Agent-based Broadcast Protocols for Wireless Heterogeneous Node Networks

Hicham Lakhlef[a,*], Abdelmadjid Bouabdallah[a], Michel Raynal[b], Julien Bourgeois[c]

[a] *Sorbonne Universites, University of Technology of Compiegne, France*
[b] *IRISA, Université de Rennes, France*
[c] *UFC/FEMTO-ST, UMR CNRS 6174, 1 cours Leprince-Ringuet, 25201 Montbeliard, France*

Abstract

Internet of Things (IoT) is a wireless network composed of a variety of heterogeneous objects such as Connected Wearable Devices (sensors, smartwatches, smartphones, PDAs ...), Connected Cars, Connected Homes,...etc. These things use generally wireless communication to interact and cooperate with each other to reach common goals. $IoT(T, n)$ is a network of things composed of T things with n items (packets) distributed randomly on it. The aim of the permutation routing is to route to each thing, its items, so it can accomplish its task. In this paper, we propose two agent-based broadcast protocols for mobile IoT, using a limited number of communication channels. The main idea is to partition the things into groups where an agent in each group manages a group of things. This partitioning is based on the memory capacities for these heterogeneous nodes. The first protocol uses a few communication channels to perform a parallel broadcasting and requires $O(\frac{n}{k})$ memory space, where k is the number of communication channels. The second protocol uses an optimal complexity of memory space for each thing to achieve the permutation routing with a parallel broadcasting using less number of channels. We give an estimation of the upper and lower bounds of the number of broadcast rounds in the worst case and we discuss experimental results.

Keywords: Internet of thing; parallel broadcasting; communication protocols; permutation routing; collision-free; energy-efficiency

1. Introduction

The Internet of things (IoT) consists of a great number of heterogeneous nodes such as Connected Wearable Devices (sensors, MEMS, robots, smartwatchs, smartphones, PDA ...), Connected Smart Cars, Connected Smart Homes, Connected Smart Cities, and the Industrial Internet. These things are equipped with data processing and communication capabilities which give them the ability of sensing, computation, and wireless communications [1, 2, 3, 4]. IoT is an attractive research subject that has started to receive growing attention from the research and engineering communities in recent years. The nodes in IoT may be mobile or static, deployed in ad hoc manner in area of interest. These things are useful in a wide range of applications of our every-day life. Such applications include smart energy, smart health, distributed intelligent MEMS, smart buildings, smart transport, smart industry, smart city, facilitating/conducting urban search and rescue, tasks in unattended and rough environments etc.,[5, 6, 7, 8, 9]. Roughly speaking, IoT is making our daily life easier and smarter.

The Internet of things generally employs large number of distributed heterogeneous things, which may be miniaturized devices that cooperate and collaborate with each other using wireless communication to achieve common goals and objectives. Each thing has an onboard radio that can be used to receive messages from its neighbors and to send the information to them. That is, each thing needs to receive information available in the local memories of other things using routing protocols. We refer the reader to Fig. 1, depicting a 15-things in IoT. Such technological development has encouraged practitioners to envision aggregating the limited capabilities of the individual things in a large scale network that may operate unattended, [1, 5, 10, 11].

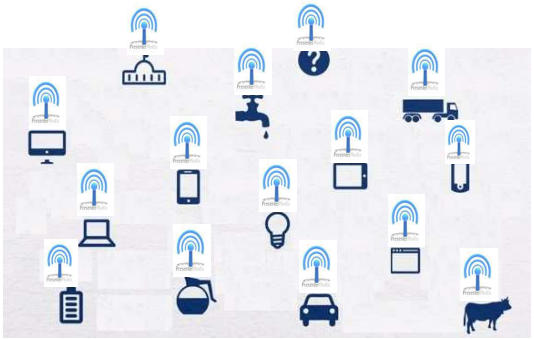


Figure 1: Example of an Internet of Things network, things cooperate and collaborate with each others to achieve a common goal

As said before, IoT will occupy a prominent place in our day-to-day life. However, the design of protocols to

*Corresponding author
Email addresses: hlakhlef@utc.fr (H. Lakhlef), bouabdal@utc.fr(A. Bouabdallah), raynal@irisa.fr (M. Raynal), jbourgeois@femto-st.fr(J. Bourgeois)

control them in order to achieve a common goal is far from being a simple task. Indeed, due to the resources limitation, a solution for an application in Internet of things should take into account the restrained capabilities (limited battery power, processing power and memory storage) of these heterogeneous devices by using as little memory and energy as possible whilst maximize the lifetime of the network [12, 13, 14]. Furthermore, the number of radio channels are limited. In addition, transmissions through wireless channels can suffer errors due to both channel and interference conditions, [15, 16, 17, 19]. It is well known that, in real applications communication channels are subject to channel noise, fault and thus errors may be introduced during transmission from the source to a receiver. Besides, in some cases some channels are damaged and are no longer used. These facts illuminate the need and the importance to have efficient protocols that perform with a limited number of communication channels, and explain the large number of research devoted to this complex field.

In this paper, we are interested in the problem of routing in single-hop IoT. Single-hop IoT is the key solution to handle routing in multi-hop networks. As preliminary step, the network is partitioned into single-hop networks called $Clusters$. Each cluster contains a set of things managed by a *Cluster-head*. Broadcasting is performed locally in each cluster and getaways are used for the communication between cluster heads to send the items to their final destinations [17, 18, 25].

The problem of permutation routing is involved when each node in the network needs to receive information from other nodes. More precisely, each node cannot decide or cannot do its task, because the information that allow it to know what to do or decide are localized at the memory spaces of other nodes. Each node has to send what it has in its local memory to allow its neighbors progress. Thus, the nodes permute their information between them to solve the problem, while minimizing the total number of retransmissions [26, 27, 28, 29].

The permutation routing problem has been subject to several researches. It was studied under the condition that each node has only one item (information) to be transmitted to another destination node. Afterward, in an effort to generalize the definition, and having regard to needs, the problem has been studied under the condition that each node i has x items, that must be transmitted to y_i nodes where each node receives exactly x items (an item has one destination). This generalization made the problem more complex to solve in a restrained resource networks. However, the real generalization of the definition of this interesting problem was not studied until the arrival of IoT. In IoT, the problem is more complicated because the nodes are heterogeneous (in term of memory capacity, energy capacity, computing capacity etc.) In IoT, the problem is defined as follows: each thing t_i stores in its local memory M_i items, each t_i must receive M_i items available in the local memories of f_i things. Thing t_i knows the destinations of all the M_i items that it holds. However, thing t_i does not know from which stations it will receive its items. The aim of the permutation routing problem is to route the items in such a way that each node has all its own items at the end. These items allow to a thing to know its task. For example, in Fig. 1, the thing lamp needs to receive information (messages) from the car and the pad, to decide if it has to be bright or not. As stated in [16, 17, 19], the permutation routing problem is one of the most important issue in the field of computer sciences.

The permutation routing can also serve as solution for privacy and security. Indeed, to prevent the attacker to know the task for a thing (soldier), we do not give to a thing all its items during the deployment, but we permute the items on a set of things. When the things are all in a safe environment, they do the permutation routing, to let each thing receive all its items from the other things that are in the same network.

In this work, we propose efficient agent-based routing solutions for the internet of things where the number of channels and the memory usage are as few as possible.

Related Works.

As said before, IoT is going to offer a large number of applications in various environments for improving the quality of our lives. Routing in IoT is a topic that has attracting the research community in last years [20]. In [21], the authors propose 6LoWPAN, a major routing protocol for IoT systems. It has been defined by the Engineering Task Force (IETF) to route data through the Internet among non IP sensors. However, 6LoWPAN is used for networks with high processing capabilities. For that, (RPL) Protocol for Low Power and Lossy Networks [22] has been designed for resources constrained devices. RPL is a distance vector routing protocol that is based on IPV6. It builds a Destination Oriented Directed Acyclic Graph (DODAG). Many metrics may be used to construct a DODAG: the Expected Number of Transmissions (ETX) [23], the remaining energy of the devices. Energy-Efficient Probabilistic Routing (EEPR) [24] is an alternative solution for routing in an IoT environment. It is based on the same idea of AODV but the transmission of a RREQ packet follows a certain forwarding probability that depends on the residual energy and the ETX metric. The authors in [20] proposed a cross-layer routing protocol to meet the performance parameters of IoT applications such as the minimum date delivery rate and maximum packet delay. All these papers do not take into account the routing of a huge number of packets in the same time using a limited number of communications channels and a very limited memory capacity of some nodes.

The number of studies specifically targeted to the permutation routing problem in radio networks has grown significantly. The common goal of all these researches is to propose efficient broadcasting protocols that use few communication channels.

It is shown in [17, 15] that the permutation routing of n items distributed on a wireless radio network of p stations and k channels with $k < p$, can be carried out efficiently

if $k \leq \sqrt{\frac{p}{2}}$ as the number k of available radio channels is significantly smaller than the number of stations p. A more energy-efficient permutation routing that handles the dynamicity of the networks in the sense *sleep/wakeup* appeared in [16]. In [30], the authors solve the problem using a concurrent broadcast protocol on multiple channels. The author in [31], derived a fault tolerant permutation routing protocol of n items distributed on mobile ad-hoc network of p stations and k channels $MANET(n, p, k)$ for short. He also assumed that $k \leq \sqrt{\frac{p}{2}}$ and in the presence of faulty nodes some data items are lost. These algorithms require two phases. These two phases approach divides the stations into groups. During the first phase, each packet is routed to a station in the group containing the destination station. In the second phase, stations in each group route packets to their final destination. In [19], the authors present for single-hop radio networks a one phase algorithm that routes the packets directly to their destinations. Its expected run time is nearly $\frac{n}{k}$ and assumes that $k \leq \sqrt{p}$. In [32], the authors presented a randomized algorithm for the permutation routing that takes $\frac{3n}{k} + O(\frac{n}{k})$ broadcast rounds with high probability.

In [33], the authors presented a fault tolerant protocol which avoids the loss of items. The existence of these faulty nodes can significantly affect the packets delivery rate. If a faulty node, participating in a permutation routing operations, drops packets, all these packets will be lost. Hence, the goal of a fault tolerant permutation routing in [33], is to provide certain level of packet delivery guarantee in spite of the presence of faulty stations. The goal of the solution in [34] is to route the overall items to their destinations while consuming as little energy as possible. The authors showed that the permutation routing problem of n packets on a $RadioNetwork(p, k)$ of p stations and k channels can be solved in $\frac{2n}{k} + (\frac{p}{k})^2 + p + 2k^2$ slots. In [35], the authors propose an optimal permutation routing on mesh networks where $\frac{n}{p} = 1$. Another approach as an application of an initialization algorithm appeared in [36]. All these approaches assume that the nodes of the network are homogeneous. The solution in [37] presents a randomized algorithm for the same problem in multi-hop network with high probability. These solutions are designed for radio or sensor networks and each node within these protocols stores exactly one item or $\frac{n}{p}$ items.

