A 2-hop Coloring-Based Collision Free Infrastructure Design for Wireless Sensor Networks

Ilker Korkmaz
Department of Computer Engineering
Izmir University of Economics
Izmir, Turkey
ilker.korkmaz@ieu.edu.tr

Orhan Dagdeviren, Mehmet Emin Dalkilic
International Computer Institute
Ege University
Izmir, Turkey
{orhan.dagdeviren, mehmet.emin.dalkilic}@ege.edu.tr

***Abstract*—This paper mainly proposes a design for a communication infrastructure for Wireless Sensor Networks. The proposed design prevents message collisions by arranging the time schedules to send, receive, forward and overhear packets of the nodes considering 2-hop graph coloring mechanism. The system aims to exclude the compromised nodes in the network using the overhearing mechanism, and copes with recovering the paths of the legitimate nodes using multipath redundancy. The proposed scheduling-based and overhearing supported infrastructure brings the advantage of providing the Sensor Networks with both reliable communication using backup paths and energy conservation by preventing the collisions.**

Keywords—2-hop coloring; wireless sensor networks; collision; scheduling; collision free scheduling

I. INTRODUCTION

Wireless Sensor Networks (WSNs) containing huge amounts of resource-constrained nodes embedded with many-purpose sensors [1] are well-known in the Internet of Things (IoT) era. Many academic and industrial researches on architectural design and application specific scenario implementation of wireless sensor WSNs have been proposed [1-5]. Integration of WSN into any kind of Information System (IS) has attracted the interest of the researchers who have been studying the event-driven or periodical automated processes within the system environment using low power nodes that run on ad-hoc designs [6,7]. WSNs in different topologies have been used with various applications integrated into many ISs that are expected to be secure regarding not only the data transmitted but the communication network used as well [8].

In this paper, we propose a design of a 2-hop coloring-based, collision free and reliable communication infrastructure. The proposed system is a deliberate infrastructure to be used as a base mechanism in any kind of wireless-based overlay network. In general, it is considered to be used as a cross-layer protocol for IoT based networks; whereas, in particular it is designed for wireless and ad-hoc networks with lots of resource-constraint nodes, i.e., wireless sensor networks. The proposed architecture contributes to the WSNs with the following mechanisms:

- 2-hop coloring based time slot assignment for the whole network to avoid any conflict/collision meanwhile the packet transmission.
- For a reliable end-to-end communication between the sink and any node of the network, establishment of a breadth-first tree with the backup paths to immediately recover the leaves of the nodes with failure.
- Providing all nodes with overhearing of their own packets being forwarded by their parents.

II. SYSTEM ANALYSIS AND DESIGN

The proposed system is designed to be a main infrastructure that could be used in many different wireless ad-hoc topologies for many different application areas. The main system is actually designed to run in an appropriate fault-tolerant manner in case that some of the nodes get faulty, are broken, or are compromised by the adversaries to passively or actively attack to the routing mechanism and/or data. The design to detect the adversaries or nodes with fault and then to recover the system is based on overhearing concept. A node in the system overhears its raw packet previously sent to its parent while the packet is to be forwarded to the grandparent on the way to sink of the tree. Even if there is no adversary in the system and all the nodes are running correctly with the legitimate rules of the system, there may be another issue that comes along with the communication media access, i.e., packet collision. Collisions in the wireless media lead to a huge energy waste overall the network due to attempting to resend the packets again. To overcome this problem and to prevent the useless energy consumption, the proposed system is designed to run in an appropriate time scheduled manner that the collisions are avoided within the communication infrastructure.

A. System Architecture

Fig. 1 shows the layers and co-layers of the proposed system architecture. A WSN protocol stack may be figured with Physical layer and Medium Access Control (MAC) layer at bottom, and application layer at top [1]. A general network layer or a kind of application specific routing protocol layer or a group of co-layers to assist networking may be wrapped in the middle. The proposed infrastructure is a cross-layer design approach that means some data from a layer may be used as a parameter in another layer protocol or may affect another layer's operating process. In Fig. 1, the arrows are depicted to show the relation between the layers and to point which layer affects which other layer in the proposed cross-layer design.

