Implementation of a Routing Protocol for Ad Hoc Networks in Search and Rescue Robotics

Filipe Araujo, José Santos
CISUC, Department of Informatics Engineering
University of Coimbra, Portugal
filipius@uc.pt, jsmp@student.dei.uc.pt

Rui P. Rocha
Institute of Systems and Robotics (ISR)
Dept. of Electrical and Computer Engineering
University of Coimbra, Portugal, rprocha@isr.uc.pt

***Abstract*—As robotics evolves, we expect to see increasing numbers of search and rescue teams (SARTs) of humans and autonomous mobile robots, especially in hazardous scenarios. Under these conditions, a third party infrastructure network might be unavailable, forcing the agents' mobile devices to self-organize in a wireless ad hoc network. Search and rescue actions pose specific challenges to these networks, as they need large communication bandwidths, and, at the same time, a very tight connection to the Command Center (CC). Hence, SART ad hoc networks are neither typical mobile ad hoc networks (MANETs), nor wireless sensor networks with firm energy restrictions.**

To respond to these requirements, we propose a routing protocol, called CHOPIN, that defines a tree rooted at the CC, while also allowing flooding within teams of nodes. We conceived, implemented and integrated this protocol in the Robot Operating System (ROS), using simple but proven methods, and open standards for messages. The field benchmarking we did against the Optimized Link State Routing Protocol (OLSR) shows the ability of CHOPIN to assist SART agents in their communication.

I. Introduction

The use of mobile devices has grown at a very fast rate for the past two decades. Advances in computing and communication have greatly increased the spectrum of applications of these devices. The ability to dynamically form a network to support collaborative work is increasingly important for military operations, environmental monitoring, or vehicle communication. In this paper, we focus on emergency scenarios, where human and robot teams collaborate under the control of a Command Center (CC) for search and rescue operations, building a map of the incident zone, sensing hazardous materials or conditions, localizing victims, and finding points of interest [24]. These tasks involve multiple types of communication, either among agents, but also to and from the CC. Mobile robot teams can contribute towards improving the efficiency and effectiveness of operations in emergency scenarios. They are presented as teams of mobile agents operating with distributed communication capacity, to reduce the risk of human agents and to automate some tasks.

To be effective, even in the worst scenarios, search and rescue teams (SARTs) must not depend on the existence of a high-bandwidth network infrastructure for their operations. While this leaves mobile ad hoc networks (MANETs) as the best option for the operational scenarios, the patterns of communication involved require specific solutions departing from previous ad hoc routing protocols. The leading role of the CC is a distinctive feature of a SART, as it must be the greater source of commands and the sink of most information collected. Arbitrary point-to-point communication, as supported in typical Mobile Ad Hoc Networks (MANETs) is simply not needed. On the other hand, agents, either human or robotic, must coordinate and have access to large bandwidth communication within their teams, to move in coordination and to build maps of their surroundings. This separates SARTs from wireless sensor networks.

To create and maintain a dynamic communication infrastructure, we propose the CHOPIN[1] routing protocol. Since we need to build a single tree rooted at the CC, we manage to escape the dilemma of proactive (more traffic) versus reactive (slower on first delivery) protocols. Regular nodes can keep their single-destination routing table always updated, whereas the CC can learn the entire topology with limited effort — proactivity does not come at a big cost.

We implemented CHOPIN in a real setting for the Robot Operating System framework [4], and benchmarked it against the very well-known OLSR protocol [10]. We made our implementation publicly available on GitHub[2]. The results we achieved are quite promising. We can say that by being designed for a very focused problem, CHOPIN is extremely simple and lightweight, being able to outperform OLSR in most practical tasks for the network sizes we measured. This, we believe, shows that CHOPIN is quite fit for SARTs comprising both human and robotic agents.

The rest of the paper is organized as follows. In Section II we overview related work. In Section III we design the CHOPIN protocol. In Section IV we describe the implementation. In Section V we evaluate and discuss the implementation and we conclude the paper in Section VI.

II. Related Work

A. Routing Protocols for Wireless Ad Hoc Networks

A MANET is an infrastructure-less network of mobile nodes that self-organize to communicate using wireless interfaces. MANETs might be particularly helpful in disaster scenarios where the infrastructure is usually absent or not working

[1]Cooperation between Human and rObotic teams in catastroPhic INcidents (http://chopin.isr.uc.pt).