Some solutions for the permutation routing in wireless node networks use clustering to divide the network into groups of single-hop networks where the cluster head routes the items to another cluster. The protocol in [38] is designed for multi-hop homogenous sensors. It needs $(\frac{n}{p})(HUBmax)(k+1) + O(HUBmax) + k^2 + k$ broadcast rounds, where n is the number of the data items stored in the network, p is the number of sensors, $HUBmax$ is the number of sensors in the clique of maximum size and k is the number of cliques after the first clustering. The drawback of this protocol is the fact that there is a high probability of collision and conflict on the communication channels, because it has not been shown a mechanism to manage the broadcasting. In [39], it is presented a secure permutation routing protocol in multi-hop networks. It uses clustering and secures the procedure of clustering and routing.

Contributions. We consider a wireless network of heterogeneous things with n items (packets) and T things, $IoT(T, n)$ for short. The n items are distributed on the T things, so each thing t has in its local memory M_t items. In this paper, we propose two, agent-based, distributed and parallel protocols for the permutation routing for Internet of Things. We partition the things into groups, where in each group there is a set of agents. The role of the agents is to manage groups of things and also manage the broadcasting on the communication channels so to provide protocols that run without collision or conflict at the communication channels. Both protocols aim to use maximally the communication channels available.
The aim of the first protocol is to use a few communication channels and perform broadcasting in parallel to optimize the number of broadcast rounds. The main idea of this protocol is to partition the things into k groups, where k is the number of communication channels that allows using $O(\frac{n}{k})$ of memory space. In this protocol the grouping is based on the number k. The aim of the second protocol is to use a minimum amount of memory for each thing, where the broadcasting in parallel is possible. In this protocol a thing t that stores M_t items uses $O(M_t)$ of memory space. Contrary to the first protocol, in this one the partitioning is based on the memory capacities of the nodes in the groups. Therefore, the number of communication channels used is smaller compared to the first one. The proposed permutation routing protocols are distributed, where nodes make autonomous decisions without any centralized control. We give an estimation of the upper bound of the number of broadcast rounds in the worst case and experiments results. Our solutions are the first to give efficient and collision-free solutions for the permutation routing problems for Internet of Things.

Outline of the paper. The paper is made up of 5 sections. The rest of sections is organized as follows: Section 2 presents the model of the network and some definitions. Section 3 presents and discusses the proposed protocols. Section 4 details the simulation results. Finally, our conclusions and suggestions for the future works are given in section 5.

2. Model and definitions

The network considered is simple-hop (the pairs can communicate directly). IoT includes large numbers of mobile thing nodes. The things communicate with each other using bidirectional links. The computation and communication capabilities are different for all thing nodes. Furthermore, the things have different memory capacities. IoT (T, n) is a thing network with n items (information or packets) and T things. Fig. 2 depicts IoT $(15, 56)$. In IoT (T, n), each thing t_i has in its local memory M_i items.

Each item has a unique destination thing. The time is divided into slots and all packet transmissions take place at slot boundaries in IoT (T, n). The permutation routing problem is to route the items in such a way that each thing t_i receives its M_i items. Fig. 2 (a) presents an example where the items are distributed in the network and each thing holds items that must be routed to their destinations. Fig. 2 (b) presents the network after the permutation routing, so each node has its information and it can proceed to perform its task.

As in [16, 17, 19], the term *broadcast* in this paper refers to *one-to-many* and to *point-to-point* transmission. The set of all broadcast operations that take place in the same time slot is referred to as a *broadcast round*.

Let $LT = \{t_1, t_2, ..., t_T\}$ be the list of T things in $IoT(T, n)$ and let $L = \{M_1, M_2, ..., M_T\}$ be the list of the memory spaces that hold the items of the things in the network. With, for each $1 \leq i \leq T$, M_i is the memory space (number of items) of thing t_i. We note:
-$MIN(M_i)$, the lowest memory space for items that a thing has in the network, i with $\forall j \neq i$ then $M_j > M_i$. If $M_j = M_i$, then $i < j$
-$MAX(M_i)$, the largest memory space for items that a thing has in the network, i with $\forall j \neq i$, then $M_j < M_i$. If $M_j = M_i$, then $i < j$.

In the presentation of the proposed protocols, we work mainly on the memory spaces for items $M_1, M_2, ..., M_T$ and when an action (add into a group, delete from a group...) is applied on a given memory space M_i this means that the action is applied on t_i, assuming that there is a function $ID(M_i)$ that gives the thing t_i.

3. Proposed Protocols

In this section we present and discuss our proposed protocols. This section is divided into two subsections. In the first one, we present an efficient protocol that uses $O(\frac{n}{k})$ of memory space for each thing and in the second subsection, we present a protocol that uses $O(M_i)$ for each thing t_i. We discuss the advantages of each one.

3.1. Protocol with $O(\frac{n}{k})$ memory on $IoT(T, n)$

The aim of this section is to show that the permutation routing on an $IoT(T, n, k)$ of T things, n items (information) and k channels, where each thing has a memory space of $O(\frac{n}{k})$ is possible if:

$$k \leq \left\lfloor \frac{\sum_{i=1}^{T}(M_i)}{MAX(M_i)} \right\rfloor . \quad (1)$$

The procedure is to divide the channels on the T things, so to perform broadcasting in parallel and to use optimally each channel. Clearly, the aim is to distribute the broadcasting on the channels. Contrary to the related works, we can see clearly that this grouping will not depend on the number of things because the things store different numbers of items. Therefore, this grouping depends on the list L of the memory spaces of things that hold the items. The procedure is to divide the things into k groups. In each group there are k agents. The role of each agent in a given group is to receive the elements that belong to a given group (one group of the k groups). In the first step each node sends its items to the agents and in the second steps the agents broadcast the items to their final destinations. Note that each agent in each group may be the destination of all things in its group, because the items in group of things $G(1)$ may have all destinations in group of things $G(2)$. Based on this, we can observe that an agent should have a memory capacity to store a set of elements of a group. Therefore its memory space must be equal to a size of a group. To satisfy this we can choose:

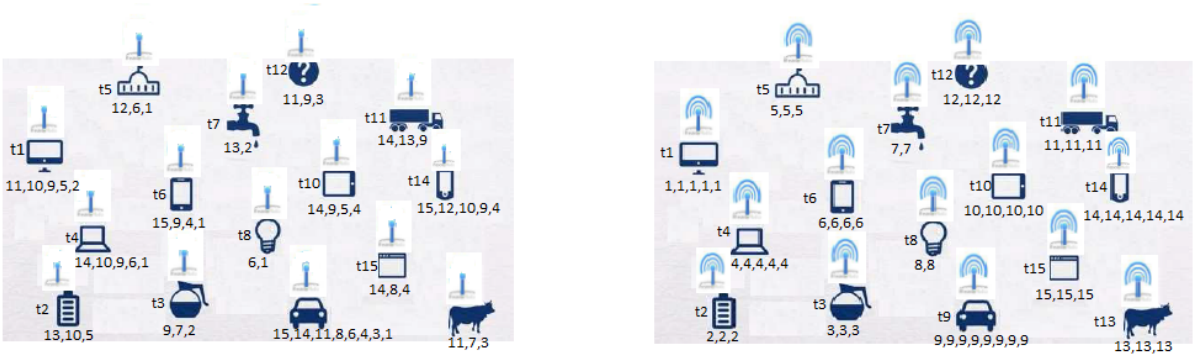
$$k \leq \left\lfloor \sqrt{\sum_{i=1}^{T}(M_i)} \right\rfloor . \quad (2)$$

This condition depends on the memory capacities of things. With this condition an agent will use in the worst case $O(\frac{n}{k})$ of memory space because in each group the overall number of items will be $\frac{\lfloor \sqrt{\sum_{i=1}^{T}(M_i)} \rfloor}{k} = \frac{n}{k}$ items. However, the problem occurs in grouping procedure. Given that the things store different number of items, thus, any grouping procedure based on the number of things will not give always groups with equal sizes. Even, if the grouping procedure is based on the memory spaces it will not give always groups with equal sizes, because it is a matter of impossibility or NP-complete problem. The inequation (1) allows to have k groups and in each group there is at most $O(\frac{n}{k})$ items.

This protocol is composed of three steps: *Grouping*, *Broadcast to agents* and *Broadcast to the final destinations*. Next we give the details of each step.

3.1.1. ***Grouping***

The first step is to partition the network into k groups, $G(1), G(2), ..., G(k)$, such that $G(j)$, $0 < j \leq k$. In each group $G(j)$ will be δ_j things. A size, $s(j)$, of a group is defined as the sum of all M_i, $i \in \{1, 2, ..., T\}$, that a group $G(j)$, $0 < j \leq k$, contains. We use *Grouping Algorithm* presented in Fig. 3 to apply the grouping of the things in the network. This algorithm takes in input a list of memory spaces $L = \{M_1, M_2, ..., M_T\}$ which represents the memory spaces that contain the items and an integer k which represents the number of channels and the number of groups. It outputs the groups $G(1), G(2), ..., G(k)$. Each group contains a set of things.
The algorithm, firstly, puts in $G(1)$, $MAX(M_i)$ of L, in $G(2)$, $MAX(M_i)$ of $L - G(1)$, and so on until it puts in $G(k)$, $MAX(M_i)$ of $L - \{G(1) \cup .... \cup G(k-1)\}$. After, at each step it adds the maximum in the new list Ln, that does not contain the elements already added into the k groups, to the group $G(j)$ that has the current minimum



(a): A network of internet of things before the permutation routing, at least one node does not have all its items

(b): A network of internet of things after the permutation routing, every thing node has its items

Figure 2: An example of IoT before and after the permutation routing

$s(j)$. By applying this algorithm, each group will have at the end a size $s(j)$.

Definition 3.1. *Let a set of T positive integers be in $L = \{M_1, M_2, ..., M_T\}$ and let a set of k positive integers be $K = \{m_1, m_2, ..., m_k\}$, with:*

- $k < T$,
- $K \subset L$,
- $m_a \geq m_b$, $a < b$, $0 < a, b \leq k$,
- *for all* $0 < i \leq k$, $m_i \geq M_j \in L - K$, $0 < j \leq T$ *and*
- $k \leq \frac{\sum_{i=1}^{T}(M_i)}{m_1}$.

By applying Grouping Algorithm in Fig. 3 on L and k, we name a full iteration when an or several integers M_i, $1 \leq i \leq T$ from a set $H \in L$ are added into each m_j, $1 \leq j \leq k$.

$IT_{L\|K}(\gamma)$ is the full iteration number γ by applying the grouping algorithm on L and k.