978-1-5090-3784-1/16/$31.00 ©2016 IEEE

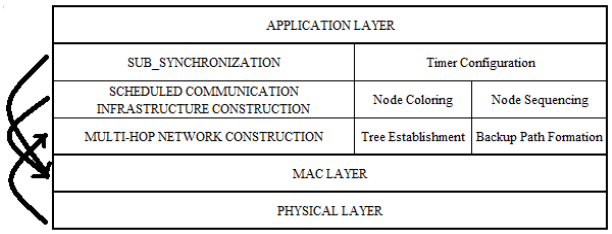

Fig. 1. Proposed system architecture.

Regarding the proposed system architecture, the physical layer implementation is used as it is. The MAC layer is oriented by the parameters from the node coloring, node sequencing and timer configuration co-layers. Node coloring module mainly colors all the nodes of the multi-hop network using the 2-hop vertex coloring approach in a tree. 2-hop coloring is used to avoid collisions in child-parent-grandparent relation that will lead to a collision free communication for a node to send to, receive from and overhear another node. By using 2-hop coloring approach, the proposed system guarantees the following cases: *i.* all 1-hop neighbors are colored in different values; *ii.* the nodes that are not in 1-hop neighborhood but have the same parent are colored in different values; *iii.* a node and its grandparent node are colored in different values; *iv.* a node and the neighbors of its parent are all colored in different values.

Node sequencing co-layer arranges the sequences to operate for the nodes within 2-hop communication for their following four operations: sending raw data packet, forwarding a received packet, receiving a packet, overhearing a packet. Timer configuration co-layer configures the timers of the nodes for their four different operations. The configured times to send, forward, receive, overhear for a node is passed to MAC layer of the node. From then on, the MAC layer will run based on those time values. Rather than using the default MAC layer protocol, the proposed system uses its time based scheduling to enable collision free WSN design. The advantage of such a mechanism is that the infrastructure can be implemented on any kind of MAC layers, which means the architecture is independent from any specific MAC protocol since the system configures the send and receive times for the nodes.

In Fig. 1, the arrow from physical layer to multi-hop network construction layer refers to the information of the positions of the nodes. It may be assumed that the nodes know their locations and learn the locations of their neighbors. Received Signal Strength Indicator (RSSI) values obtained from the physical signal power strength of the packet received can be used for this objective. RSSI values are used in neighborhood decision. According to the signal strength of the received packets a node may predict the relatively distances of its one-hop neighbors. RSSI is neither a well-qualified approach to find exact distances between the nodes nor a popular method to find the precise ranges of the neighbors of a node; the state-of-the-art methods may cover more complex technologies for accurate results [9]. Such technologies may consume much more power than the RSSI mechanism. On the other side, using GPS devices may also assist on positioning with an extra cost. However, for a node in the proposed infrastructure, to discover the distances of 1-hop neighbors with little errors is adequate instead of finding the distances with high precision. A node just needs to consider whether it is a neighbor of another one; and if that is so, the distance to that node will be obtained using RSSI value. These neighborhood information and the distance values will be used in network construction layer to implement the system tree. If a detailed localization method is used, the locations instead of the distances in 1-hop neighborhood may also be sent.

The system is designed to construct a multi-hop tree at the beginning. The established tree will contain one sink, one main path for all nodes to access that sink, and possible redundant paths for all nodes in case of a recovery. The formation of backup paths overlaying the same network is taken care to help the nodes alter their paths to sink in case of security.

Regarding the attacker-aware and fault-tolerant design of the system, overhearing mechanism lies at bottom to detect the nodes compromised or broken. A node, who sent a packet to its parent in past, overhears its packet in correct time schedule while the packet is being forwarded from parent to grandparent. Hence, a node is always monitored by its child via overhearing of the related communication between the node and its parent. During the overhearing of the communication for the specified packet, the overheard packet is compared bit-by-bit with the related packet stored previously. If the packet forwarded from a parent of a node to the grandparent of the node is decided invalid by the node, a fault will be detected, and so will the faulty node. Of course the passive adversaries that only listen the network but not modify any packet cannot be detected. Whereas the active ones that modify or drop the packets can directly be detected. Such kind of detection is so quick. When an adversary injects a packet to the system, it may not be detected by its any child since the corresponding time may not be an overhearing time; nevertheless there may be a collision since that specific time slot could be reserved for another operation. In such a scenario, the problematic node may not be detected quickly, but the faulty area will be detected since the actual system avoids the collisions.

