Wireless Communication Data Link Protocol for Cooperating and Tele-operated Mobile Robots

Mónica Pérez Vernet*

Lehrstuhl für Informatik VII, University of Würzburg, Germany

***Abstract*—This paper highlights the main features of DECOH-WIP an innovative wireless Data Link protocol designed for cooperating and tele-operated robots applications. It provides seamless connectivity capabilities wihin a flexible and robust distributed coordination, as no other protocol currently does. DECOH-WIP and Bluetooth's link controller and baseband layer are contrasted.**

I. INTRODUCTION

DECOH-WIP stands for **DE**centralized **CO**operative **H**ierarchy **WI**reless **P**rototcol. It is a wireless data link protocol designed to most conviniently cover the necessities regarding the communication between mobile cooperating robots [3, 4], that are working autonomously or are being teleoperated by a local or remote user [5, 7]. The protocol is in its initial development stages, therefore many features can be still added or suppressed. This paper gives a short overview of the more than hundred pages of protocol specifications and compares it with Bluetooth [1, 2, 6], highlighting its major limitations when compared to DECOH-WIP in this field of applications.

II. DECOH-WIP PROTOCOL FEATURES

The protocol auto-reconfigure itself, to permit a dynamic variation in the number of communicating robots at any time. The maximum number of permissible working robots is limited in order to preserve the performance. This number is, nevertheless, higher or similar to what current wireless commercial protocols are able to provide, as it will be discussed later on. Also, the mode of operation, whether it is autonomous or tele-operated, is self-adaptive, i.e. it can change dynamically to one another, without provoking any functional disruption. Moreover, in teleoperated mode, the user can choose between teleoperating a single robot, or teleoperating a group of robots, whether their activities are cooperative or parallel independent ones. There is also the possibility for the user, when no tele-operation is desired, but still some delivery of information might be useful, of sending short messages to any robot, a subgroup of robots or all of them, while they are autonomously cooperating with each other, i.e. the user can still communicate with the robots, by means of very short messages, avoiding the overheads originated by the change of mode. On the other hand, a local or remote user can always profit from monitoring the activities and status of the robots. However, the permanent existence of a local or remote tele-operator or monitoring user is not mandatory within DECOH-WIP. In general, the data interchanged can be *asynchronos* or *synchronous* in nature.

The protocol permits to have any number of robots ready to become active to participate in the flow of activities, as it will be discussed later. This activities can be composed of a single task, or several simultaneuous ones. These tasks can be interrelated or totally independent. Hence, it is possible to organize the robots in different groups that can be addressed at different periodicities. Groups are allowed to appear, disappear, merge or further subdivide dynamically. The activities are not fixed or predefined, they might change according to the necessities, i.e the protocol does not impose any restrictions on higher protocol layers[1]. Furthermore, cooperating activities, due to their nature, require seamless communication capabilities. In DECOH-WIP every station[2] can communicate with every other station with no restrictions. As it will be seen later, this feature distinguishes DECOH-WIP from other protocols. Also, robots don't follow predefined roles. The different roles a robot can play are determined, at any given time, by the actual flow of events. Activities are application layer defined, while roles are data link layer defined. Thus, roles are transparent to higher layers in the protocol stack. Also, different levels of priorities are considered. Within a priority scenario a number of normal, express and emergency channels are available. The coordination is distributedly performed. All the robots intervene in the robustness characteristics of the system, providing for auto-recovering features. The protocol assumes that the robots are unreliable equippment, thus, DECOH-WIP withstands multiple stations break downs.

* Telecommunications Engineer with speciality in Telematics, graduated in the ETSIT at University of Vigo, Spain (moni_vernet@yahoo.de).

[1] The Open Systems Interconnection (OSI) Protocol Stack Model from the International Standards Organization (ISO) is followed.

[2] **Station** stands generally for a robot, in any of its possible roles, or for the PC or Laptop where the tele-operator might be locally or remotely logged in.

A. DECOH-WIP Pre-Design Assumptions

1) Within the radio link coverage area: If one robot receives uncorrupted packets, then all other robots do too, i.e. it is not possible that packets will reach some robots in a corrupted state, meanwhile the same packtes will be uncorruptively received by others simultaneously. This will be further elaborated.
2) The physical layer will throw away corrupted packets without informing the data link layer. Therefore high levels of external electromagnetical interference (EMI), packets collisions and silence will not be distinguishable from one another in the data link layer.
3) Under normal circumstances, there is an upper bound in the time the medium is unusable for transmission[3].
4) Decisions are based on what is received and not in what is sent.