Lemma 3.1. *By applying the grouping algorithm, whatever the values M_i, $i \in \{1, 2, ..., T\}$ and k satisfies inequation (1), then the minimum size $s(j)$ is always:*

$$s(j) \geq \frac{MAX(M_i) \times (k-1)}{2k-3}. \quad (3)$$

Proof $IT_{L\|K}(1)$ is the maximal full iteration that allows having the minimum of $s(j)$ because if there is $IT_{L\|K}(\gamma)$, with $\gamma > 1$ this means that to all m_i was added one or several M_i, including $MAX(M_i)$. Therefore, this means that there was in some previous step all $m_i + C_i \geq MAX(M_i)$, where C_i is the sum of all values from L added to m_i. Consequently, $IT_{L\|K}(\gamma), \gamma > 1$ will not give the minimum $s(j)$. To apply $IT_{L\|K}(1)$, the sum $\sum_{i=1}^{T}(M_i)$ must be the minimum as possible and satisfies inequation (1). This sum is $\sum_{i=1}^{T}(M_i) = MAX(M_i) \times k$ because $(MAX(M_i) \times k) - \epsilon$ does not satisfy inequation (1). $\forall m_i$, no integer will be added to m_1 because $\sum_{i=2}^{T}(M_i) = MAX(M_i)(k-1)$. Therefore, by applying the grouping algorithm, the sum $\sum_{i=1}^{T}(M_i) = MAX(M_i) \times k$ will give the minimum $s(j)$.

To have the minimum we should find the k-tuple $(m_1, m_2, m_3, ..., m_k)$ where m_2 is the minimum, that if chosen, no number from $L-K$ will be added to it. That is, the k-tuple allows, following the grouping algorithm to add the $L-K$ to $m_3, ..., m_k$ without adding any value to m_2. Note that m_1 is the maximum and it is invariable, so it is out of computation. To let the difference maximal between m_2 and $m_3, ..., m_k$ after adding the sum, say X, to $m_3, ..., m_k$, we must have $m_2 = m_3 = ... = m_k$. This means that, the sum X, to be add to $m_3, ..., m_k$ divided by m_2 must be inferior than or equals to $k-2$. Because if $\frac{X}{m_2} > k-2$, following the grouping algorithm a positive value will be added to m_2. To assemble our conclusions, the condition to not add any element to m_2 is:

1. **INPUT** : A set L and k
2. **OUTPUT** : A list of groups,
3. $LG \leftarrow \{G(1), G(2), ..., G(k)\}$
4. **begin**
5. $X \leftarrow L; b \leftarrow 1;$
6. **while** $(b \leq k)$ **do**
7. $G(b) \leftarrow MAX(X);$
8. $X \leftarrow X - G(b-1);$
9. $b \leftarrow b + 1$
10. **end while;**
11. $Ln \leftarrow L - \{G(1), ..., G(k-1), G(k)\}$;
12. $LG \leftarrow \{G(1), G(2), ..., G(k)\}$;
13. **while** $(Ln \neq \oslash)$ **do**
14. $MIN(LG) \leftarrow MIN(LG) \cup MAX(Ln);$
15. $Ln \leftarrow Ln - \{MAX(Ln)\}$
16. **end while;**
17. Return (LG)
18. **end**

Figure 3: Grouping Algorithm

$$\frac{X}{m_2} \leq (k-2). \quad (4)$$

As $X = (m_1 \times k) - (m_1 - (k-1)m_2)$, we can write:

$$\frac{(m_1 \times k) - (m_1 - (k-1)m_2)}{m_2} \leq (k-2). \quad (5)$$

(5) $\Leftrightarrow \frac{(MAX(M_i)\times(k-1))}{2k-3} \leq m_2$. □

Lemma 3.1 does not give always integer values. To deal with each case to have the exact size we give the following equations that can be deduced from inequation (3):

if k is odd:

$$s(j) \geq MAX(M_i) - (\frac{MAX(M_i)}{k} \times \frac{k-1}{2}), \quad if\ MAX(M_i)\textbf{mod}\ k = 0, \quad (6)$$

or

$$s(j) \geq MAX(M_i) - (\left\lfloor \frac{MAX(M_i)}{k} \right\rfloor \times \frac{k-1}{2} + \frac{k-1}{2}), \quad if\ MAX(M_i)\textbf{mod}\ k \geq \frac{k-1}{2}, \quad (7)$$

or

$$s(j) \geq MAX(M_i) - (\left\lfloor \frac{MAX(M_i)}{k} \right\rfloor \times \frac{k-1}{2}), \quad if\ MAX(M_i)\textbf{mod}\ k < \frac{k-1}{2}. \quad (8)$$

if k is even:

$$s(j) \geq MAX(M_i) - (\frac{MAX(M_i)}{k+1} \times \frac{k}{2}), \quad if\ MAX(M_i)\textbf{mod}\ (k+1) = 0, \quad (9)$$

or

$$s(j) \geq MAX(M_i) - (\left\lfloor \frac{MAX(M_i)}{k+1} \right\rfloor \times \frac{k}{2}), \quad if\ MAX(M_i)\textbf{mod}\ (k+1) \leq \frac{k}{2}, \quad (10)$$

or

$$s(j) \geq MAX(M_i) - (\left\lfloor \frac{MAX(M_i)}{k+1} \right\rfloor \times \frac{k}{2}) + 1), \quad if\ MAX(M_i)\textbf{mod}\ (k+1) > \frac{k}{2}. \quad (11)$$

Lemma 3.2. *Let a set of positive integers be in $L = \{M_1, M_2, ..., M_T\}$ and let a set of positive integers be in $K = \{m_1, m_2, ..., m_k\}$ as in Definition 3.1. And let mk be the maximum value of k that can be chosen with respect to inequation (1).*

Then, if we apply the grouping algorithm on L, with mk, the largest group, lg, will be with a size of $S(lg) \leq 2MAX(M_i) - 1$.

Proof Given that, from inequation (1), $\frac{\sum_{i=1}^{T}(M_i)}{m_1} \geq k$. If k chosen is the maximum as possible, $mk = \left\lfloor \frac{\sum_{i=1}^{T}(M_i)}{m_1} \right\rfloor$. To let mk maximum we may add until $u = MAX(M_i) - 1$ to $\sum_{i=1}^{T}(M_i)$. If $u > MAX(M_i)-1$, the value $\frac{\sum_{i=1}^{n}(M_i)+u}{MAX(M_i)} \geq (k+1)$, therefore mk will not be the maximum, which represents a contradiction. Therefore, we conclude until now that: to let mk maximum we may add until $u = MAX(M_i) - 1$ to $\sum_{i=1}^{T}(M_i)$.
We obtain the largest group if $MAX(M_i) = m_1 = m_2 = ..., = m_k$ because, following the grouping algorithm, the value u will be added to $MIN(M_i)$. Therefore, the largest group will be $MAX(M_i)+MAX(M_i)-1 = 2MAX(M_i)-1$. Consequently, for different values of $m_1, m_2, ...,$ and m_k, $S(lg) \leq 2MAX(M_i) - 1$. □

Lemma 3.3. *Let a set of positive integers be $L = \{M_1, M_2, ..., M_m\}$ and a set of positive integers be $K = \{m_1, m_2, ..., m_k\}$ as in Definition 3.1*

Then, if we apply the grouping algorithm on L, with $k < mk$ the largest group, lg, will be with a size of:
*Case 1: $S(lg) \leq (\frac{mk}{k}+1)\times MAX(M_i)-1$, if mk **mod** $k = 0$, or*
*Case 2: $S(lg) \leq (\lfloor \frac{mk}{k} \rfloor +1) \times MAX(M_i)$, if mk **mod** $k \neq 0$.*

Proof We first proof the Case 1: for $mk = k$, the size is $(1+1) \times MAX(M_i) - 1$. This is true, from the previous lemma. From the previous lemma, to let mk maximum we may add until $u = MAX(M_i) - 1$ to $\sum_{i=1}^{m}(M_i)$. Let $G(j)$, $1 \leq j \leq k$, initialized to 0. Each $G(j)$ will contain one m_i and the values to be added to m_i. If $mk\,\textbf{mod}\,k = 0$, to each $G(j)$ will be added $(\frac{mk}{k})\times MAX(M_i)$ except int_k that will be added to it $(\frac{mk}{k}) \times MAX(M_i) + MAX(M_i) - 1$. Therefore, in this case the largest group will be of size $(\frac{mk}{k})\times MAX(M_i)+MAX(M_i)-1 = (\frac{mk}{k}+1)\times MAX(M_i)-1$. Consequently, for different values of $m_1, m_2, ...,$ and m_k, $S(lg) \leq (\frac{mk}{k}+1) \times MAX(M_i) - 1$, if $mk\,\textbf{mod}\,k = 0$.

For the Case 2: this second case can be observed directly from Case 1. In this case, to each $G(j)$ will be added $\frac{mk-mk\textbf{mod}k}{k} \times MAX(M_i)$, as mk **mod** $k \neq 0$. And remains $(mk-k) \times MAX(M_i)$. Given that $(mk-k) < k$. So to each $(mk-k)$ of the group $G(j)$, $1 \leq j \leq k$, will be added $MAX(M_i)$. Therefore, in this case the largest size is $\frac{mk-mk\textbf{mod}k}{k} \times MAX(M_i) + MAX(M_i) = (\lfloor(\frac{mk}{k})\rfloor + 1) \times MAX(M_i)$. Consequently, for different values of $m_1, m_2, ...,$ and m_k, $S(lg) \leq (\lfloor(\frac{mk}{k})\rfloor + 1) \times MAX(M_i)$, if mk **mod** $k \neq 0$.

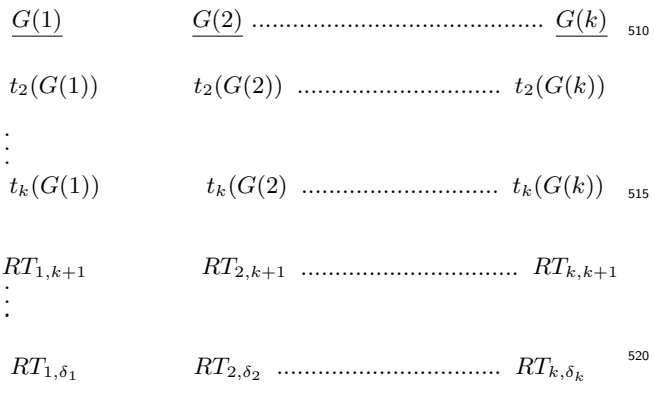


Figure 4: A model for the grouping step

Example

We run the grouping algorithm on the network of Figure 2 (b), assuming the number of channels is 2 ($k = 2$). We have $X = L = \{M_1, M_2, ..., M_{15}\}$ (line 5). As $k = 2$ the algorithm will create two groups, $G(1)$ and $G(2)$ (line 3).