B. System Establishment

The layered design shown in Fig. 1 covers the following modules running for establishment phase, respectively: *constructTree()*, *coloring()*, and *sequencing()*.

It is assumed that there is no adversary in the network during the period that the proposed system is being established through these modules. When the system infrastructure is set on the whole WSN, the application layer may run any specific WSN scenario. During this establishment period the nodes of the system run according to the rules of the MAC layer protocol available on. After the establishment is completed and the system is set up on all nodes, the system architecture begins to run. After then, if a fault is detected the nodes will attempt to change their child-parent-grandparent relations and so they will try to use other backup paths to access to sink of the network.

The *constructTree()* module runs the procedure in all nodes to form a multipath tree structure in which all distributed nodes within the environment are connected and all nodes assign their parents on the way to sink. The procedure is started by the sink node that is predetermined in the system. All nodes are

homogenous in terms of their capabilities and resources; however the sink is assumed to be extremely powerful and is the gateway of the WSN to outside any IS.

While the tree is being constructed, there are also clusters formed in. In proposed system design, there are two main parameters predetermined and passed to sink before the tree construction begins. First parameter is about the tree structure, whether the tree from the given graph is formed in a breadth-first manner or not. The latter parameter is about the depth of the clusters to be formed within tree. It is considered that if the nodes are sparse distributed regarding their locations in the environment, the breadth-first tree or any tree structure may be formed in similar result. On the other side, if the density of the environment gets increased, i.e., dense graph, and the average neighborhood degree of the nodes gets increased as well, choosing a breadth-first tree formation could have an advantage on forming other tree structures regarding to the depth of the tree. According to the breadth-first tree formation the nodes will attempt to connect themselves to the nearest node to sink.

Regarding the finite state machine mechanism for tree formation together with clustering formation, all nodes at first learn their neighbors; the neighbor list for a node is filled with unique node identification (ID) numbers. ID of 0 refers to the sink node. Sink node broadcasts the first CONSTRUCT_TREE message. The following arguments are put in the CONSTRUCT_TREE message: the distance to sink value of the node that sends the message, the predetermined depth for a cluster, the cluster level, and ID of the related cluster's leader. The distance to sink value refers to the shortest hop number for a node to access to sink. According to the predetermined depth parameter for the clusters, the nodes will arrange the cluster levels. When a node receives CONSTRUCT_TREE message, it finds its cluster level, connects itself to the corresponding parent (choosing the tree formation as to be breadth-first), puts itself into either member or leader state for the corresponding cluster level and finally broadcasts CONSTRUCT_TREE message with its leader ID. Every node receives at least one CONSTRUCT_TREE message and sends exactly one CONSTRUCT_TREE message. Without counting the messages sent to fill the neighbor lists of the nodes, which is an assumption for this module, the module's message complexity in a sensor network with N nodes is the total number of the nodes from top to down in whole tree-network, O(N); the space and time complexities of the module running on each node is constant, O(1).

The purpose to color the nodes is mainly to prevent the collision of transmissions. By giving different colors to the nodes that could give rise to collisions in case of attempting to access the wireless media during the same time slot, these nodes will not be allowed to be active to transmit during the same times. The time is divided to multiple slots regarding to the colors, and each color is assigned with a different time slot. The nodes with same colors have no relation regarding to possible collisions and therefore each node does its operation in its color. Hence, each node uses a time schedule related with its color assigned.

The main objective of *coloring()* module is to give an appropriate color to each node in the tree. For this reason, the tree is defined as a graph G, in which the nodes are defined as the vertex entries V and the neighborhood relation of the nodes are defined as the edges E between the nodes. Rather than implementing a distributed coloring algorithm in each node, the central sink node is chosen to compute the coloring procedure itself. Each node sends their related information to sink to make the sink node gather the data and learn the graph G = (E, V). After the sink computes the coloring for the graph, i.e., the whole network, it sends back the color value assigned to each node. The *coloring()* module involves the three following sub-procedures, respectively: *sendNeighbors()*, *findColors_at_SINK()*, and *sendColors()*.