B. Medium Access Control (MAC)

The communication medium is air. Its shared nature makes it very sensitive to interferences. DECOH-WIP strategy minimizes the probability of packet collisions while offering, at the same time, reliability of delivery at a sensible delay cost, as it will be analysed later on. Two phases displaying different medium access mechanisms are used in DECOH-WIP: *Power-Up* and *Steady-State*. Most of the time communication will occur under the *Steady-State* phase. Nevertheless, *Power-Up* can be triggered under auto-recovering situations or at initialization. Under success, *Power-Up* will lead to *Steady-State.* In both phases, all stations will decode, at least, the header of any received packets.

1) *Power-Up:*

The goal of the *Power-Up* phase is that, a communication coordinator will result elected, from the stations taking part in this phase. The first station successfully acknowledged will become the coordinator and *Steady-State* will begin. During *Power-Up*, as there is no coordinator yet established, there is high risk of contention. Thus, in order to lower the probability of collisions, the stations use a back-off mechanism every time they intend to transmit, which consists in waiting a random time before listening to the medium. The new comers listen to the medium for at most T. T is long enough to determine, by the packets decoded, if any, whether the phase is *Power-Up* or *Steady-State*:

- If after T nothing is decoded then *Power-Up* is assumed to be the phase in progress: the new comer proclaims itself the new coodinator and waits for the first acknowledgement (ACK). If nothing is received, several trials will be attempted. The number of attempts will be such that if no success is achieved, then with a very high probability the station is alone, in which case the *Power-Up* phase will be re-entered periodically till more stations appear into the scene.
- If packets belonging to the *Steady-State* phase are decoded, the new-comer leaves *Power-Up* and enters *Steady-State*, synchronising itself with the communicating ring. Waiting for the appropiate moment to register as an active robot.
- If packets belonging to the *Power-Up* phase are decoded, then the proper action will be triggered:
  - If a coodinator proclamation is decoded, then the new comer sends its ACK;
  - If an ACK to a coordinator is decoded, then the new comer leaves *Power-Up*.

  In both cases, the new comer enters *Steady-State*, waiting for the appropiate moment to register as an active robot.

2) *Steady-State:*

Any station can result elected as a communication's coordinator. The coordinator will be known as the *master*. At any given time, there will be only one *master*. A station that is not having the role of *master*, will have the role of a *slave* or a *floating assistant*, in case the station is a robot; or will have the role of a *PC-User*, in case the station is a local PC or Laptop were a user can login, locally or remotely. At any given time, there will be only one *floating assistant*. The robot having the role of a *floating assistant* will be changing periodically in a given pattern among the *slaves*.

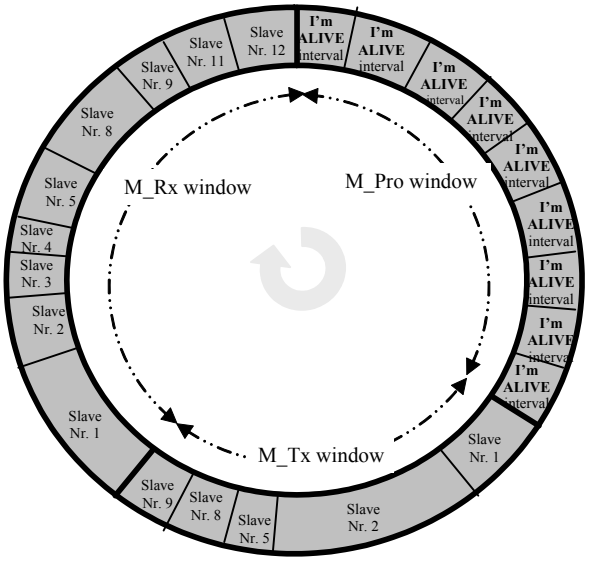

Fig.1 *Steady-State* DECOH-WIP Communication Ring.