As $k = 2$, the algorithm first adds into $G(1)$ the thing $MAX(X)$, which is the thing t_9 that has $M_9 = 8$ (line 7). After it adds t_{14} into $G(2)$ (line 7). We have now $ln = L - \{M_1, M_2\}$ (line 11). After, in instruction 14 the algorithm adds into $G(2)$ the max in Ln so $G(2) = \{t_{14}, t_1\}$. So at line 15 $ln = L - \{t_{14}, t_1, t_9\}$. After, in line 14 we add t_4 into group $G(1)$ because $|G(1)| < |G(2)|$. So $G(1) = \{t9, t4\}$. After we add t_{10} into group $G(2)$ because $|G(2)| < |G(1)|$ (line 14). So $G(2) = \{t_{14}, t_1, t_{10}\}$. After we add t_6 into group $G(1)$ because $|G(1)| < |G(2)|$. So $G(1) = \{t_9, t_4, t_6\}$. After we add t_2 into group $G(2)$ because $G(2) < G(1)$. So $G(2) = \{t_{14}, t_1, t_{10}, t_2\}$. After we add t_3 into group $G(1)$ because $|G(1)| < |G(2)|$. So $G(1) = \{t_9, t_4, t_6, t_3\}$. After we add t_5 into group $G(2)$ because $G(2) < G(1)$. So $G(2) = \{t_{14}, t_1, t_{10}, t_2, t_5\}$. After we add t_{11} into group $G(1)$ because $|G(1)| < |G(2)|$ (line 14). So $G(1) = \{t_9, t_4, t_6, t_3, t_{11}\}$. After we add t_{12} into group $G(2)$ because $|G(2)| < |G(1)|$ (line 14). So $G(2) = \{t_{14}, t_1, t_{10}, t_2, t_5, t_{12}\}$. After we add t_{13} into group $G(1)$ because $|G(1)| < |G(2)|$ (line 14). So $G(1) = \{t_9, t_4, t_6, t_3, t_{11}, t_{13}\}$. After we add t_{15} into group $G(2)$ because $|G(2)| < |G(1)|$ (line 14).
So $G(2) = \{t_{14}, t_1, t_{10}, t_2, t_5, t_{12}, t_{15}\}$. After we add t_7 into group $G(1)$ because $|G(1)| < |G(2)|$ (line 14). So $G(1) = \{t_9, t_4, t_6, t_3, t_{11}, t_{13}, t_7\}$. After we add t_8 into group $G(2)$ because $|G(2)| < |G(1)|$ (line 14). So $G(2) = \{t_{14}, t_1, t_{10}, t_2, t_5, t_{12}, t_{15}, t_8\}$. At this point the list ln is empty and the algorithm ends by giving two groups : $|G(1)| = 28$, $|G(2)| = 29$.

3.1.2. ***Broadcast to agents***

Once the grouping is finished and the communication channels are assigned to groups. The next step is identifying the agents in each group. The role of each agent is to receive in its local memory all elements that belong to a (one) group, (i.e. one group of k groups, $G(1)...G(k)$). The probability that a given thing t_{i1} having a memory M_{i1} has already items belongs to a given group $G(j)$, $1 \leq j \leq k$, is $\frac{M_{i1}}{s(j)}$. It is greater than the probability that another thing t_{i2} having M_{i2}, with $M_{i2} < M_{i1}$, has already items belong a given group $G(j)$, $1 \leq j \leq k$, which is $\frac{M_{i2}}{s(j)}$. Therefore, to perform less number of broadcast rounds in each group, we choose, as agents, the k things that have the k maximum memory spaces for items in L in each group. The algorithm of agents election is shown in Fig. 5. This algorithm is executed by each thing node in each group. It chooses the first k things that have the largest memory capacities as agents. In the first iteration ($i = 1$) of the loop for the nodes elect an agent for the group $G(1)$, in the second the nodes elect an agent for the group $G(2)$ and so on until the group $G(k)$. As the nodes in each group are at single hop and as the algorithm on each node receives the same entrees the algorithms output the same results in each group.

Let $t_1(G(j)), t_2(G(j)), ..., t_{\delta_j}(G(j))$ the things in a group $G(j)$, $1 \leq j \leq k$, where $t_1(G(j)), t_2(G(j)), ..., t_k(G(j))$ are the agents in this group. Each item of the n items is written under the form $ITEM(s, a, d)$, with s is the source group of the item and a is the destination group and d is the final destination. In the Broadcast to agents algorithm, agent $t_r(G(j))$, $1 \leq r \leq k$, copies the items destined to the group $G(r)$. Namely, each agent $t_r(G(j))$ has to store in its memory the items $ITEM(s, a, d)$, with $a = r$. The details of this procedure are in the algorithm in the Fig. 6. This algorithm is executed in parallel in the groups using a channel $C(j)$ for each group $G(j)$.

Fig. 4 presents a model for the grouping procedure where things $t_f(G(j))$, $1 \leq f, j \leq k$, will be the agents and $RT_{I,j}$ are the remaining things (not agents).

Algorithm for each group $G(j)$**:**
// executed in parallel for the groups $G(1), G(2), ..., G(k)$
begin
for $(i = 1; i \leq k; i++)$ **do**
$NwAgnt\_for\_G(i) = t_z(G(j))$ with $z \in \{1, \delta_j\} \wedge M(t_z(G(j))) \geq M(t_h(G(j))), \forall h \in \{1, \delta_j\}$;
end for
end

Figure 5: Agents election algorithm

Algorithm for each group $G(j)$:
begin
  for $i \in \{1, ..., \delta_j\}$ **do**
// executed in parallel, not ordered
    Thing $s = t_i(G(j)$ broadcasts on the channel $C(j)$ the items $ITEM(s, a, d)$ one by one;
    The agent $t_r(G(j))$ copies in its local memory the item $ITEM(s, r, d)$
  end for
end

Figure 6: Broadcast to agents algorithm

Using Broadcast to agent algorithm, things in each group $G(j)$, broadcast in parallel their items one by one on the channel $C(j)$. At each step, the agent that acts for a group copies in its local memory, from $C(j)$, the items that have as destination things in this group.

Lemma 3.4. *As the thing nodes in each group $G(j)$ broadcast one by one on the channel $C(j)$, and as the broadcast takes place in parallel in the groups, the largest group is the one that performs more broadcast rounds. Consequently, the number of broadcast rounds for this step is:*
$\leq (\frac{mk}{k} + 1) \times MAX(M_i) - 1$, *if* $mk \, \boldsymbol{mod} \, k = 0$,
or
$\leq (\lfloor (\frac{mk}{k}) \rfloor + 1) \times MAX(M_i)$, *if* $mk \, \boldsymbol{mod} \, k \neq 0$. □

Example :

In the network example of Fig. 2 (a). The thing t_9 is the agent for group $G(1)$ in group $G(1)$. This means it stores in its local memory the items broadcast by things in group $G(1)$ and destined to things in group $G(1)$. The thing t_4 is the agent for group $G(2)$ in group $G(1)$. This means is stores in its local memory the items broadcast by things in group $G(1)$ and destined to things in group $G(2)$. The thing t_{14} is the agent for group $G(1)$ in group $G(2)$. This means is stores in its local memory the items broadcast by things in group $G(2)$ and destined to things in group $G(1)$. The thing t_1 is the agent for group $G(2)$ in group $G(2)$. This means it stores in its local memory the items broadcast by things in group $G(2)$ and destined to things in group $G(1)$.

3.1.3. ***Broadcast to the final destinations***

The aim of this step is to broadcast the items sorted by the agents to their final destinations. Namely, broadcast each $ITEM(s, a, d)$ to its destination node d in the group $G(a)$. At this stage, the agents in different groups hold the items of nodes that are not in the agents groups. Therefore, the main concern is managing the broadcast on each channel $C(j)$, $1 \leq j \leq k$, because this last will be the destination of different agents in different groups. This step is decomposed into two phases. The aim of the first phase is to let each agent to know the exact moment when it should broadcast. In the second phase, the agents use the result of the first phase to broadcast correctly the items. The details are next in Phase 1 and Phase 2.

A) Phase 1

Given that each agent cannot monitor the channel continually, because this will prevent it from receiving relevant information on any other channel. To let each agent $t_j(G(j))$ know the exact moment when it should broadcast on the channel $C(j)$, we use this mechanism: let the agents that act for a group $G(j)$ be ordered in a list $H(j) = \{t_1(G(j)), t_2(G(j)), ..., t_{k-1}(G(j))\}$. The algorithm for *Broadcast to the final destinations* step is presented in Fig. 7. This algorithm is executed in parallel, using a channel $C(j)$ for each set of agents $H(j)$.

Taking turns, agents $t_1(G(j)), t_2(G(j)), ..., t_{k-1}(G(j))$ broadcast on the channel $C(j)$, the number of items (i.e $|ITEM(1, j, d)|, |ITEM(2, j, d)|, ..., |ITEM(k-1, j, d)|$). We note that only these agents need to broadcast the number of items. The last agent $t_k(G(j))$ does not need to broadcast, this agent is the thing t in the group $G(j)$ that stores more items in its M_t. That is, because its probability to have already elements that belongs to it is high compared to other things, and therefore there is a high probability to perform less number of broadcast rounds and high probability to prevent other nodes to wait for rounds that in really there was no broadcast operations during them. After, each agent $t_x(G(j))$ can know its turn to broadcast by calculating the sum of number of items to be broadcast by agents $t_1(G(j)), t_2(G(j)), ..., t_{x-1}(G(j))$.

Lemma 3.5. *This phase, clearly, takes $k-1$ broadcast rounds. Because in each group, $k-1$ agents broadcast the number of items on the channels in parallel.*

B) Phase 2

As in Phase 1, a channel $C(j)$, $1 \leq j \leq k$ is assigned to a group $H(j)$, to let the agent broadcast the items to the final destinations. With the sums computed by each agent in Phase 1, each of them broadcasts in the appropriate time its items on the channels. For each item $ITEM(s, a, d)$ broadcast in the channel of group $G(a)$, the final destination thing $t = d$, copies it in its memory.

Algorithm for each list $H(j), 1 < j < k$:
begin
  for $a \in \{1, ..., k-1\}$ **do**
    $t_a(G(j))$ broadcasts $|ITEM(k-1, j, d)|$ on $C(j)$
  end for
  $a \leftarrow 1$
  while$(a < k)$ **do**
    Thing $t_a(G(j))$ broadcasts at time $Time(a)$;
    Thing t_d copies in its local memory the item $ITEM(s, j, d)$;
    $a \leftarrow a + 1$
  end while;
end

Figure 7: Algorithm Broadcast to the final destinations

Procedure : Time(a)
begin
$X \leftarrow 0; h \leftarrow 1;$
while$(h < a)$ **do**
$X \leftarrow X + |ITEM(h, j, d)|$;
$h \leftarrow h + 1$
end while;
Return(X)
end

Figure 8: Procedure Time(a)

Lemma 3.6. *As the broadcasts take place in parallel for the k groups, therefore, the group that contains more items will be the last group to finish the broadcast. Consequently, from lemma 3.3, the number of broadcast rounds for Broadcast to the final destinations is:*
$\leq (\frac{mk}{k} + 1) \times MAX(M_i) - 1$, *if* $mk \bmod k = 0$,
or
$\leq (\lfloor(\frac{mk}{k})\rfloor + 1) \times MAX(M_i)$, *if* $mk \bmod k \neq 0$. □

Theorem 3.7. *From lemma 3.2, lemma 3.3, lemma 3.4, lemma 3.5 and lemma 3.6, the total number of broadcast rounds for this protocol using $O(\frac{n}{k})$ memory has an upper bound:*
$\leq 2(\frac{mk}{k} + 1) \times MAX(M_i) + k$, *if* $mk \bmod k = 0$,
or
$\leq 2(\lfloor(\frac{mk}{k})\rfloor + 1) \times MAX(M_i) + k - 1$, *if* $mk \bmod k \neq 0$.

