i)$ of $Lm = \{M_1, M_2, .., M_T\}$, in $G(2)$ the thing node which has $MAX(M_i)$ of $Lm - MAX(G(1))$ and so on until $G(k)$ where we put in it, $MAX(M_i)$ of $Lm - MAX(G(k-1))$. After, at each step we add the maximum of list $Lm = \{M_1, M_2, .., M_T\} - MAX(G(i))$, $1 \leq i \leq k$, that does not contain the elements already added into the k groups, to the group $G(j)$ that has the current minimum $S(j)$. But, if one of k groups of list Ln is full, i.e, that we have added $\frac{T}{k}$ thing nodes, we remove it from its list Ln. We continue the last step, by adding the $MAX(Ln)$ into not full minimum group, and removing full groups from list until Ln is empty. By applying Algorithm 1:*Grouping thing nodes* , each group will have at the end $\frac{T}{k}$ thing nodes. This step is done locally and we can see that the overhead involved in assigning each thing node t into a group $G(i)$ doesn't take any time slot.

B. Broadcast items to its group

The main role of this procedure is to transmit the items of each thing node to agent nodes of the destination group. The computation is discussed with respect to $G(i)$, $1 \leq i \leq k$, but the same computation is done in parallel in all the groups using separate channels. We assign channel $C(i)$ to group $G(i)$ and we expect that there is at least k working thing nodes in each group. We denote the j^{th} thing node in group $G(i)$ by $t_j(G(i)$, $1 \leq j \leq \frac{T}{k}$; and the j^{th} agent thing node in group $G(i)$ by $t_{aj}(G(i))$, $1 \leq j \leq k$. With the condition (2) a thing node $t_j(G(i))$ will use in the worst case the largest size $S(lg)$, of a group divided by number of thing nodes in it $\frac{T}{k}$, so we can write : $\frac{S(lg)}{T/k} = MAX(M_i)$ of memory spaces. The principal tasks in this procedure are as follows:

1) Each thing node $t_j(G(i)$, in group $G(i)$, gets the *IDs* of all working thing nodes and all agent nodes $t_{aj}(G(i))$ in its group.
2) Each agent thing node $t_{aj}(G(i)$, in group $G(i)$, informs all the other groups about its *ID* and update faulty agent nodes if there are more than k working thing nodes in its group.
3) Each thing node $t_i(G(j)$ of the groups $G(j)$, $1 \leq j \leq k$, broadcasts their items to unique thing agents $t_{aj}(G(i))$ of group $G(i)$.

Algorithm 1 Grouping thing nodes

```
INPUT: set $Lm = \{M_1, M_2, .., M_T\}$
OUTPUT: $LG = \{G(1), G(2), .., G(k)\}$
  $Ls \leftarrow Lm$
  $Ln \leftarrow LG$
  while $Ln \neq \phi$ do
    if $|G(MIN(Ln))| < T/k$ then
      $G(MIN(Ln)) \leftarrow MAX(Ls)$
      $Ls \leftarrow Ls - MAX(Ls)$
    else
      $Ln \leftarrow Ln - G(MIN(Ln))$
    end if
  end while
  $i \leftarrow k$
  while $Ls \neq \phi$ do
    $G(i) \leftarrow MAX(Ls)$
    $Ls \leftarrow Ls - MAX(Ls)$
    $i \leftarrow i - 1$
  end while
  return $(LG)$
```

Algorithm 2 $Faulty\_nodes(t_i(G(j)))$

```
{do in parallel for each $G(j)$ on channel $C(j)$}
OUTPUT: $f_i$ : Number of faulty thing nodes
$Status\_nodes[]$ : Array of thing nodes status in $G(j)$
  if $Time\_to\_Broadcast(t_i(G(j)))$ then
    Thing $t_i(G(j)$ broadcasts $ID(t_i(G(j))$ on $C(j)$
  else
    if Thing $t_i(G(j)$ receives $ID(t_m(G(j))$ then
      $Status\_nodes[t_m(G(j)] \leftarrow true$
    else
      $Status\_nodes[t_m(G(j)] \leftarrow false$
      $f_i \leftarrow f_i + 1$
    end if
  end if
  return $Status\_nodes[], f_i$
  {Definition of Function: $Time\_to\_Broadcast()$}
  Function $Time\_to\_Broadcast$: $t_i(G(j))$
  $d \leftarrow 0$
  for $h \leftarrow 1, h < i, h++$ do
    $d \leftarrow d + 1$
  end for
  return $(d)$
```