[2]https://github.com/jsmpereira/chopin-routing.

978-1-4799-6606-6/14/$31.00 ©2014 IEEE

properly. Wireless sensor networks are similar to MANETs, but nodes do not need to be mobile. Moreover, communication is usually directed to one or more sinks [11], [26], while saving energy is extremely important to minimize maintenance [2]. These requirements promote the adoption of communication protocols like ZigBee [15] that trade bandwidth for energy.

The effort of properly classifying and organizing wireless ad hoc routing solutions is massive, given the solutions involved. We do not start such effort here, because it was done before [23], [1]. Nevertheless, it is worthwhile pointing out the classification of Dua *et al.* in the context of vehicular networks [13]. They divide routing protocols into geographic, topology-based, hybrid, clustered, and data-fusion. Geographic-based protocols [25], [3] use the location of nodes, typically the destination and the current forwarding node, to select the next hop of the path. These protocols use a very small amount of memory, as they only keep information of their surroundings. However, in our case, networks have moderate size, so we can opt for topology-based approaches, which develop non-localized notions of the topology. A particular geographic protocol similar to our own is TREBOL [17]. Like CHOPIN, TREBOL also builds a tree, but, since it uses geographic information, nodes, including human agents, must carry devices to determine their location.

Clustered algorithms are not very different from what we do, although in a different way: we assume that nodes belong to teams, thus being explicitly clustered. Inside these restricted teams, nodes communicate by flooding. It is also inside these teams that nodes exchange the largest amounts of data, *e.g.*, to create maps of the scenes. Data-fusion algorithms are appropriate to merge information, possibly on the way to the sink [19], but they are of little interest to us, because nodes consolidate their maps inside the teams they belong to. Hence, we adopt a topology-based protocol, although, as we shall see, in a way that is considerably dissimilar from more common MANETs.

A main source of classification for wireless ad hoc routing protocols is whether they are reactive or proactive. The latter ones bear some similarities to wired protocols, because nodes actively collect information of the network and keep routing tables for each possible destination. In the former protocols, nodes only collect paths to specific destinations as needed. This approach conserves bandwidth and energy, but imposes larger delays on routing. AODV [21] and DSR [18] are very well-know examples of reactive protocols, but we can find a very large number of these protocols in the literature [12], [5]. DSDV [22], Babel [8] and OLSR [10] are two very well-known protocols that build routing tables prior to the routing function. In SART networks, the dilemma between reactive or proactive is less important. Since we only have one remote destination, the CC, using a proactive approach is not too expensive, while having the network ready to send reduces latency. In fact, we can create a tree proactively, as we see in many wireless sensor networks. The main difference is that, unlike wireless sensor networks, nodes do not need to engage in periodically shutdown of radios, or any other aggressive scheme, to conserve energy [2].

We could not find much work in the literature concerning the need to create wireless ad hoc networks in SART with robots. We mostly found starting research projects and position documents. In particular, we could not find the specific requirements of SART identified in any of these documents. We are not aware of any other work that has tackled the specific requirements of a SART team made of human and robotic agents in a way so detailed as ours. The other papers we mention, consider the adaptation of generic routing protocols with point-to-point capabilities that do not seem particularly useful for disaster scenarios. Overall, we realized that existing MANETs are very generic, not suiting the specific needs of the scenarios we tackle. Wireless sensor networks are also not exactly up to search and rescue, as they mostly focus on collecting data (without any maps) and saving energy. Hence, we chose to develop an ad hoc network that fits our specific needs.

III. Design of the Routing Protocol

A. CHOPIN Requirements

Most data will travel from human and robotic agents into the CC, whereas communication in the reverse direction generally takes the form of commands. We assume the existence of teams of robots that, according to their responsibilities, search points of interest, build maps, and collect sensory data for the CC, after reconciling information within the team. Arbitrary point-to-point communication does not seem important for SART. Overall, we identified the following requirements:

1) All nodes have routing responsibilities;
2) The CC communicates with individual nodes or with all nodes;
3) Individual nodes communicate with the CC;
4) Individual nodes communicate inside the same team;
5) Nodes know to which team they belong.