Theorem 3.8. *From lemma 3.1, lemma 3.4, lemma 3.5 and lemma 3.6, the total number of broadcast rounds for this protocol using $O(\frac{n}{k})$ memory has an lower bound:*
$\geq 2MAX(M_i) - (\left\lfloor \frac{MAX(M_i)}{k} \right\rfloor \times (k-1))$, *if k is odd*
or
$\geq 2MAX(M_i) - ((\left\lfloor \frac{MAX(M_i)}{k+1} \right\rfloor - 1) \times k + 3)$, *if k is even.*

3.2. Protocol with $O(M_i)$ memory, $1 \leq i < T$ for each t_i in IoT(T, n)

In this section, we present an agent-based broadcasting solution for the permutation routing problem in $IoT(T, n)$ that uses an optimal memory space. In this protocol, each thing of memory space for items M_i uses $O(M_i)$ of memory space.

In this protocol an agent $X_{j1,j2}$ may be composed of a single thing or a set of things. $X_{j1,j2}$ is the agent in group $G(j1)$ and acts for group $G(j2)$. $X_{j1,j2}$ stores only items $ITEM(j1, j2, d)$ that have $G(j1)$ as the source group and $G(j2)$ as a destination group. In this protocol, each agent $X_{j1,j2}$ that has M_i of memory space for items does not stores more than $3M_i$ of items.

Let $x(j1, j2)_1, x(j1, j2)_2, ..., x(j1, j2)_{\beta(j1,j2)}$, be the $\beta(j1, j2)$ memory spaces for items of things composing the agent $X_{j1,j2}$ if $|X_{j1,j2}| > 1$.

The main idea to implement the protocol with $O(M_i)$ of memory space is to set up in each group, a *relief agents*. These relief agents are called to store the items when the agent became full. Each agent $X_{j1,j2}$ may have one or several relief agents. $Z_{j1,j2,N}$, $1 \leq N \leq J-2$, is a relief agent number N for agent $X_{j1,j2}$ in group $G(j1)$, where J is the number of groups after applying the grouping algorithm (section 3.2.1). Each relief agent $Z_{j1,j2,N}$, is called to store the items that have $G(j1)$ as source group and $G(j2)$ as destination group if the agent $X_{j1,j2}$ is full (i.e. it stores $3M_i$ items). $Z_{j1,j2,N}$ is also composed of a single thing or a set of things.
Let $z(j1, j2, N)_1, z(j1, j2, N)_2, ..., z(j1, j2, N)_{\beta(j1,j2,N)}$, be the $\beta(j1, j2, N)$ memory spaces for items of things composing the relief agent $Z_{j1,j2,N}$ if $|Z_{j1,j2,N}| > 1$.

Observe that $J-2$ relief agents for each agent are necessary and sufficient in each group. Because if there are J groups, there are J agents in each group. Let us assume that in a given group $G(j)$, all items have $G(1)$ as destination group. $X(j, 1)$ will be able to store its items and the items of $X(j, 2)$, because following the grouping algorithm $X(j, 2) < 3X(j, 1)$. Notice that it remains $J-2$ agents that have $G(1)$ as destination group. Therefore, $J-2$ agents are necessary and sufficient for the agent $X(j, 1)$.

Definition 3.2. *Let a set of m positive integers be in S, $S = \{int_1, int_2, ..., int_m\}$, where $int_a \geq int_b$, $a < b$, $0 < a, b \leq m$*

and let $X1$ and $X2$ be two subsets from set S, where $X1 \cap X2 = \varnothing$.

We say that $s_f(X1) \geq s(X2)$ if there is $X1 = \{e_1 \cup e_2 \cup ... \cup e_v\}$, with, $v \leq m$ and $e_1 + e_2 + ... + e_v \geq s(X2)$ and $e_1 + e_2 + ... + e_{v-1} < s(X2)$, where $s(X2)$ is the size of $X2$ (the sum of the elements in $X2$).

This protocol is composed of three algorithms: *Grouping and agents assigning*, *broadcast to agents* and *broadcast to the final destinations*. The details of each algorithm are discussed next.

3.2.1. Grouping and agents assigning

The algorithm of grouping, assigning of agents and relief agents is presented in Fig. 9. This algorithm groups the things to have a protocol that uses an optimal memory space for the permutation routing. Indeed, this protocol requires $O(M_i)$ memory space for each thing t_i. The algorithm assumes that there are enough memory spaces for items to have at least two groups (two groups is the minimum number that allows broadcasting in parallel). Fig. 10 depicts the model of grouping and agents assigning, where there are agents, relief agents and remaining things. The algorithm sets at the beginning the number of groups $j = 2$ (instruction (2)). From instruction (3) to instruction (12) the algorithm makes the first two groups. From instruction (13) to instruction (58), the algorithm at each time, checks if it is possible to add a new group. If the rest of memory spaces for items is not enough to make a new group, the algorithm proceeds to distribute these memory spaces for the current j groups following the Grouping Algorithm in Fig. 3. This is done from instruction (60) to instruction (61).

We detail the explanation of the algorithm of grouping, assigning of agents and relief agents using the memory spaces for items of 42 things, in IoT (42, 326) presented in Fig. 11.

Example:

Within the example of memory spaces for items in Fig. 11, we have:
$L = \{15, 15, 13, 13, 13, 13, 12, 12, 12, 12, 12, 12, 11, 11, 11, 10\}$ $\cup \{10, 10, 9, 9, 9, 8, 8, 8, 7, 7, 7, 7, 6, 6, 6, 6, 6, 5, 4, 4, 3, 3, 3, 3\}$ $\cup \{3, 3, 3, 3\}$. Applying the algorithm, we add into $G(1)$, the agent $X_{1,1}$, which is the thing having the maximum memory space, which is the first thing t_i that has $M_i = 15$, so $G(1) = \{15\}$ (instruction 4). After, we add the agent $X_{1,2} = \{15\}$, because $s_f(X_{2,1}) \geq s(X_{1,1})$. So $G(1) =$ $\{15, 15\}$ (instruction (6)). In the next instruction (6) we add the first agent, $X_{2,1} = MAX(L - \{X_{1,1} \cup X_{1,2}\}) = 13$ into group $G(2)$. So $G2$ becomes $G2 = \{13\}$. After, we add the second agent $X_{2,2} = \{13\}$ because $s_f(X_{2,2}) \geq s(X_{2,1})$ (instruction (9)). Therefore, we have $G(2) = \{13, 13\}$. The list L becomes $L = L - \{15, 15, 13, 13\}$ (instruction 10). So, at this stage, the grouping into two groups is possible. Because, for each group there are two agents that act for it, one agent in $G(1)$ and the other in $G(2)$. At this stage, relief agents are not needed, because each agent can store all items that are in its group.

In the next step, we enter into the loop *while* to check if we may add a new group (instruction (13)). If that is possible, we add it and in the next iteration of the loop, we check if the grouping into four groups is possible and so on, as long as the updated list L is not empty. Typically, we can add a new group if there is enough memory spaces for items that allow saving items for the new group. This is by setting up agents for the new group in each group.

In instruction (14), we assume that adding a new group $G(3)$ is possible, so we increment the number of groups. Therefore, in our example $j = 3$. Also, we record the list L in another list S, because if the grouping into j group is not possible we need the memory spaces that cannot make a new group to distribute them on the current $j - 1$ groups.

From instruction (15) to instruction (25), we add the agents into the new group $G(3)$. We add 3 ($y \leq 3$) agents, in instruction (17) and (21) (to each group an agent that acts for it). At the last iteration of loop *for* (instruction (25)), we get $G(3) = X_{3,1} \cup X_{3,2} \cup X_{3,3} = \{13, 13, 12, 12\}$, where $X_{3,3} = \{12, 12\}$ and we get $S = L - \{13, 13, 12, 12\}$. For each memory space added we delete it from the list S (instructions (18) and (22)). If the memory space for items cannot make agents in the new group the algorithm jumps to label A to get out from the loop because the grouping into j groups is not possible (instruction (23)).

From instruction (26) to instruction (34), we add to each agent in group $G(j)$, $j-2$ relief agents. At the end of the loop *for* (instruction (34)), we get $Z_{3,1,1} = \{12, 12\}$, $Z_{3,2,1} = \{12, 12\}$, $Z_{3,3,1} = \{11, 11, 11\}$, therefore, $G(3) =$ $G(3) \cup Z_{3,1,1} \cup Z_{3,2,1} \cup Z_{3,3,1}$ and $S = S - G(3)$.

From instruction (35) to instruction (43), we add to each agent in each group $G(u), u < j$ of the previous groups a relief agent which may be called from an agent to store the items that have as destination group $G(j)$. These relief agents are: $Z_{1,1,1} = \{10, 10\}$, $Z_{1,2,1} = \{10, 9, 9\}$, $Z_{2,1,1} = \{9, 8\}$, $Z_{2,2,1} = \{8, 8, 7\}$. From instruction (44) to instruction (56), to each group $G(u), u < j$, an agent which acts for the group $G(j)$ is added. And for each agent added, we add its relief agent. From instruction (44) to instruction (56), we get:

For $G(1)$:
$X_{1,3} = \{7, 7, 7\}$, so $S = S - \{7, 7, 7\}$, $Z_{1,3,1} = \{6, 6, 6, 6, 6\}$, so $S = S - \{6, 6, 6, 6, 6\}$

For $G(2)$:
$X_{2,3} = \{5, 4, 4\}$, $Z_{2,3,1} = \{3, 3, 3, 3, 3, 3, 3, 3\}$,
$Z_{2,2,1} = \{5, 5, 5, 4\}$, $Z_{2,3,1} = \{4, 4, 4, 4, 4\}$.

So the grouping into three groups, $G(1)$, $G(2)$ and $G(3)$ is possible, where there are 3 agents in each group and each agent has 2 relief agents.

The instruction (60) is executed if the grouping into j group is not possible. In this case, the remaining of memory spaces for items, which is in L (recovered in instruction (57)) is distributed on the groups following the algorithm 3 (instruction (61)). $RT_{c,d}$ is the remaining thing number d added into the group c, $1 \leq c \leq j$, see Fig. 10.