1) sendNeighbors() sub-procedure:

It is assumed that each node learns about their neighbors and fills in their neighbor lists before the tree construction begins. After the tree is constructed each node sends their neighbor list together with its maximum possible children list to its parent on the way to sink. Maximum possible children list involves not only the actual children list but also the probable children who are the candidates in case of a failure in the future. An exact child of a node is the one actually sends packets to the node; whereas a possible child of a node is the one that actually sends packets to another node but may change its parent and may adapt to its new parent to send packets in the future. A possible child of a node must be a neighbor of the node and also its height on the tree must be 1 hop far away to sink than the node's. The aim is to color them in different values to avoid collision. Not only node coloring but also the node sequencing is necessary to form a collision free and scheduled infrastructure.

Each node also forwards the neighbor lists received from its children to its parent. At this level the aggregation of the data before forwarding may be considered. However, regarding the set up time to be completed faster, the nodes may not store or process any list information and may directly forward the neighbor lists of their children to their parents. At the end, the sink gathers each neighbor list of each node. The message complexity in this procedure is related to the depth, d, of the tree. If the height of a node i, which is the hop distance of the node to the sink, is given as h_i, the neighbor list of that node is conveyed to sink in h_i retransmissions. The message transmissions for N nodes in the whole tree will be $\Sigma\ h_i$ for i=1,2,..,N-1, assuming that the node with the ID of 0 is the sink node. This total sum, which is the message complexity, can be at most O(N.d) where the depth of the tree is d.

Regarding the probable faults or failures, if a node detects either the case that its parent has a fault or that its parent is compromised, the node attempts to change its parent to a legitimate node and tries to survive its mission in the network. Within the proposed design, the recovery concept is based on each node's attitude that each node attempts to recover itself if possible. By this way, the message and time complexities for a recovery process for a node could be low and the exact values would be related to the number of the node's neighbors or the average neighborhood degree since the process is handled in a local region rather than the whole network. The proposed recovery concept is based on the predefined backup path

formations in main tree. If the constructed running tree structure needs to be changed, the node that detects the fault of its parent will change its actual parent and so will the path to sink. This mechanism requires for a node to know at the beginning phase all its possible parents and children as well. Alternative design may take account an on-demand reformation of the tree when a fault on the communication path is detected. However on-demand reformation may consume much more energy since the system needs to be reset up for the new parent and children information.

The worst-case space complexity to store the maximum possible children list or neighbor list is directly related to the maximum neighborhood degree in the network. However, it takes a storage of O(N), which is practically reserved in the nodes before deployment. For each node to compute the number of its maximum possible children is also related to the neighborhood degree of the node, that is the number of the iterations to search for the levels of the neighbors is the number of the neighbors for a node.

2) findColors_at_Sink() sub-procedure:

The sink constructs the whole graph G by gathering the neighbor lists from the nodes and fills the adjacency matrix. Regarding the overhearing concept, not only the child-parent relation but the parent-grandparent relation is considered as well. Due to child-parent-grandparent relations, a 2-hop coloring is taken care rather than a 1-hop coloring. Therefore, to compute the coloring algorithm all the nodes within 1-hop and 2-hop ranges are defined as adjacent in the adjacency matrix. At the end, no pair of nodes, which are able to communicate in 2-hop ranges, will be assigned with the same color.

Regarding the vertex coloring, the proposed system does not involve the distributed algorithms to color the graph. That is because the energy consumption is not to be raised by extra messaging for specifically solving the coloring issue. Since the input data required to centrally solve the coloring problem in the network has already gathered in sink node, which is powerful in resource, the coloring of the nodes are computed at sink without any extra messaging between the nodes.

The coloring algorithm computed centrally on sink node is the one suggested by Leighton [10] for practical vertex coloring applications. The algorithm is heuristic and based on Recursive Largest First (RLF) approach that the nodes with more neighbors are colored at first. RLF algorithm was investigated by Klotz [11] and its worst case time complexity was pointed out as to be $O(|V|^3)$. Furthermore, it is emphasized by Chiarandini et al. [12] that even the complexity of RLF heuristic algorithm is not the best, RLF results are practically well-qualified among the heuristic approaches that solves the graph k-coloring problem in polynomial time without assuring a specific approximation ratio.