1) Task 2.1: This task has the main role of determining all failed f_i thing nodes and all k agent nodes, in each group $G(i)$,$1 \leq i \leq k$. This task consists of two subtasks :

a) Subtask 2.1.1: In the first subtask, each thing node in group $G(i)$ broadcasts its *ID* one after another to all thing nodes in its group *G(i)*. This broadcast is done in parallel using channel $C(i)$, $1 \leq i \leq k$. Every node knows the slot when to broadcast its *ID* but the faulty node do not broadcast. Finally, each thing node assigns a correct serial number to itself among all the working thing nodes in group $G(i)$.

We use for this subtask Algorithm 2: $Faulty\_nodes$. This task takes $\frac{T}{k}$ slots and each thing node remains awake for $\frac{T}{k}$ slots.

b) Subtask 2.1.2: The next subtask is to select the agent nodes in each group $G(i)$. The role of the k agents $t_{aj}(G(i))$, $1 \leq j \leq k$, in every group $G(i)$, is to receive in their local memory spaces all the n_m items of different groups $G(j)$, $1 \leq j \leq k$, that have destination addresses in group $G(i)$. We must choose, as agents, the k working nodes that have the maximum memory spaces for items in $Lm = \{M_1, M_2, .., M_T\}$. This choice therefore consists of performing the least number of broadcast rounds in each group.

Let $t_1(G(j)), t_2(G(j)), ..., t_{T/k}(G(j))$ the nodes in $G(j)$, $1 \leq j \leq k$, where $t_{a1}(G(j)), t_{a2}(G(j)), .., t_{ak}(G(j))$ represent the k agents among the T/k nodes of this group. Note that this selection can be done locally in each thing node and does not require any time slots.

2) Task 2.2: We have to ensure that any items are not sent to faulty agents in $G(i)$. All thing nodes in the same group $G(i)$, have complete informations about all the working, faulty nodes and agents in it. However they don't know the *IDs* of the working agents in the other groups. So, by applying the Algorithm 3: $Select\_well\_agent$, each agent $t_{aj}G(i)$, $1 \leq j \leq k$, in $G(i)$ broadcasts its *ID* to all nodes in all network. We use one channel $C(1)$ to send the *IDs* of the agents. When agent $t_{aj}(G(i))$ broadcasts its *ID*, all the thing nodes listen to these broadcasts. If nodes don't receive the *ID* of agent $t_{ap}(G(m))$ it means the agent is faulty, so we increment fa_i: the number of faulty agents in all $IoT(T, n)$ network. Next, nodes of $G(i)$ choose another agent node for this group and we add one time slot for next broadcasting; but if the total number of working nodes is less than k then all the nodes in network may reinitialize the nodes in the whole network and restart the permutation routing protocol.

This subtask uses one channel $C(1)$ and all the thing nodes remain awake during the broadcast of *IDs*. It takes on average $k + fa_i/k$ time slots to complete for each group, i.e., $k^2 + fa_i$ time slots overall, and each thing node remains awake for $k^2 + fa_i$ time slots.

3) Task 2.3: In this task, we need to specify the exact slots when t_i will transmit its items in order to avoid collisions and subsequent item losses. Each node t_i knows the list of memory spaces. This start time is the sum of this memory spaces $\sum_{p=1}^{i} M_{t_p}$ slots. Therefore the process takes at most $\frac{T}{k}$ and at least $\frac{T}{k} - f_i$ periods, if f_i faulty thing nodes exist. In the i^{th} period, $1 \leq i \leq \frac{T}{k} - f_i$, the node t_i in $G(j)$ routes its M_{t_i} items with destination to agent in $G(m)$. The transmission takes $|M_{t_i}|$ slots but it starts at $|M_{t_1}| + |M_{t_2}| + .. + |M_{t_{i-1}}|$ slots. Hence, all nodes know when to wake up and transmit their items.