3.2.2. Broadcast to agents

Let J be the final number of groups after applying the grouping algorithm of the previous step. In this step, each thing in each group $G(j)$, $1 \leq j \leq J$, broadcasts its items in the channel $C(j)$. In each group, the agent $X_{j,j2}$ copies from the channel $C(j)$ the items $ITEM(j, j2, d)$ that have $G(j)$ as source group and have as destination the group $G(j2)$. Proceeding sequentially, the things in each group broadcast one by one on the channel, each agent that has $|X_{j,j2}| = 1$ copies without any problem the items. However, in each agent that has $|X_{j,j2}| > 1$, the things in this agent should know the time at which each thing should start copying the items from the channel $C(j)$.

The mechanism used for agents that have $|X_{j,j2}| > 1$ to know the exact moment when each thing $x(j, j2)_i$, $1 \leq j, j2 \leq J$, $0 \leq i \leq \beta(j, j2)$ it should start copying the items from channel $C(j)$ is the following: Let $M(x(j, j2)_i)$ be the memory capacity of $x(j, j2)_i$. Each thing $x(j, j2)_i \in X_{j,j2}$, $1 \leq, j, j2 \leq J$, has a counter $c(j, j_2)_i$. For each item $ITEM(j, j2, d)$ broadcast in the channel, $x(j, j2)_i$ increments $c(j, j2)_i$. The station $x(1, j)_1$, $1 \leq j \leq J$, is the first to start copying the items. Each thing $x(j, j2)_i$, $f_1 \neq 1$, starts copying the items $ITEM(j, j2, d)$ after $c(j, j2)_i = 3(M(x(j, j2)_1) + M(x(j, j2)_2) + ... + M(x(j, j2)_{i-1}))$, with, $0 \leq i \leq \beta(j, j2), x(j, j2)_i \in X_{j,j2}$.

We recall that for each group $G(j)$, relief agents $Z_{j,j2,h}$, $1 \leq j2 \leq J$, $1 \leq h \leq J - 2$, takes turn of copying when the agent $X_{j,j2}$ is full. Therefore, firstly, each $Z_{j,j2,h}$ should know when the agent $X_{j,j2}$ and the relief agents $Z_{j,j2,h1}$, $1 \leq h1 < h$ are full. That is, each relief agent $Z_{j,j2,h}$ should know the exact moment when it starts copying and therefore each $z(j, j2, h)_i, 0 \leq i \leq \beta(j, j2, h)$ in

1. **begin algorithm**
2. $j \leftarrow 2$;
3. **if** $(|L| \geq 4)$ **then**
4. Add $X_{1,1} = MAX(L)$ into group $G(1)$;
5. **if** $(\exists X_{1,2} = x(1,2)_1 \cup x(1,2)_2 \cup .... \cup x(1,2)_{\beta(1,2)}$ in $L - X_{1,1}$, with $s_f(X_{1,2}) \geq s(X_{1,1}))$ **then**
6. Add $X_{1,2}$ into $G(1)$; Add $X_{2,1} = MAX(L - \{X_{1,1} \cup X_{1,2}\})$ into group $G(2)$
7. **end if**
8. **if** $(\exists X_{2,2} = x(2,2)_1 \cup x(2,2)_2 \cup .... \cup x(2,2)_{\beta(2,2)}$ in $L - \{X_{1,1} \cup X_{1,2} \cup X_{2,1}\}$, with $s_f(X_{2,2}) \geq s(X_{2,1}))$ **then**
9. Add $X_{2,2}$ into $G(2)$;
10. $L \leftarrow L - \{X_{1,1} \cup X_{1,2} \cup X_{2,1} \cup X_{2,2}\}$
11. **end if**
12. **end if**
13. **while** $(L \neq \varnothing)$ **do**
14. $j \leftarrow j + 1$; $S \leftarrow L$;
15. **for** $(y = 1$**to**$y = j)$ **do**
16. **if** $((|S| \geq 2)$ and $(y == 1))$ **then**
17. Add $X_{j,y} = MAX(S)$ to $G(j)$;
18. $S \leftarrow S - X_{j,y}$
19. **end if**
20. **if** $(\exists X_{j,y} = x(j,y)_1 \cup x(j,y)_2 \cup .... \cup x(j,y)_{\beta(j,y)}$ in S, with $s_f(X_{j,y}) \geq s(X_{j,y-1}))$ **then**
21. Add $X_{j,y}$ into $G(j)$;
22. $S \leftarrow S - X_{j,y}$;
23. **else GOTO A;**
24. **end if**
25. **end for**
26. **for** $(e = 1$**to**$e = j)$ **do**
27. **for** $(y = 1$**to**$y = j - 2)$
28. **if** $(\exists Z_{j,e,y} = z(j,e,y)_1 \cup z(j,e,y)_2 \cup .... \cup z(j,e,y)_{\beta(j,e,y)}$ in S, with $s_f(Z_{j,e,y}) \geq s(Z_{j,e,y-1}))$ **then**
29. Add $Z_{j,e,y}$ into $G(j)$;
30. $S \leftarrow S - Z_{j,e,y}$;
31. **else GOTO A**
32. **end if**
33. **end for**
34. **end for**
35. **for** $(y = 1$**to**$y = j - 1)$ **do**
36. **for** $(e = 1$**to**$e = j - 1)$ **do**
37. **if** $(\exists Z_{y,e,j-2} = z(y,e,j-2)_1 \cup z(y,e,j-2)_2 \cup .... \cup z(j,e,y)_{\beta(y,e,j-2)}$ in S, with $s_f(Z_{y,e,j-2}) \geq s(Z_{y,e,j-3}))$
38. Add $Z_{y,e,j-2}$ into $G(y)$;
39. $S \leftarrow S - Z_{j,e,j-2}$
40. **end if**
41. **else GOTO A**
42. **end for**
43. **end for**
44. **for** $(y = 1$**to**$y = j - 1)$ **do**
45. **if** $(\exists X_{y,j} = x(j,y)_1 \cup x(y,j)_2 \cup .... \cup x(y,j)_{\beta(y,j)}$ in S, with $s_f(X_{y,j}) \geq s(X_{y,j-1}))$ **then**
46. Add $X_{y,j}$ into $G(y)$;
47. **for** $(e = 1$**to**$e = j - 2)$ **do**
48. **if** $(\exists Z_{j,e,y} = z(j,e,y)_1 \cup z(j,e,y)_2 \cup .... \cup z(j,e,y)_{\beta(j,e,y)}$ in L, with $s_f(Z_{j,e,y}) \geq s(Z_{j,e,y-1}))$ **then**
49. Add $Z_{y,j,e}$ into group $G(y)$;
50. $S \leftarrow S - Z_{j,e,y}$;
51. **else GOTO A**
52. **end if**
53. **end for**
54. **else GOTO A**
55. **end if**
56. **end for**
57. $L \leftarrow S$
58. **end while;**
59. **GOTO B;**
60. A: $j \leftarrow j - 1$;
61. B: **if** $(L \neq \varnothing)$ **then Grouping**$(L, j, LG = \{G(1), G(2), ...G(j)\})$;
62. **end algorithm**

Figure 9: Grouping and Agents Assigning Algorithm

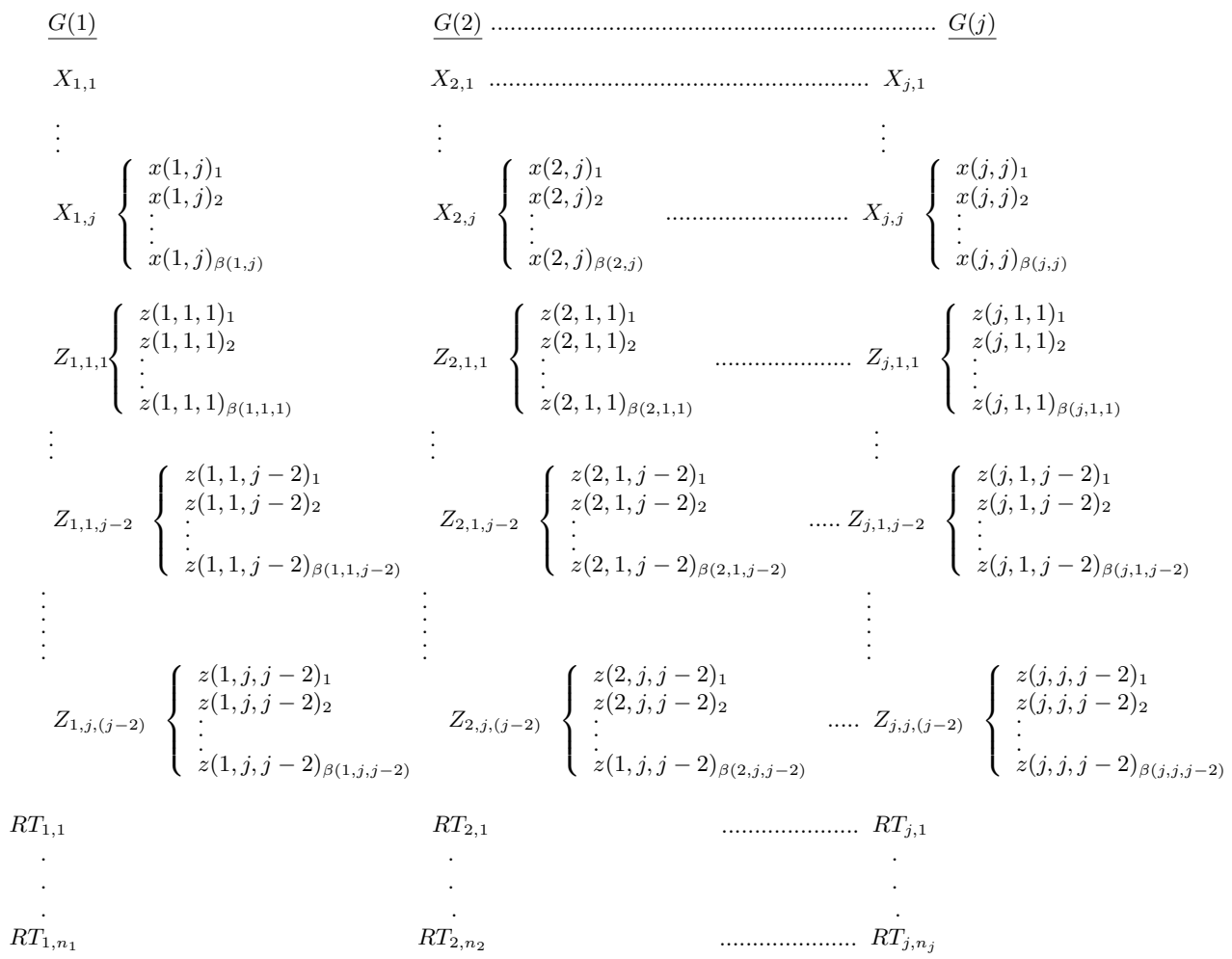


Figure 10: A model for the grouping and agents assigning

15, 15, 13, 13, 13, 13, 12, 12, 12, 12, 11, 11, 11, 10, 10, 10, 9, 9, 9, 8, 8, 8, 7, 7, 7, 7, 6, 6, 6, 6, 6, 5, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3

Figure 11: An example of memory spaces for items of IoT(42, 326)

$Z_{j,j2,h}$ should know the moment when it should start copying. The mechanism used for this purpose is the following: Each relief agent $Z_{j,j2,h}$, $1 \leq h \leq J-2$ for the agent $X_{j,j2}$ has a counter $C(j, j2, h)$. For each item $ITEM(j, j2, d)$ broadcast in the channel $C(j)$, $Z_{j,j2,h}$ increments $C(j, j2, h)$. Each agent $Z_{j,j2,h}$, starts copying after $C_{j,j2,h} = 3(M(X_{j,j2}) + M(Z_{j,j2,1}) + M(Z_{j,j2,2}) + ... + M(Z_{j,j2,h-1}))$. Now, we give the moment when each thing $z(j, j2, h)_i \in Z_{j,j2,h}$, $1 \leq h \leq J-2$, has a counter $c(j, j2, h)_i$. For each item $ITEM(j, j2, d)$ broadcast in the channel $C(j)$, $z(j, j2, h)_i$ increments $c(j, j2, h)_i$. The station $z_{1,j2,h} \in Z_{j,j2,h}$ is the first to start copying the items at the time $C(j, j2, h)$. Each thing $z(j, j2, h)_i$, $i \neq 1$, starts copying after $c(j, j2, h)_i = C(j, j2, h) + 3(M(z(j, j2, h)_1) + M(z(j, j2, h)_2) + ... + M(z(j, j2, h)_{i-1}))$.