3) sendColors() sub-procedure:

The sink broadcasts to all nodes the 2-hop coloring result of the whole graph as a list and the maximum number of the possible children in the network as well. The value of maximum number of the possible children in the network will also be used in node sequencing. Each node learns its color and the colors of the other nodes to communicate as well. Using this information, a node will send or forward during time slots assigned to its corresponding color and will receive or overhear during time slots assigned to the other colors of its possible children and possible parents. The nodes will be restricted to operate differently than the scheduled colors. By coloring, the children nodes are sequenced in time order to send their packets to the same parent, respectively, without any collision. On the other side, the parent stores these packets and since the parent sends only in its time slots assigned with its color, the parent needs to be sequenced for the time slots to forward the stored packets of its children with no collision. Therefore, the time division based scheduling needs not only coloring but sequencing as well.

The *sequencing()* module is the last one to complete the time table of the nodes to send/forward and receive/overhear. The final scheduled time slots will be in such a format as given in Fig. 2.

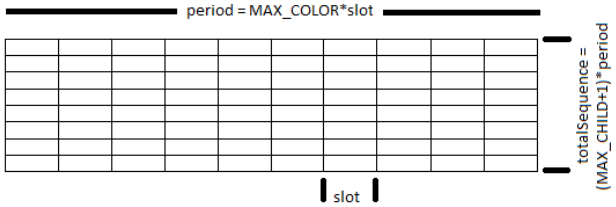

Fig. 2. Timetable of the system after coloring and sequencing.

In Fig. 2, the colors are related mainly to the column based division of the time table. Each column has a different color assigned to and during each slot in each column only the same colored nodes, which will not lead to any collision as they are at least three-hop far away between each other, may send/forward the data. The rows in Fig. 2 are arranged by the *sequencing()* module of the system. MAX_CHILD refers to the maximum possible children number in the network. In each row a node will either send its raw packet or forward a stored packet previously received from its children. Let each slot in first row refer to the time period of the send operation for a node, then the next rows will be the forward operations for its children's packet. As each next row after the first one is dedicated to a different possible child data, the value of the number of the rows in timetable will be assigned with the maximum possible children number plus one.

A *slot* in Fig. 2 is the total time slot required for a successful communication of a data packet; a sender processes its data from application layer into a physical layer packet and successfully transmits the packet on the medium, then the receiver receives the packet and finally processes it. A *period* in Fig. 2 is the total time for all nodes in different colors to transmit their packets successfully. A period time is equal to one slot unit times the number of the colors. So, the whole timetable requires the amount of time, referred as *totalSequence* in Fig. 2, which is equal to the multiplication result of total row size with a period time.

A parent either sends its raw packet or forwards the previously received packets from its children. Each send/forward operation will be done in the same column. To arrange the order of forwarding the packets from its children to its parent each node runs a sequencing procedure. The packet

of the child with the lowest ID will be forwarded first as to make the sequence simple. The nodes that has at least one possible child sends a CHILD_SEQUENCE message to all its possible children and the children arranges their row sequences of the timetable in the same approach. Each child will receive a different CHILD_SEQUENCE message from its each different possible parent. If any failure is detected by a node, the node may change its parent according to backup path supported tree structure. Since the same sequence information of the node changing its path is stored on all possible parents and all possible children of those parents there will be no conflict for the system to switch between any of the backup paths of tree.

Each parent broadcasts the CHILD_SEQUENCE message to its all possible children. In this case, the message complexity of the *sequencing()* module for the whole network is O(N). Each node runs the sequence computation for each possible parent of it. In this case, the time complexity of the module is the number of maximum possible parents in the network. Each node also stores the possible children list of itself.

The same scheduling timetable is used in all nodes as to offer a collision free communication infrastructure overall the network. Regarding the timetable in Fig. 2, the time practically passes from left to right in row-base; however the beginning column may be selected for any color value. After a *period* ends, the time continues with the next *period* in the next row. After the time amount of a whole *totalSequence* is passed, the whole communication will keep on going with repeating the same scheduled timetables. The network's time synchronization is an assumption in this work and out of scope of this paper.

The proposed infrastructure is simulated successfully in ns-2.34 [13] for a WSN with 50 nodes assuming that all nodes have packets to send forever. Using the timer configuration implementation, each node set its *sendTimer* (to send/forward), *receiveTimer* (to listen) and *overhearTimer* (to overhear) using its timetable values after coloring and sequencing. It is seen in simulations that the timers expired on the appropriate moments for each node to operate without any collision for pre-dedicated operations in the corresponding slots. During the time period between any two consecutive operating slots for a node, the node gets into *idle* state and sleeps to preserve its remaining power with the aim of prolonging the lifetime.