We use Algorithm 4: $Transmit\_items\_to\_Agents$ to perform this task. Each thing node has to remain awake for at most $MAX(M_{t_i})$ slots to transmit its items an overall $\frac{T}{k}MAX(M_{t_i})$ slots for transmission and they have to remain

Algorithm 3 $Select\_Well\_Agent(t_j(G(i)))$

```
{use one channel $C(1)$ for all thing nodes in $IoT(T,n)$}
OUTPUT: $fa_i$ : Number of faulty agent;
$Status\_agents[]$ : Array of $k$ agents status

  if $Time\_to\_Broadcast(t_j(G(i)))$ AND $t_j(G(i))$ is agent $t_{ap}(G(m))$ then
    Thing $t_j(G(i)$ broadcasts its $ID$ on channel $C(1)$
  else
    Thing node $t_j(G(i)$ receives $ID$ of agent $t_{ap}(G(m))$
    if $ID$ of agent is receipted then
      $Status\_agents[t_{ap}(G(m))] \leftarrow true$
    else
      $fa_i \leftarrow fa_i + 1$
      $Status\_agents[t_{ap}(G(m))] \leftarrow false$
      Select new agent $t_{ap}(G(m))$ for $G(m)$
      if new agent Exist then
        Insert new agent in $Status\_agents[]$ after the old faulty agent
        Add one Time slot for broadcasting a new agent
      else
        RESTART the permutation routing protocol
      end if
    end if
  end if
  return $Status\_agents[]$
  {Definition of Function: $Time\_to\_Broadcast()$}
  Function $Time\_to\_Broadcast$: $t_{ap}(G(m))$,$fa_i$
  $d \leftarrow 0$
  for $h \leftarrow 1, h < p + (m-1) * k, h++$ do
    $d \leftarrow d + 1 + fa_i$ {add $fa_i$ Time slots for Broadcasting}
  end for
  return $(d)$
```

awake for at most $\frac{T}{k}MAX(M_{t_i})$ slots to receive items as an agent thing node.

This task takes overall at most $2\frac{T}{k}MAX(M_{t_i})$ time slots; and all the thing nodes remain awake for at most $2\frac{T}{k}MAX(M_{t_i})$ time slots.

C. Broadcast to the correct thing node

At the end of step 2, all agents $t_{aj}(G(i))$, $1 \leq j \leq k$, in $G(i)$ hold at most $S(i) \leq \frac{T}{k}MAX(M_i)$ with destination in $G(i)$.

Each agent in $G(i)$ has complete information about the faulty nodes of $G(i)$, all the items destined for the f_i faulty nodes can be dropped before they are sent to their correct destinations. Therefore, the main concern is managing the broadcast on each channel $C(i)$ because node will be the destination of different agents $t_{aj}(G(i))$ of its group $G(i)$ at the same time. The first task plays, essentially, a channel reservation role. Finaly, the second task is to broadcast items in parallel for each group $G(i)$, $1 \leq i \leq k$,on channel $C(i)$.

1) Task 3.1: First, we determine the list Lp of memory spaces M_{a1}, $M_{a2}, .., M_{ak}$ of items in agent nodes $t_{a1}, .., t_{ak}$.

Algorithm 4 $Transmit\_items\_to\_Agents(t_i(G(j)))$

```
{do in parallel for each $G(j)$ on channel $C(j)$}
INPUT: set $L = \{M_{t_1}, M_{t_2}, .., M_{t_{\frac{T}{k}-f_i}}\}$ in $G(j)$

  if $Time\_to\_Broadcast(t_i(G(j)))$ then
    for $s \leftarrow 1, s \leq M_{t_i}$ do
      Thing $t_i(G(j))$ broadcasts items to $t_{aj}(G(m))$
    end for
  else
    if $t_i(G(j)$ is the agent $t_{aj}(G(m))$ then
      Agent $t_{aj}(G(m))$ copies items in its local memory
    else
      Thing $t_i(G(j))$ drop items {not destined for it}
    end if
  end if
  {Definition of Function: $Time\_to\_Broadcast()$}
  Function $Time\_to\_Broadcast$: $t_i(G(j))$,$M_{t_i}$
  $d \leftarrow 0$
  for $h \leftarrow 1, h < i, h++$ do
    $d \leftarrow d + M_{t_i}$ {$d = |M_{t_1}| + |M_{t_2}| + .. + |M_{t_i}|$ }
  end for
  return $(d)$
```

Next, we can compute the exact time to broadcast them. We note that k broadcast rounds suffice to fill the list $Lp = \{M_{a1}, M_{a2}, .., M_{ak}\}$. Now, a simple addition operation allows to each agent $t_{aj}(G(i))$, $1 \leq j \leq k$, to know the exact moment $|M_{a1}| + |M_{a2}| + .. + |M_{aj}|$ slots to broadcast. The details of the procedure are in the Algorithm 5: *time_to_send*. This task takes overall k time slots and the entire agent nodes remain awake for k time slots.