3.2.3. Broadcast to final destinations

Using a principle similar to the one in section 3.1.3, in this step the agents broadcast each item $ITEM(j, j2, d)$ to its final destination t_d. To implement this step, we put now the agents that store items with same destination group in new group.

Let $H(j)$, $1 \leq j \leq J$, be the group of agents that store items that have as destination the group $G(j)$. Namely, $H(j) = \{X_{1,j} \cup Z_{1,j,1} \cup ... \cup Z_{1,j,J-2}\} \cup \{X_{2,j} \cup Z_{2,j,1} \cup ... \cup Z_{2,j,J-2} \cup ...\} \cup \{X_{J,j} \cup Z_{J,j,1} \cup ... \cup Z_{J,j,J-2}\}$ is the set of agents and relief agents that act for group $G(j)$. In this last step, each channel $C(j)$ is assigned to the agents that have items to be broadcast to destinations in group $G(j)$. The principle of this step is to allow each agent in $H(j)$ to know the exact time at which it should broadcast its items in the channel $C(j)$. The algorithm for this step is presented in Fig. 12. This algorithm is executed in parallel in each group, using a channel $C(j)$ for each set of agents $H(j)$, $1 \leq j \leq J$.

Algorithm for each set $H(j), 1 < j < J$**:**
begin
  for $i \in \{1, ..., J\}$ **do**
    if $(|X_{i,j}| = 1)$ **then**
      Broadcast items one by one on $C(j)$;
      Thing t_d copies $ITEM(i, j, d)$ in its local memory;
      else
       $v \leftarrow 1$
      while $(v < \beta(i, j))$ **do**
      Agent $x(i, j)_v \in X_{i,j}$ broadcasts items
       $ITEM(i, j, d)$ on $C(j)$;
       Thing t_d copies $ITEM(i, j, d)$ in its local memory;
        $v \leftarrow v + 1$
      end while;
    end if
    $t \leftarrow 1$
    while $(t \leq J - 2)$
      if $(|Z_{i,j,t}| = 1)$ **then**
        Broadcast items one by one on $C(j)$;
        Thing t_d copies $ITEM(i, j, d)$ in its local memory;
        else for $(v = 1, v < \beta(i, j, t), v + +)$
              Agent $z(i, j, t)_v \in Z_{i,j,t}$ broadcasts items
              $ITEM(i, j, d)$ on $C(j)$;
              Thing t_d copies $ITEM(i, j, d)$ in its memory
            end for
      end if
      $t \leftarrow t + 1$
    end while;
  end for
end

Figure 12: Broadcast to final destinations algorithm

4. Experimental results

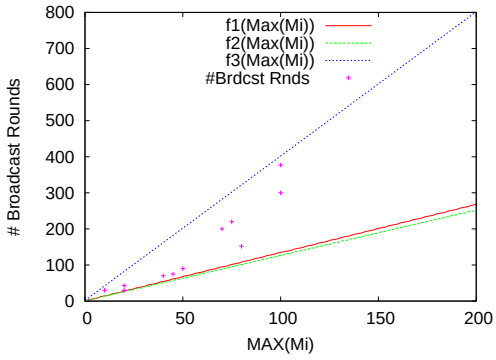


Figure 13: Number of broadcast rounds with $k = 3$

To demonstrate the effectiveness and the performance of the proposed protocols and to compare them to the related works, we implemented them using $C++$. We simulated the two protocols on a laptop with processor Intel(R) Core(Tm) i5, 2.53 Ghz with 4 GB of memory. The following simulation results are the average of 100 tests (for each number of nodes) on the topology of connected and randomly generated networks of 100, 200, 400, 600, 800 and 1000 things in 50 x 50 (m^2) simulation area.

The points addressed in this section are:

- Demonstrate through different tests the number of broadcast rounds with the theorems theoretically proved,

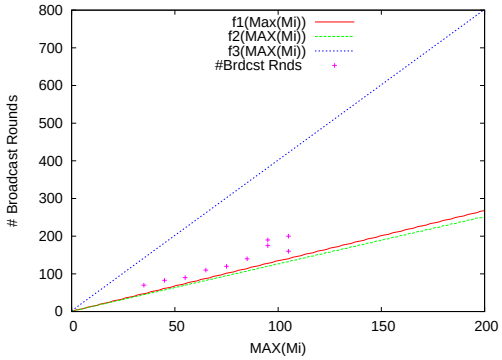


Figure 14: Number of broadcast rounds with $k = 6$

- Comparison of the two protocols together and with other protocols that use other grouping procedures,
- Comparison of the protocols with other protocols existing in the state of the art, applied for the permutation routing in IoT.

Fig. 13, Fig. 14, and Fig. 15 present the number of broadcast rounds for our first proposed algorithm with $k = 3$, $k = 6$, $k = 7$ respectively. With a simulation on 1000 things for diffrent values of $MAX(M_i)$ (35, 60, 45, 55, 65, 75, 85, 95, 105), where $f1(MAX(M_i)) = 2(\frac{mk}{k} + 1) \times MAX(M_i) + k$, if $mk \bmod k = 0$, $f2(MAX(M_i)) = 2(\lfloor(\frac{mk}{k})\rfloor + 1) \times MAX(M_i) + k - 1$, if $mk \bmod k \neq 0$, and $f3(MAX(M_i)) = 2MAX(M_i) - ((\lfloor \frac{MAX(M_i)}{k+1} \rfloor - 1) \times k + 3)$.

These results come to confirm our theoretical studies on the number of broadcast rounds. We see in Fig. 13, Fig. 14, and Fig. 15 that for different experiments, the number of broadcast rounds is always greater or equals to $f1(MAX(M_i))$ or $f2(MAX(M_i))$ and inferior or equals to $f3(MAX(M_i))$.

Given that the number of broadcast rounds depends on the group that has the highest number of items (from Theorem 3.8). Therefore, to show the superiority of our grouping procedures in the first step of our protocols (abbreviated as MaMG, for *MAX MAX Grouping*, with MaMG1 is for the protocol that uses $O(\frac{n}{k})$ memory space and MaMG2 is for the protocol that uses $O(M_i)$ memory space for each thing t_i. We compare the number of broadcast rounds of our protocols and the number of broadcast rounds with other protocols that use other grouping procedures. The principle of the first grouping to compare with is to add randomly the memory spaces that contain the items to the k groups (abbreviated as RRG, for *Random Random Grouping*), this protocol is important to compare with, because it adds directly the elements in the groups, without spending a time to seek the min or the MAX. The principle of the second grouping procedure is to add at each step to the minimum of the k groups the maximum of the memory spaces (abbreviated as MiMG, for *MIn MAX Grouping*). Fig. 16 compares the number of broadcast rounds of our protocols and other protocols that use different procedures

of grouping. We see that, using our grouping algorithm optimizes the number of broadcast rounds because this grouping is the nearest to give groups with same sizes in order to balance the loads.

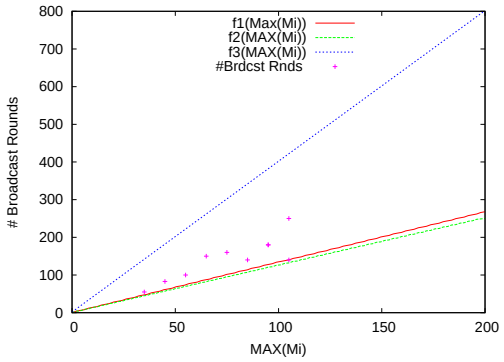


Figure 15: Number of broadcast rounds with $k = 7$

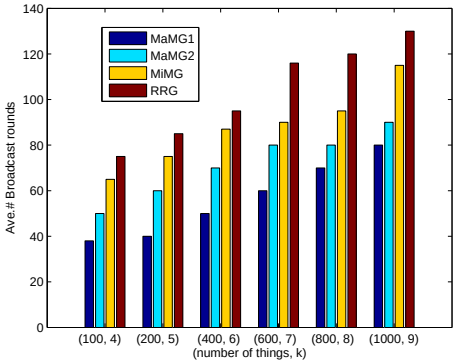


Figure 16: Comparison of the number of broadcast rounds with other protocols that use other procedures of grouping

To show the advantages of our solutions and the need to them, we compare the results of our works and the results of the literature works on the broadcast for the permutation routing applied on the broadcast for the internet of things (these protocols are applied as such they are on IoT). This comparison is shown in Fig. 17 and Fig. 18. Fig. 17 depicts a comparison with a protocol in [17] that uses $O(\frac{n}{k})$ of memory space and Fig. 18 compares with a solution in [17] that uses $O(\frac{n}{T})$ memory space, which is optimal in term of memory space used as our solution that uses $O(M_i)$ memory space. The result shown are based on the number of things and the number of channels used k.