From this list of requirements, we can see that none of the big families of protocols is entirely suited for SARTs.

B. Network Creation (the Tree)

In the CHOPIN routing protocol, we consider three main forms of communication: 1) flooding within the boundaries of a team, initiated by any node; 2) flooding initiated by the CC; and 3) point-to-point communication from particular nodes to the CC, and vice-versa. To ensure communication with the CC, in cases 2 and 3, we create a tree rooted there.

We define two types of messages to build the tree: the `Base Station Beacon` and the `Node Beacon`. As names say, the first type of message comes from the CC[3]. Nodes spread information of where the CC is, by flooding these beacons. These messages carry the hops they travelled so far and a version number, thus being very lightweight. Nodes will rebroadcast the message if its version number is higher, or if it announces a better distance. This ensures that the `Base`

[3]We adopted the name `Base Station Beacon` because the CC is similar to a Base Station.

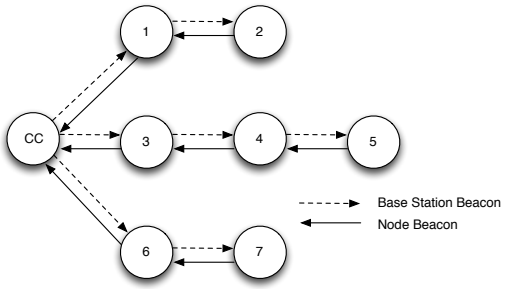

Fig. 1. The flow of CHOPIN messages.

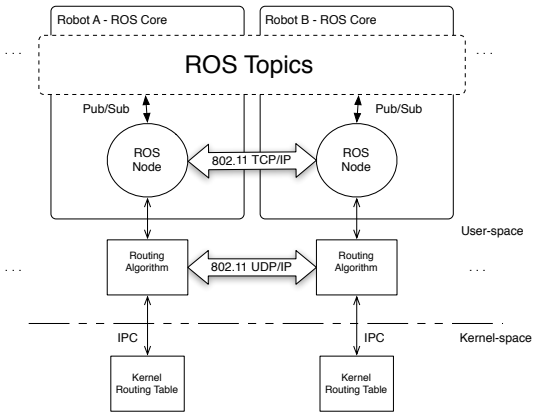

Fig. 2. The architecture of CHOPIN nodes.

`Station Beacon` reaches the entire network. To send a message to the CC, nodes must use the reverse path, *i.e.*, they use the node that announced the shortest path (for the latest version number).

This mechanism ensures that all nodes know the next hop in direction to the CC. Communication in the reverse direction is necessary to ensure that the CC can send commands to specific nodes. To inform the CC of their presence, the other nodes use `Node Beacon` messages. On retransmission of the `Node Beacon`s, nodes attach their own addresses to the message. This gives the CC complete information of topology. The `Node Beacon` messages also provide neighborhood information to the nodes. We show the flow of these messages in Figure 1. Although we do not ensure the bidirectionally of links in this version of the protocol, these messages can help to ensure that nodes only use bidirectional links to reach the CC[4]. However, we did not find this possibility very important, because nodes are expected to move frequently, thus making quick changes to the connectivity of the network. This will overcome possible blind spots originated by the lack of bidirectionally.

C. Routing Components

One way of implementing routing protocols is to divide their functionality into user space, to maintain routing information and compute paths, and into kernel address space, to forward packets accordingly. Running the processing in user space simplifies deployment and reduces kernel complexity, at the cost of requiring communication between privileged and non-privileged spaces. Based on this idea and following an OLSR implementation[5], we keep a program running as a service of the operating system, to exchange information with the network neighborhood and compute routes, before populating the kernel routing table.

To exchange routing information between peer nodes, we opted for the UDP protocol. UDP is connectionless and best-effort, thus lightweight. CHOPIN receives UDP messages on port 269 [7] and floods messages using the network broadcast address. One should notice that the CHOPIN protocol does not interfere with other application-level services. These may still use the TCP/IP stack and the 802.11 wireless network to communicate. However, since the network is ad hoc, the possibility of delivering messages depends on the presence (and accessibility) of the destination in the kernel routing table.