Algorithm 5 $Procedure\ Time\_to\_send(t_{aj}(G(i)))$

```
{do in parallel for each $G(i)$ on channel $C(i)$}
OUTPUT: set $Lp = \{\}$ memory spaces of $k$ agents

  Call $(Time\_to\_sendt_j(G(i)), Lp)$
  Procedure: $Time\_to\_send$: $t_j(G(i))), Lp$
  if $Time\_to\_Broadcast(t_j(G(i)))$
     AND $t_j(G(i))$ is agent $t_{aj}(G(i))$ then
    $Lp \leftarrow Lp + \{M_{aj}\}$ { put $\{M_{am}\}$ in the set $Lp$ }
    Agent $t_{aj}(G(i))$ broadcasts $M_{aj}$ in group $G(i)$
  else
    Agent $t_{aj}(G(i))$ receives $M_{am}$ on channel $C(i)$
    $Lp \leftarrow Lp + \{M_{am}\}$ { put $\{M_{am}\}$ in the set $Lp$ }
  end if
  End Procedure
  {Definition of Function: $Time\_to\_Broadcast()$}
  Function $Time\_to\_Broadcast$: $t_{aj}(G(i))$
  $d \leftarrow 0$
  for $h \leftarrow 1, h < j, h++$ do
    $d \leftarrow d + j$
  end for
  return $(d)$
```

Step		Max. completion Time slots	Max. awake Time slots
Step2	Task 2.1	$\frac{T}{k}$	$\frac{T}{k}$
	Task 2.2	k^2	k^2
	Task 2.3	$2\frac{T}{k}MAX(M_i)$	$2\frac{T}{k}MAX(M_i)$
Step3	Task 3.1	k	k
	Task 3.2	$2\frac{T}{k}MAX(M_i)$	$2\frac{T}{k}MAX(M_i)$

TABLE I
THE NUMBER OF SLOTS FOR COMPLETION AND THE MAX. NUMBER OF SLOTS THAT A NODE REMAINS AWAKE.

2) Task 3.2: In this last task, we transmit items of agents $t_{aj}(G(i))$, $1 \leq j \leq k$ at their corresponding times, one by one. The details of this procedure are in the Algorithm 6: $Broadcast\_to\_final\_destinations$.

This task takes $2(|M_{a1}| + |M_{a2}| + .. + |M_{ak}|) \leq 2\frac{T}{k}MAX(M_i)$ time slots to transmit/receive items (since this is the maximum number of items it holds as agent thing node), and all the thing nodes remain awake for at most $2\frac{T}{k}MAX(M_i)$ time slots.

The number of time slots for completion and the maximum awake time slots are shown in Table I for all protocol execution steps. Therefore, all steps takes overall $k + k^2 + \frac{T}{k} + 4\frac{T}{k}MAX(M_i)$ time slots and all the thing nodes remain awake for $k + k^2 + \frac{T}{k} + 4\frac{T}{k}MAX(M_i)$ time slots.

Algorithm 6 $Broadcast\_to\_final\_destinations(t_i(G(i)))$

```
{do in parallel for each $G(i)$ on channel $C(i)$}
INPUT: set $Lp = \{M_{a1}, M_{a2}, .., M_{ak}\}$ in $G(i)$
                    {$Lp$ is calculated in $Algorithm$ 5}
  if $Time\_to\_Broadcast(t_{an}(G(i)))$
  AND $t_j(G(i))$ is agent $t_{ap}(G(i))$ then
    for $s \leftarrow 1, s \leq M_{an}$ do
      agent $t_{ap}(G(i))$ sends Items to final thing node
    end for
  else
    if Thing node $t_j(G(i)$ is final destination then
      Thing node $t_j(G(i)$ copies items in its local memory
    else
      Thing node $tj(G(i)$ drop items
    end if
  end if
  {Definition of Function: $Time\_to\_Broadcast()$}
  Function $Time\_to\_Broadcast$: $t_j(G(i))$,$M_{an}$
  $d \leftarrow 0$
  for $h \leftarrow 1, h < M_{an}, h++$ do
    $d \leftarrow d + M_{an}$ {$d = |M_{a1}| + |M_{a2}| + .. + |M_{ak}|$}
  end for
  return $(d)$
```

VII. CONCLUSION

We have presented a simple fault-tolerant and energy-efficient reservation-based DAMA protocol for permutation routing protocol for single-hop wireless network of things. Our protocol performs well when the number channels satisfy the condition $k \leq \sqrt{T}$. However, there is a check in our protocol for this condition if it is violated, we do a reset of the entire network. This protocol allows a parallel broadcasting using several communication channels. Therefore, the proposed protocol allows in a real application to be fault-tolerant, energy-efficient and works without conflicts or collisions on the communication channels.