At any time, a change of *master* can occur. One of the key issues is, that a change of *master*, under normal circumstances, will not affect the communicating ring. This ring will be mantained during the change, and finally the new *master* in functions will further follow the ring in its coordinating duties. The current *master* will elect a successor based upon priorities. When several requests are pending, the one with the highest priority will succeed, they are, in order, from higher to lower priority: *master* with

[3] Unaccessability due to collisions won't be worse as in *pure Aloha* for the same amount of traffic, as a matter of fact, the DECOH-WIP MAC strategy will produce better results as in CSMA/CA, regarding in-band interference internally originated.

emergency (cession), *slave* with emergency, *PC-User* with emergency, *master* normal (cession), *PC-User* normal and *slave* normal. A *master* request will not be ignored. It is also possible that the *master* changes due to the break down of the station bearing the *master* role, which will be discussed later on, under auto-recovery features. The result is a hierarchy of roles that will change in time, involving all the communicating parties, leading to a cooperative and decentralized coordination. A *master*, *slave*, *floating assistant* or *PC-User* will be referred to, in general, as a *station*. These roles do not propagate to higher layers in the protocol stack. A robot is programmed to be able to performe all the three roles: *master, slave* or *floating assistant* as required; whereas, a PC or Laptop for tele-operation and/or monitoring is programmed to be able to perform all two roles: *PC-User* or *master* as required.

The *master* will be in one of three situations: processing, transmitting or receiving. Therefore, the communication will be divided in three temporal windows: *M_Pro*, *M_Tx* and *M_Rx windows*, as shown in **Fig. 1**. These variable windows follow each other always in the same order. After a succession of these three windows a *cycle* of duration T_C in completed. T_C can vary in length, but its maximum value is constrained to a threshold that guarantees the responsiveness of all the working *stations*. The *master* will coordinate the access to the medium by all the *stations*. Several lists of robots will belong to the common knowledge of all the *stations*:

- *active robots list:* all participating robots, whether they are involved in any activity or not, including the *master*. They are registered with the *master*;
- *working robots list:* active robots engaged in an activity;
- *communication robots list:* subgroup of active robots that will be expected to communicate with the *master* in the *M_Rx window*. This list will be broadcasted, every *cycle*, as first *master's* duty within the *M_Tx window*, before beginning with the sequence of time slots allocated for the scheduled *slaves*.

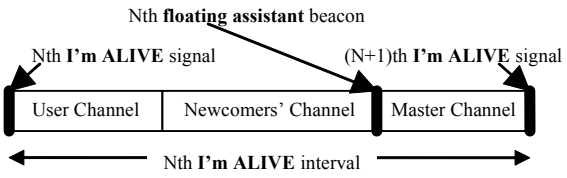

Fig. 2 Channels within *I* intervals in the *M_Pro window*.

For instance, in **Fig. 1** it is shown a situation in which the *active robots list* is comprised of, at least, 12 robots, 5 of them are scheduled to receive data, and 9 of them to send data. The scheduling permits to address robots at different periodicities, both for transmissions and receptions, as well as changing the robots addressed from *cycle* to *cycle*. It is important to note, that these lists will be used by the *master*, the other stations might need them only in the case of becoming the *master*, or for health monitoring. Also, these lists could be used in future DECOH-WIP versions for power save and control strategies.

When entering the *Steady-State* phase, the *cycle* will begin with the *M_Pro window*. This window will have an integral number of *I* intervals. This intervals are fixed in length and delimited by *I'M ALIVE* signals, *master's* beacon. These beacons are used, only within this window, for synchronization by all the *stations* and for health monitoring by the *floating assistant*. Each *I* interval is subdivided in three channels, as shown in **Fig. 2**:

- *The User Channel*: The *PC-User* can request here to become the *master*, with or without emergency, in order to be able, once in its role as *master*, to perform teleoperation. On the other hand, the *PC-User* can send here short messages to a concrete station, a subgroup, or all of them.
- *The Newcomers's Channel*: Here robots that are not yet registered, called new-comers, can request an identity, *ID*. Also, already registered robots, can here request to become the *master* with emergency.
- *The Master Channel*: Here the *master* answers the requests produced in the *User* or *Newcomer's Channel*, if any.

Between the *Newcomer's* and *Master Channel*, the *floating assistant* is expected to send its beacon. Every *I* interval, during the *M_Pro window*, the *floating assistant* will change, being one of the successive *slaves* in the *active robots list*. This will permit the *master* to scan the healthy state of all the registered robots, eventhough they might not be scheduled for communication in the next windows. Also, the last *floating assistant* whose beacon was sent before the ending of the *M_Pro window* will remain in its role till the next *M_Proc window*. Within the *M_Proc window*, the *floating assistant's* duty is to monitor the health of the *master*. If within this window the *master* breaks down, it will be the *floating assistant* the one to take over the *master's* role. When a *floating assistant's* beacon is missing, the *master* will schedule this robot within the next *M_Tx window* to check its health status.