D. The Application Layer

To map the environment and gather sensor data, robotic agents take full advantage of the ROS framework [4]. The communication between different agents occurs through ROS topics, in a publish/subscriber model [14]. To exchange information between different robotic agents we used the `foreign_relay` [16] and the `wifi_comm` [6] packages. Figure 2 shows the overall architecture of robotic agents making use of the ROS platform for distributed communication. The "ROS Topics" field represents the group of topics shared between the nodes of the network, through the `foreign_relay` package. The routing protocol provides the knowledge of the neighborhood and network infrastructure, which are necessary for remote authoring in topics.

IV. Implementation

We developed the CHOPIN protocol using the Common Lisp language, more specifically the implementation Steel Bank Common Lisp (SBCL)[6]. Our implementation runs on the Ubuntu operating system, Linux Kernel version 3.9. Our implementation follows a generic message format for mobile ad hoc networks [9]. To take care of the messages, CHOPIN runs two main threads. One for reading and processing, the other for sending the messages. Through a configuration file, one can set some parameters, like the network interface, the IP address, the broadcast address of the network or the port.

A. Data Structures

B. Messages

The protocol maintains information in three main structures: the `DUPLICATE-SET`, the `LINK-SET`, and the

[4]If nodes announce the list of their neighbors, they can determine whether their neighbors hear them.

[5]http://www.olsr.org/?q=download.

[6]http://www.sbcl.org.

```
<packet> := <pkt-header>
            <message>*

<pkt-header> := <version>
                <pkt-flags>
                <pkt-seq-num>?
                <tlv-block>?

<message> := <msg-header>
             <tlv-block>
             (<addr-block><tlv-blocks>)*

<msg-header> := <msg-type>
                <msg-flags>
                <msg-addr-length>
                <msg-size>
                <msg-orig-addr>?
                <msg-hop-limit>?
                <msg-hop-count>?
                <msg-seq-num>?

<tlv-block> := <tlvs-length>
               <tlv>*

<tlv> := <tlv-type>
         <tlv-flags>
         <tlv-type-ext>?
         (<index-start><index-stop>?)?
         (<length><value>?)?
```

Fig. 3. Structure of a packet in RFC 5444.

TABLE I
STRUCTURE OF AN ENTRY IN THE HASH TABLE `DUPLICATE-SET`.

orig-addr	msg-type	seq-num	exp-time

TABLE II
STRUCTURE OF AN ENTRY IN THE `LINK-SET` HASH TABLE.

local-addr	neighbor-addr	l-time

TABLE III
STRUCTURE OF AN ENTRY IN THE `ROUTING-TABLE` HASH TABLE.

destination	next-hop	hop-count

`ROUTING-TABLE`. For efficiency, we use dictionaries (hash tables) to implement these structures.

The `DUPLICATE-SET` keeps track of received messages, to avoid processing duplicates. It maintains the following fields (see Table I): the originating address, the type of message, and the sequence number. These also serve as the key to access data, whereas the expiry time determines when to delete the entries. As in OLSR, the `LINK-SET` keeps associations between local interfaces and remote addresses of nodes (neighbors and CC). This information has a limited duration in time. We show the format of this structure in Table II. The `ROUTING-TABLE` structure, in Table III, keeps the next hop to reach a given destination (if available). The key to access this storage is the destination. CHOPIN first stores paths in this table, before copying them to the kernel routing table. This interaction between kernel and user space takes place through the `NETLINK` socket family.

A fourth structure, the `OUT-BUFFER`, stores messages waiting to be written to the socket. `OUT-BUFFER` is a first-in-first-out (FIFO) queue, where messages await to enter the UDP socket. The two threads of CHOPIN share the access to this queue: one reads messages to send them to the socket, the other writes messages related to the protocol, often in response to timers. This latter thread is also responsible for receiving the messages from peers. We use a semaphore to synchronize the threads.

We followed the Request For Comments (RFC) 5444 [9] for the message format. The adoption of RFC 5444 brings several advantages, including ease of processing, extensibility and the ability to reduce the number and size of packets, due to the inclusion of various messages and a compact representation of addresses. The design is also suited to enable certain decisions based on partial header information, to reduce the processing effort. The basic structure of a packet that conforms to the RFC 5444 specification is shown in Figure 3.