We plan to simulate this protocol and are convinced that it will work well if the previous constraints are met. The number of emissions from each node should be much lower than the theoretical results found.

REFERENCES

[1] R. Dechter , L. Kleinrock: Broadcast Communication and Distributed Algorithms.IEEE Trans. Computers, vol. 35, pp. 210-219, 1986.
[2] I. Chlamtac , S. Kutten: Tree-Based Broadcasting in Multihop Radio Networks.IEEE Trans. Computers, vol. 36, pp. 1209-1223, 1987.
[3] W. C. Fifer , F. J. Bruno: Low cost packet radio. Proc. IEEE, Vol. 75, pp. 33-42, 1987.
[4] O. Iova, F. Theoleyre , T. Noel: Using Multiparent Routing in RPL to Increase the Stability and the Lifetime of the Network.Elsevier Ad Hoc Networks, vol. 19, pp. 45-62, June 2015.
[5] A. Radhakrishnan , M. L. Madhav: A Survey on Efficient Broadcast Protocol for the Internet of Things. IJECS, Vol. 5, pp. 18838-18842, 2016.
[6] K. Sivalingam, M.B. Srivastava, P. Agrawal: Low Power Link and Access Protocols for Wireless Multimedia Networks. Proc. IEEE Vehicular Technology Conf., 1997.
[7] A. Datta : A Fault-Tolerant Protocol for Energy-Efficient Permutation Routing in Wireless Networks. IEEE Trans. Computers, Vol. 54, pp. 1409-1421, 2005.
[8] D. Bertsekas , R. Gallager: Data Networks, 2nd Edition, Prentice Hall, (1992). ISBN: 0-13-200916-1
[9] J. Yick, B. Mukherjee , D. Ghosal: Wireless sensor network survey. Computer Networks, vol. 52, no. 12, pp. 2292-2330, 2008.
[10] J. Gubbi, K. Krishnakumar, R. Buyya, M. Palaniswami: Iternet of Things (IoT): A vision, architectural elements, and future directions. Journal of Future Generation Computer Systems, Vol. 29, Issue 7, pp. 1645-1660, 2013.
[11] A. Datta , A. Y. Zomaya: New energy-efficient permutation routing protocol for single-hop radio networks. Proc. 8th International Computing and Combinatorics Conference (COCOON 02), LNCS Vol. 2387, pp. 249-258, (2002).
[12] H. Lakhlef, M. Raynal , J. Bourgeois: Efficient Broadcast Protocol for the Internet of Things. 30th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 998-1005, 2016.
[13] H. Lakhlef, B. Bouabdallah, M. Raynal , J. Bourgeois: Agent-based Broadcast Protocols for Wireless Heterogeneous Node Networks. Computer Communications, Vol. 115, pp. 51-63, (2018).
[14] K. Nakano , S. Olariu: Randomized initialization protocols for radio networks. IEEE Trans. Parallel and Distributed Systems, Vol. 11, pp. 749-759, 2000.
[15] K. Nakano, S. Olariu , J. L. Schwing: Broadcast-efficient protocols for mobile radio networks. IEEE Trans. Parallel and Distributed Systems, Vol. 10, pp. 1276-1289, 1999.
[16] K. Nakano, S. Olariu , A.Y. Zomaya: Energy-Efficient Permutation Routing in Radio Networks. IEEE Trans. Parallel and Distributed Systems, vol. 12, no. 6, pp. 544-557, 2001.
[17] R. A. Powers: Batteries for low-power electronics. Proc. IEEE, Vol. 83, pp. 687-693, 1995.
[18] M. Fine , F. A. Tobagi: Demand assignment multiple access schemes in broadcast bus local area networks. IEEE Trans. Computers, Vol. 33, pp. 1130-1159, (1984).