III. Conclusion

In this paper, a design of a 2-hop coloring-based, collision free and reliable communication infrastructure is proposed. To prevent the whole network from extra energy consumption of re-transmission of conflicted packets, this infrastructure avoids the collisions caused by simultaneous transmissions from any nodes in same colors. The infrastructure also enables simultaneous conflict-free transmissions from the nodes with same color, which not only reserves the energy but also makes the communication schedule more efficient than a ring-based ordered schedule in terms of delay. With the help of multipath redundancy, a node may exclude its broken parent from its route to sink and may adapt to its new parent to establish a new reliable communication path. Rather than arranging the recovery paths after a failure, establishing backup recovery paths for each node at the beginning while constructing the main multi-hop tree not only reserves total energy of the network but quickens the recovery process as well.

With a cost of delay in communication, to overhear a parent's transmission leads to an energy-efficient monitoring approach without any message conflict. This monitoring also brings the advantages of perceiving any attempt of compromise between child-parent-grandparent paths and adapting to new backup paths for the leaves of a parent in suspect.

The future work will be to focus on the implementation of this infrastructure in a WSN with realistic packet traffics.

Acknowledgement

The authors thank to Dr. Kayhan Erciyes and Dr. Albert Levi for their help and comments on the design proposed.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E Cayirci, "Wireless sensor networks: a survey", Computer Networks, vol. 38(4), pp. 393-422, March 2002.

[2] A.E. Kouche, "Towards a wireless sensor network platform for the internet of things: Sprouts WSN platform", Proc. of IEEE ICC 2012 Ad-hoc and Sensor Networking Symposium, pp. 632-636, 2012.

[3] K. Doddapaneni et al., "A model-driven engineering framework for architecting and analysing wireless sensor networks", Proc. of Third International Workshop on Software Engineering for Sensor Network Applications (SESENA), pp. 1-7, 2012.

[4] J. Wu, M. Dong, K. Ota, M. Tariq, and L. Guo, "Cross-domain fine-grained data usage control service for industrial wireless sensor networks", IEEE Access, vol. 3, pp. 2939-2949, December 2015.

[5] L. Dariz, M. Selvatici, M. Ruggeri, and R. Abrishambaf, "Smart and wearable wireless sensors: scenario analysis and communication issues", Proc. of IEEE International Conference on Industrial Technology (ICIT), pp. 1938-1943, 2016.

[6] V.N. Satyanarayana and A.B. Beneyaz, "Smart WSN-based ubiquitous architecture for smart cities", Proc. of International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2366-2373, 2015.

[7] K.G. Srinivasa, N. Siddiqui, and A Kumar, "ParaSense - A sensor integrated cloud based internet of things prototype for real time monitoring applications", IEEE Region 10 Symposium (TENSYMP), pp. 53-57, 2015.

[8] I. Korkmaz, O. Dagdeviren, F. Tekbacak, and M.E. Dalkilic, "A survey on security in wireless sensor networks: attacks and defense mechanisms", book chapter in Theory and Practice of Cryptography Solutions for Secure Information Systems (ed. A. Elci), pp. 223-251, IGI Global, 2013.

[9] F. Sun, H. Yin, and W. Wang, "Bounds on performance of UWB TOA estimation using finite resolution quantization", IEEE Wireless Communications and Networking Conference (WCNC), pp. 2620-2625, 2013.

[10] F.T. Leighton, "A graph coloring algorithm for large scheduling problems", Journal of Research of the National Bureau of Standards, vol. 84, no. 6, pp. 489-506, 1979.

[11] W. Klotz, "Graph coloring algorithms", Mathematik-Bericht 5, TU Clausthal, pp. 1-9, 2002.

[12] M. Chiarandini, G. Galbiati, and S. Gualandi, "Efficiency issues in the RLF heuristic for graph coloring" Proc. of 9th Metaheuristics International Conference (MIC), pp. 461-469, 2011.

[13] The Newtwork Simulator – ns-2, http://www.isi.edu/nsnam/ns/ (accessed 5 Juy 2016).