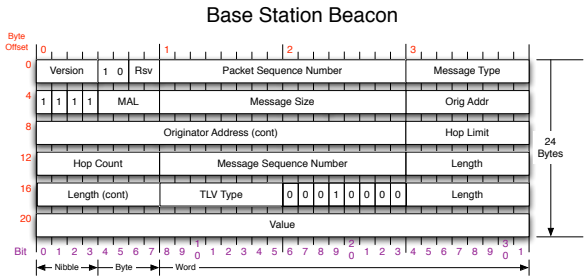

Fig. 4. Structure of a `Base Station Beacon` message according to the RFC 5444 specification.

In this figure, the $\star$ means zero or more times, the `?` means optional. It is worth mentioning the concept of *type-length-value* (TLV) and TLV block. A block aggregates zero or more TLVs associated with a packet, message or address block. In the current implementation, only the message has an associated `tlv-block`. The first field indicates the size of the TLVs included in the block. The adopted TLV uses the field for the type — `tlv-type`. The parameters specified in `tlv-flags` indicate how to interpret the following fields. In our case, they tell that the TLV comprises a *length* field and a *value* field.

1) The `Base Station Beacon`: `Base Station Beacons` (Figure 4) have a two-second interval. Its TLV block transports as a single attribute the IP address of the Base Station. As the message is broadcast over the network and through intermediate nodes, the content of this TLV is changed to the IP of the node forwarding the message. The field with the originator address never changes and always displays the message source. When the message goes through an intermediate node 1) the TLV value is changed to the current IP node, 2) the hop-count is incremented and 3) the hop-limit is decremented. Each intermediate node adds or updates the entry for the Base Station in its routing table, indicating the value it finds in the TLV block as the next hop (see Figure 5).

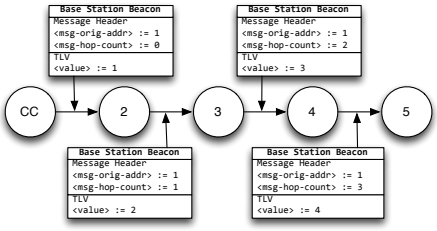

Fig. 5. Example of the contents of a message `Base Station Beacon` during its path.

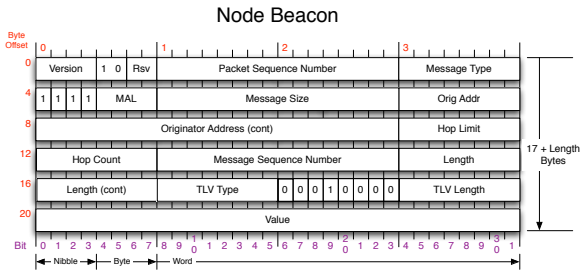

Fig. 6. Structure of a `Node Beacon` message, according to the RFC 5444 specification.

2) The `Node Beacon`: The `Node Beacon` message initially carries the address of the originator node. Along its reverse path to the Base Station, it accumulates the addresses of intermediate nodes. This lets the CC determine the entire network topology. Intermediate nodes change this message as follows (refer to Figures 6 and 7): 1) the IP address of the current node is added to the value of the TLV; 2) the hop-count is incremented; and 3) the hop-limit is decremented.

When a message is considered for processing, the `LINK-SET` table is updated. If there is already an entry in the table for the node that originated the message, its lifetime is extended; otherwise, a new entry is created. Similarly, an entry is added in the table `DUPLICATE-SET` with the structure shown in Table I. Then, the message can be written in the output buffer for forwarding. One should notice that the CC does not forward messages.

C. Timers

CHOPIN needs timers to send periodical messages and to verify whether routing information on its tables is out of date. We used the following timers and assign them the following default values: 1) `new-beacon`, with an interval of $2s$, serves to generate a new (`Base Station`) `Beacon Message`; 2) `check-duplicate-holding`, with a $30s$ interval serves to delete old entries from the `DUPLICATE-SET` key-value store; and 3) `check-link-set-validity`, with three times the interval of `new-beacon` serves to delete old entries from the `LINK-SET` key-value store. Given that OLSR is a protocol with several years of development and widely tested, whenever possible we adopted similar values for our own timers. Nevertheless, we let users of the protocol specify different values for the timers using a configuration file.