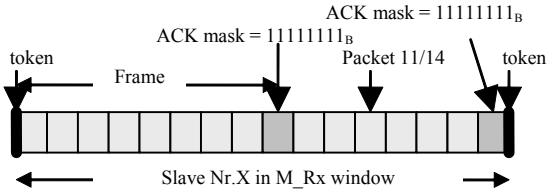

Fig. 3 Time slot for *slave* transmission. 14 packets are sent in two frames successfully acknowledged.

All the stations have an identifier electronically engraved, which is a unique number within all devices using DECOH-WIP. This number will be long, therefore it is only used during registration, where the stations acquire a short dynamicaly assigned identity *ID* given by the *master*. This *ID* will remain assigned to the same station during the lifetime of the communicating ring. Everytime *Power-Up* is triggered, *IDs* will be assigned from scratched. The robots lists, among other informations, map these *IDs* with the

long electronically engraved identifier. Robots can be assigned *IDs* from 1 on, 0 will be reserved for the *PC-User*, as well as for broadcasting.

Out of the *M_Proc window*, *slaves* will be assigned variable length time slots, for reception in the *M_Tx window* and for transmission in the *M_Rx window*. Every packet has in its header, among other informations, three identifying fields: the *source*, the *destination* and the *acknowledging* station. *Source* or *acknowledging* can involve only one station, whereas *destination* can involve a concrete station, a group of stations, or all of them. Here, advantage of the broadcasting characteristics of the medium is taken. If a station decodes a packet header with 0 or its *ID* within the *destination* field, then it will further decode the rest of the packet, also if its *ID* is in the *acknowlegding* field. Packets are further organized in frames, frames are variable in size and can contain a maximum of 8 packets. During its time slot a station can transmit a variable number of frames, each packet transmitted within the same time slot will be identified by a duple $\{N_P, N_M\}$, where N_M is the total number of packets to be transmitted in the slot and N_P is the number of the current packet, with numbering beginning at the start of the slot. From the duple, the number of frames needed to transmit N_M packets can be calculated, as well as the number of packets contained in the last frame.

DECOH-WIP uses selective negative acknowledgements, SNACKs, in a per frame bases. After the transmission of each frame, 1 byte ACK mask is expected. If the ACK mask is not received, reception time-out occurs and the whole frame is resent. Frame resent, in an ACK mask missing case, will be repeated for the same frame a maximum number of times M_F. M_F is dimensioned to guarantee with certain high probability that after M_F unsuccessful trials, the acknowledging station can safely be assumed to having broken down. Duplicate packets will be acknowledged and thrown away by the receiving station. When the ACK mask is received, each bit in the mask will correspond with a packet in the frame being acknowledged: 1_b means received, 0_b means not received; if the frame is composed of less packets than 8, then the remaining bits will be set to 1_b and ignored by the station processing the mask. Only the missing packets will be resent in the next frame. This process is repeated till the whole frame is correctly received. This acknowledging mechanism can be deactivated for slots carrying *synchronous* traffic.

Although *source* and *destination* can be set freely for every packet, their value in reality mean who else, besides the default *source* and *destination*, will be also receiving the packets. By default:

- during the *M_Tx window*, the *master* sends information to each of the scheduled *slaves*;
- during the *M_Rx window*, the *master* receives information from each of the scheduled *slaves*.

This means, that in the case it is desired to send information from one *slave* to another *slave*, the originating *slave* will set the address of the destination *slave* in the packets sent during the time slot assigned by the *master* to the originating *slave*. This packets will be received by both, the *master* and the destination *slave*, or group of *slaves*. This will occur only within spurious *slave-slave* communication needs. If these *slaves* become talkative, then one of those *slaves* should become the new *master*. Monitoring algorithms can be used to classify the stations in order of increasing connectivity needs. This information can be saved within the *active robots list*. If any station, during the *M_Tx* or *M_Rx windows* doesn't transmit when it is expected to, and this situation persists over a threshold, then this station is assumed to have broken down. In the case of a *slave*, the *master* will update its lists, crossing the *slave* but not forgetting it[4] and it will broadcast the new updated list. If the *master* is breaking down during *M_Tx* or *M_Rx windows*, it will be the *slave*, with whom the dying *master* was respectively transmitting or receiving, the one in charge of becoming the new *master*. Exceptions occur during these windows, for instance, during broadcasting from the *master*, in such situations it will be the *floating assistant* to take over the role of *master*, the next *slave* in being *floating assistant* will take over too.