We see in these figures that our proposed protocols achieve the broadcast for the permutation routing problem in IoT using less number of broadcast rounds compared to the protocol in [17]. This is because the grouping method used in the protocols of [17] are based on the number of nodes and not on the number of items. As consequence, they do not give groups with moderate differences in the sizes and the number of broadcast rounds depends directly on the grouping algorithm. If the items are distributed

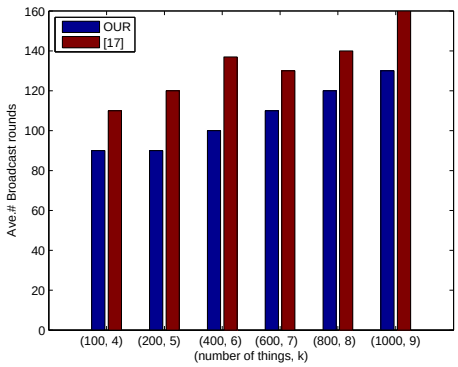


Figure 17: Comparison of the number of broadcast rounds of protocols that use $O(\frac{n}{k})$ memory space, our protocol and the protocol in [17]

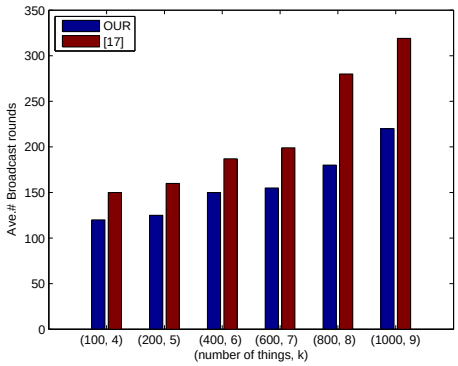


Figure 18: Comparison of the number of broadcast rounds of our protocol that uses $O(M_i)$ memory space and the one in [17] that uses $O(\frac{n}{T})$ memory space

on the groups with a few differences then the number of broadcast rounds will be near to the optimal. That is, the grouping in [17] is not suitable for heterogeneous nodes, that store different number of items as the internet of things. We remark that our protocol with $O(M_i)$ memory space for each thing t_i, uses more broadcast rounds compared to our protocol that uses $O(\frac{n}{k})$ memory space. That is, the first one must use less number of channels, because each agent in this protocol needs to have other agents (relief agents) in order to help it to store the elements to have a protocol that achieves the permutation routing with an optimal memory space complexity.

5. Conclusion

In this work, we presented a root research paper for the broadcast of information in the Internet of things. We proposed two agent-based memory and energy-efficient permutation routing protocols for single hop network composed of heterogeneous nodes. The first performs less number of broadcast rounds compared to the second protocol and uses $O(\frac{n}{k})$ memory space, where n is the total

number of packets and k is the number of communication channels. The second protocol uses less communication channels number compared to the first one and it is optimal in term of memory usage complexity. It uses $O(M_i)$ memory space for each thing t_i that stores at the beginning M_i items. These protocols perform efficiently with respect to the number of broadcast rounds using a parallel broadcasting. Both protocols use a few communication channels to achieve the broadcast operations and achieve the permutation routing without conflict and collision at the communication channels. Consequently, the proposed protocols permit in real application to be tolerant to channel faults.

However, some open problems remain to be investigated in the future, such as:

- Studying a fault tolerance permutation routing for IoT
- Proposing a dynamic solutions (appearing and disappearing of things)
- Proposing solutions that deal with the sleep/wakeup states for things to save energy
- Studying of the different attacks that may occur and propose secure protocols

References

[1] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer, P. Doody, *Internet of Things Strategic Research Agenda*, Chapter 2 in Internet of Things Global Technological and Societal Trends, River Publishers, ISBN 978-87-92329-67-7, 2011

[2] M. Vecchio, R. Giaffreda, F. Marcelloni *Adaptive Lossless Entropy Compressors for Tiny IoT Devices*,IEEE Transactions on Wireless Communications, Volume: 13, Issue: 2 pp: 1088-1100, 2014

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, *Internet of Things for Smart Cities*, IEEE Internet of Things Journal, Vol. 1, No. 1, 2014

[4] X. Li, R. Lu, X. Liang, X. Shen, J. Chen and X. Lin, *Smart community: An internet of things application*, IEEE Commun. Mag., vol. 49, no. 11, pp.68 -75, 2011

[5] K. Chang and J. Chen ,*A survey of trust management in WSNs, internet of things and future internet*, KSII Trans. Internet Inform. Syst., vol. 6, no. 1, pp.5-23, 2012

[6] W. Lia , F. C. Delicatob, P. F. Piresb, Y. C. Leea, A. Y. Zomayaa, C. Micelib, L. Pirmezb *Efficient allocation of resources in multiple heterogeneous Wireless Sensor Networks*, Journal of Parallel and Distributed Computing Volume 74, Issue 1, Pages 1775-1788, January 2014

[7] S. Distefanoa, G. Merlinoc, A. Puliafitoc , *A utility paradigm for IoT: The sensing Cloud*, Journal of Pervasive and Mobile Computing, 2015

[8] Y. Zhou, C. Huang, T. Jiang, and S. Cui *Wireless Sensor Networks and the Internet of Things: Optimal Estimation With Nonuniform Quantization and Bandwidth Allocation*, IEEE SENSORS JOURNAL, VOL. 13, NO. 10, OCTOBER 2013

[9] H. Ning, H. Liu, and L. T. Yang *Aggregated-Proof Based Hierarchical Authentication Scheme for the Internet of Things*, IEEE Transactions on Parallel and Distributed Systems, VOL. 26, NO. 3, Pages 657-667, 2015

[10] D.P.F. Moller, H. Vakilzadian *Wireless communication in aviation through the Internet of Things and RFID*, , 2014 IEEE International Conference on Electro/Information Technology (EIT), pp.602,607, June 2014

[11] L. Atzori, A. Iera, and G. Morabito *The Internet of Things: A survey* Journal Computer Networks: The International Journal of Computer and Telecommunications Networking Volume 54, Issue 15, Pages 2787-2805, October 2010

[12] 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, Volume 29, Issue 7, Pages 1645-1660, September 2013

[13] J. Yick, B. Mukherjee, and D. Ghosal, *Wireless sensor network survey*,Computer Networks, vol. 52, no. 12, pp. 2292-2330, 2008

[14] O. Iova, F. Theoleyre and 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

[15] K. Nakano, S. Olariu and J. L. Schwing. *Broadcast-efficient algorithms on the coarse-grain broadcast communication model with few channels*, Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing,Pages 31-35,DOI: 10.1109/IPPS.1998.669885,Orlando, FL, 1998

[16] K. Nakano, S. Olariu and A. Y. Zomaya *Energy-Efficient Permutation Routing in Radio Networks*, IEEE Transactions on Parallel and Distributed Systems, Vol. 12, No. 6, pp. 544-557, 2001

[17] K. Nakano, S. Olariu and J. L. Schwing *Broadcast-Efficient Protocols for Mobile Radio Networks*, IEEE Transactions on Parallel and Distributed Systems, Vol. 10, No. 12, pp. 1276-1289,1999

[18] J. L. Trff, A. Ripke *Optimal Broadcast for Fully Connected Networks*,HPCC 2005: , First International Conference on High Performance Computing and Communications, 45-56, Sorrento, Italy, September 21-23, 2005

[19] S. Rajasekaran, L. Fiondella, D. Sharma, R. A. Ammar, N. Lownes *Communication and energy efficient routing protocols for single-hop radio networks*, J. Parallel Distrib. Comput, 2012

[20] P. Di Marco, G. Athanasiou, P. Mekikis, and C. Fischione, *MAC-Aware Routing Metrics for the Internet of Things*, Computer Communications, Volume 74, Pages 7786, 2016

[21] Z. Shelby, C. Bormann *6LoWPAN: The Wireless Embedded Internet*, , Torquay, UK:Wiley John & Sons, 2009

[22] B.A. Klein *RPL: IPv6 Routing Protocol for Low Power and Lossy Networks*, , (2011) . 59-66

[23] S. M. Das, H. Pucha, K. Papagiannaki, Y. C. Hu *Studying Wireless Routing Link Metric Dynamics*, Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, San Diego, 327-332, 2007

[24] Park, S., Cho, S. and Lee, J. *Energy-Efficient Probabilistic Routing Algorithm for Internet of Things*, Journal of. Applied Mathematics, 2014

[25] K. Akkaya, F. Senel, B. McLaughlan *Clustering of wireless sensor and actor networks based on sensor distribution and connectivity* , Elsevier Journal of Parallel Distributed Computing Volume 69, Issue 6, Pages 573587, June 2009

[26] J. L. Trff, A. Ripke *Optimal broadcast for fully connected processor-node networks* , J. Parallel Distrib. Comput. 68, 887-901, 2008

[27] C. Zhan, Victor C. S. Lee, J. Wang, and Y. Xu *Coding-Based Data Broadcast Scheduling in On-Demand Broadcast* , IEEE Transactions on Wireless Communications, VOL. 10, NO. 11, 2011

[28] M. Raynal, J. Stainer, J. Cao, W. Wu *A Simple Broadcast Algorithm for Recurrent Dynamic Systems* , 28th IEEE International Conference on Advanced Information Networking and Applications, AINA 2014, Victoria, BC, Canada, May 13-16, 2014

[29] R. Narasimhan, *Cooperative broadcast channels with hybrid*

ARQ,IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC),Pages 1579-1584, Toronto, ON, Canada, 2011

[30] J. F. Myoupo *Concurrent Broadcasts-Based Permutation Routing Algorithms in Radio Networks*, IEEE Symposium on Computers and Communications, pp. 1272-1278, 2003

[31] A. Datta *A Fault-Tolerant Protocol for Energy-Efficient Permutation Routing in Wireless Networks*,IEEE Trans. Computers 54(11): 421 2005

[32] S. Rajasekaran, D. Sharma, and R. Ammar, N. Lownes *An Efficient Randomized Routing Protocol for Single-Hop Radio Networks*, Proc. of the 39th International Conference on Parallel Processing (ICPP), Pages: 160 - 167, 2010

[33] D. Karimou and J. F. Myoupo, *A Fault Tolerant Permutation Routing Algorithm in Mobile Ad Hoc Networks*, International Conference on Networks-Part II, pp. 107-115, 2005

[34] A. Datta and A. Y. Zomaya, *An Energy-Efficient Permutation Routing Protocol for Single-Hop Radio Networks*,IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 4, pp. 331-338, 2005

[35] I. S. Walls and J. erovnik *Optimal Permutation Routing on Mesh Networks*, International Network Optimization Conference, Belgium, 22-25, 2008

[36] D. Karimou and J. F. Myoupo *An Application of an Initialization Protocol to Permutation Routing in a Single-hop Mobile Ad-Hoc Networks*, Journal of Super-computing, Vol. 31, No. 3, pp. 215-226, 2005

[37] D. Karimou and J. F. Myoupo *Randomized Permutation Routing in Multi-hop Ad Hoc Networks with Unknown destinations*, IFIP International Federation of Information Processing, Vol. 212, pp. 47-59, 2006

[38] A. B. Bomgni, J. F. Myoupo, *A Deterministic Protocol for Permutation Routing in Dense Multi-Hop Sensor Networks*,Wireless Sensor Network 2(4): 293-299, 2010

[39] H. Lakhlef, A. Bomgni et J. F. Myoupo., *An Efficient Permutation Routing Protocol in Multi-Hop Wireless Sensor Networks*, International Journal of Advancements in Computing Technology (3) 207-214, 2011