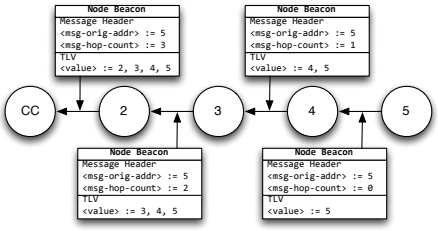

Fig. 7. Example of the contents of a message `Node Beacon` during its path.

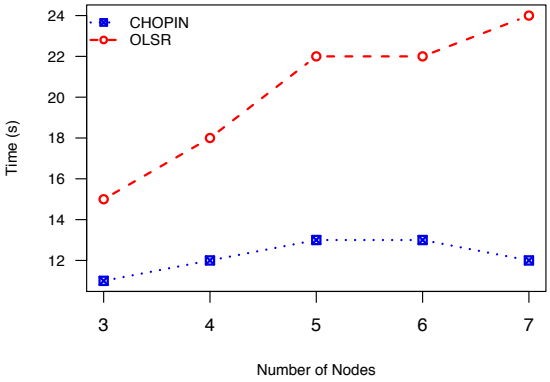

Fig. 10. Convergence speed in the Test Scenario 1 for 3 to 7 nodes.

V. Experimental Evaluation

In our evaluation, we compare OLSR and CHOPIN using real nodes. The tests measure the convergence time of the network, the throughput, and the number of messages used in networks of up to 7 nodes. These nodes were laptop computers with the Ubuntu operating system, WiFi 802.11 network interfaces, and static IP addresses. We ran the CHOPIN and the OLSR protocol using the 802.11 ad hoc mode of operation, where one participant creates a network in ad hoc mode and the remaining nodes join in.

We used the `sysstat` tool[7] to gather data for the evaluation. This set of tools provides system statistics, by querying information from the Linux `proc` file system. We used the `sar` tool, available on `sysstat`, to collect a large spectrum of data from the network, including UDP, TCP and errors. We also used other tools, like `ping`, to send ICMP ECHO requests and Secure Copy (SCP), to transfer files between nodes.

A. Topology Generation

We considered the topologies of Figures 8 and 9 for our field tests. In Topology 1, we have up to 7 nodes and 6 hops, while in Topology 2 we consider an unstable path, where the last node jumps from one branch to the next. Since all nodes were in the range of each other, we used the `Iptables` application to enforce the desired topologies.

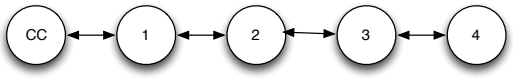

Fig. 8. Topology 1.

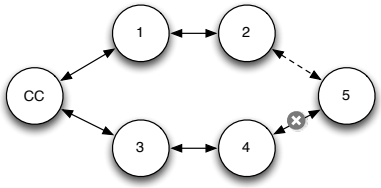

Fig. 9. Topology 2.

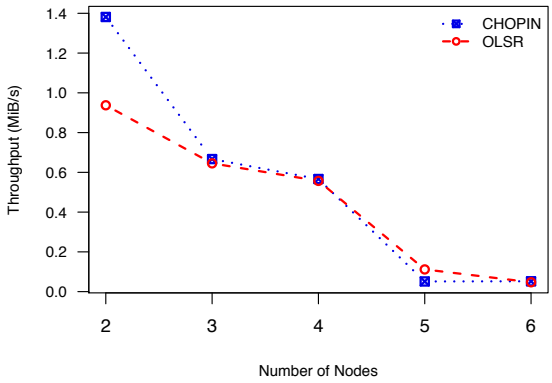

Fig. 11. Throughput in MiB/s from 2 to 6 nodes.

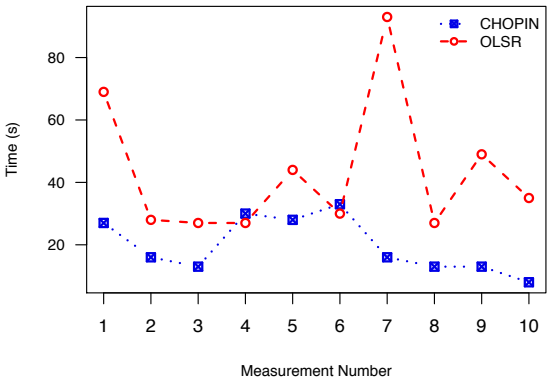

Fig. 12. Convergence time with forwarding in s.

B. Convergence Speed

In this test, we observe the time since node starts running CHOPIN until the information in the protocol data structures stabilizes. We used `Iptables` rules and the `ping` tool. As soon as the protocol converges, and a multi-hop path is established to the destination node, the CC starts receiving responses to its `ping` requests. The process was repeated for an increasing number of nodes, reaching a maximum of 7. In Figure 10, we show the convergence times for different network sizes. We can see that the CHOPIN protocol converges faster (we show averages of five measurements). The main cause for this increased speed is simplicity. The CHOPIN protocol maintains only three data structures, more specifically hash tables. Unlike this, OLSR requires eight repositories of information and needs additional processing for the MPR functionality. Furthermore, it requires a shortest path algorithm to calculate the paths.

C. Throughput

We must also consider the bandwidth available to applications. The throughput test puts load on the network and checks the average transfer rate achieved. We performed ten transfers of a 50 MiB file from one to three nodes, using the Secure Copy (SCP) program. From four to six nodes, nodes transfer a file of 6 MiB. The transfer is initiated on the first node and sent to the farthest node of the network. We kept using Topology 1 in this test. Note that the information for six nodes has to go through four intermediate nodes to reach its destination. In Figure 11, we can observe that the protocols reach a similar level of utilization of the network (we show average values for the 10 transfers). As the number of nodes increases, throughput decreases due to multi-hop operation and interference.

[7]http://sebastien.godard.pagesperso-orange.fr/.

D. Convergence Time with Topology Changes

In addition to the characteristics of the environment, mobility will often cause communication failures and reconstruction of paths. We use Topology 2 to evaluate this case. The path between node marked with 5 in the figure, sometimes goes through the lower branch (node 4), other times through the upper one (node 2). During our tests, we initiated a transfer of a 700 MiB file from the CC to node 5. This transfer puts the network under load for a considerable amount of time. During this time, at intervals of 20 to 40s, we change the `Iptables` rules to switch topology. In Figure 12, we can observe that OLSR shows a slower convergence than CHOPIN. Again, this is due to the higher complexity of OLSR, because OLSR must recompute and rebroadcast the entire topology on every change. In particular, mobility may force a recalculation of the MPRs, with the corresponding impact on convergence time. Unlike this, the CHOPIN protocol shows a faster convergence because it needs fewer communication interactions.

E. Number of Messages

In the same experiment, we measured the number of transmitted and received UDP datagrams, *i.e.*, the control messages exchanged by the protocols. In Figure 13, we can see that OLSR uses about 50% fewer control messages. This result is due to the optimization of resources promoted by the MPRs,

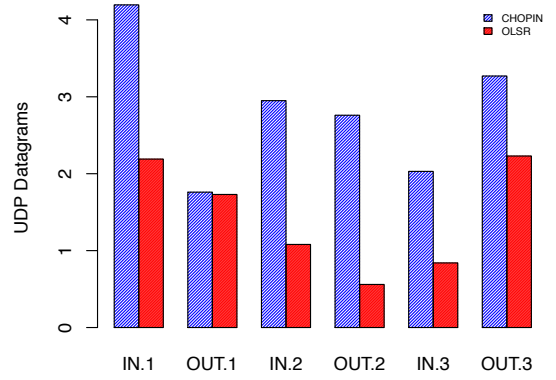

Fig. 13. Average number of UDP datagrams sent/received for 3 nodes.

whereas the CHOPIN protocol, so far, does not have any comparable approach. In a future version, we intend to adopt simple techniques to improve this result without significantly changing the protocol. For example, Lim and Kim [20] explore 1-hop and 2-hop neighborhood information to drastically cut network traffic (packet forwarding and arrival). In fact, this is pretty much what OLSR does with the MPRs. Their results show that savings in the range 1.5~10 times are possible.

VI. Conclusions and Future Directions

This document proposes the CHOPIN protocol to maintain an operational ad hoc network in search and rescue scenarios. Given the list of requirements for these scenarios, we found that existing MANET solutions, like OLSR, presented generic approaches that were not entirely fit to our needs. The same could be said about wireless sensor networks, which focus on energy consumption, while we need high bandwidth communication to transport images and robotic maps from the hazard scenes. The resulting CHOPIN protocol is conceptually simple, taking inspiration from multiple sources currently available on the field. While the design is inspired on the trees of wireless sensor networks, for the implementation we followed the best practices of OLSR.

Compared to the much more mature OLSR, we achieved very promising results and some interesting perspectives for future developments. CHOPIN was able to beat OLSR in most metrics we evaluated. However, one of the points for future improvement is the reduction of signaling costs, e.g., by using MPRs, as these may bring significant savings.

Acknowledgment

This work has been supported by the CHOPIN research project (PTDC/EEA-CRO/119000/2010) funded by “Fundação para a Ciência e a Tecnologia” (FCT).

References

[1] E. Alotaibi and B. Mukherjee. A survey on routing algorithms for wireless ad-hoc and mesh networks. *Computer Networks*, 56(2):940 – 965, 2012.

[2] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537 – 568, 2009.

[3] F. Araujo, L. Rodrigues, J. Kaiser, C. Liu, and C. Mitidieri. Chr: A distributed hash table for wireless ad hoc networks. In *Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS) (ICDCSW'05) - Volume 04*, ICDCSW '05, pages 407–413, Washington, DC, USA, 2005. IEEE Computer Society.

[4] R. Barraquand and A. Negre. The ROS Framework at a Glance, 2012. From online slides available at http://barraq.github.io/fOSSa2012/slides.html.

[5] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, DIALM '99, pages 48–55, New York, NY, USA, 1999. ACM.

[6] G. Cabrita and P. Sousa. wifi_comm - ROS Wiki., 2011.

[7] I. Chakeres. IANA Allocations for Mobile Ad Hoc Network (MANET) Protocols. RFC 5498 (Proposed Standard), Mar. 2009.

[8] J. Chroboczek. The Babel Routing Protocol. RFC 6126 (Experimental), Apr. 2011.

[9] T. Clausen, C. Dearlove, J. Dean, and C. Adjih. Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format. RFC 5444 (Proposed Standard), Feb. 2009.

[10] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[11] D. Culler, D. Estrin, and M. Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, 37(8):41–49, Aug 2004.

[12] D. Djenouri and N. Badache. On eliminating packet droppers in manet: A modular solution. *Ad Hoc Networks*, 7(6):1243 – 1258, 2009.

[13] A. Dua, N. Kumar, and S. Bawa. A systematic review on routing protocols for vehicular ad hoc networks. *Vehicular Communications*, 1(1):33 – 52, 2014.

[14] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, June 2003.

[15] S. Farahani. *ZigBee Wireless Networks and Transceivers*. Newnes, Newton, MA, USA, 2008.

[16] B. Gassend. foreign_relay - ROS Wiki., 2012.

[17] M. Gramaglia, M. Calderon, and C. J. Bernardos. Trebol: Tree-based routing and address autoconfiguration for vehicle-to-internet communications. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1–5, May 2011.

[18] D. B. Johnson, D. A. Maltz, and J. Broch. Ad hoc networking. chapter DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks, pages 139–172. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[19] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28 – 44, 2013.

[20] H. Lim and C. Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '00, pages 61–68, New York, NY, USA, 2000. ACM.

[21] C. Perkins, E. Royer, and S. Das. RFC 3561 Ad hoc On-Demand Distance Vector (AODV) Routing. Technical report, 2003.

[22] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, SIGCOMM '94, pages 234–244, New York, NY, USA, 1994. ACM.

[23] R. Rajaraman. Topology control and routing in ad hoc networks: A survey. *SIGACT News*, 33(2):60–73, June 2002.

[24] R. Rocha, D. Portugal, M. Couceiro, F. Araujo, P. Menezes, and J. Lobo. The CHOPIN project: Cooperation between Human and rObotic teams in catastroPhic INcidents. In *11th IEEE International Symposium on Safety, Security, and Rescue Robotics*, Linköping, Sweden, October 2013.

[25] I. Stojmenovic. Position-based routing in ad hoc networks. *Comm. Mag.*, 40(7):128–134, July 2002.

[26] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, Aug. 2008.