C. Acknowledging a Broadcasted Message

When only one, of the intended receivers of a broadcasted message, is acknowledging it, then there is no guarantee that the message would have successfully reached eveybody. If some of the intended receivers got the message and others didn't, then and following the first assumption is *section A*, there are only two possibilities leading to this event:
(1) the receivers, missing the message, are out of the radio link coverage area; or
(2) the receivers, missing the message, are broken down.
Case (1) can be treated as case (2), since communication with those robots won't be feasible anyway. DECOH-WIP is such that it permits the robots in case (1) to discover that they are alone. In this case they will be triggering *Power-Up* periodically, two things can happen here:
(1) This lost robot can accidently re-approach the group with whom it was cooperating. Here reintegration will occur; or
(2) new different robots will approach the lost robot. In this case a different communicating ring will be established. Case (2) might pose a problem, since the two communicating rings will provoke external in-band interference to one another, giving that they are sufficiently close in space and only one frequency is being used. Avoiding external in-band interference is part of the future work in DECOH-WIP. Avoiding lost robots, within the same cooperating group, might be overcome by the exchange of positioning data at application level.

[4] Unless a *Power-Up* phase is re-entered. A broken down *slave* can be repaired, and register again.

III. FUTURE WORK IN DECOH-WIP

Besides refining the already existing features, dimensioning all the design parameters, implementating and testing. There is a very important open point. External in-band interference protection. Two different mechanisms can be followed: *cellular systems* with frequency re-use, or alternatively, *frequency hopping*. While the first introduces the disadvantage of needing cellular base stations, the second has the challenge of providing reliably: the current frequency used, the frequency sequence and the hopping instants, within a variable length time division multiplexing MAC algorithm.

IV. DECOH-WIP AND BLUETOOTH

Bluetooth is a complete protocol stack designed to offer peer-to-peer wireless communications at a minimum RF interference and low cost, while operating at very low power. Some examples of Bluetooth usage modes are: replacement of computer peripherals cabling with wireless links; use of a cordless headset to connect to multiple devices; as Internet bridge; to transfer or update files between devices, etc. The comparison will be made between DECOH-WIP and Bluetooth release 1.0 at the link controller and baseband layer, within Bluetooth's transport protocol group.

In Bluetooth, a device can have multiple connections, e.g. in a *scatternet* one device can play the role of *slave* in one *piconet* and the role of *master* in a different *piconet*; or within the same *piconet*, the *master* can be connected to several *slaves*. But all these connections are peer-to-peer or point-to-point connections and they are assumed to be independent from one another. The transmission times are regulated by the *master's* clock and are divided into *master* and *slave* transmission fixed slots, 625 μs. A *master* starts its transmission on even-numbered slots exclusively, meanwhile, a *slave* starts its transmissions on odd-numbered slots exclusively. A particular *slave* transmits if and only if the last *master* transmission was destined exclusively to this *slave*. Thus, the medium access mechanism for Bluetooth communications is a packet-based, time-division duplex TDD polling scheme. On the other hand, multi-sloted transmissions by one peer are possible, although they are limited to an odd number of slots (one, three or five), to guarantee even slots for the *master's* transmissions and odd slots for the *slave's* transmissions. Due to this TDD polling scheme, only the *master* can directly broadcast data to the other *slaves*. Whereas *slaves* in a piconet can only unicast (a point-to-point transmission) to the *master* of the piconet. This goes opposite to the seamless connectivity requirements needed among cooperating activities, i.e. any member of the group should be able to communicate with any other member in order to solve tasks together, with the help of each other. As a consequence, in the case that a *slave* will need to communicate with another *slave*, a Bluetooth *master* will have to act as intermediate node, receiving the flow of data from the originating *slave* via one TDD channel, buffering this data, and forward it later on to the receptor *slave* via a different TDD channel. This results awkward, since, the traffic is not only duplicated, but the broadcasting properties of the medium are ignored. Moreover, the responsiveness of the *slaves* is affected, since, in these situations, what it is sent it is not inmidiatedly received. Meanwhile, when broadcasting is used, only one hop is necessary to reach the receptor, as in DECOH-WIP.

On the other hand, Bluetooth *master-slave* role switch is not very mature in version 1.0 specifications besides being an optional feature. The *master-slave* role switch in Bluetooth is even more difficult due to its strong events clocking. All events during connection are synchronized with the *master's* clock and every *slave's* clock will have a different offset with the master. Furthermore, the *master's* clock determines the current frequency used in the frequency hopping, and the *master's* address determines the frequency pattern or sequence. Thus, not only TDD slots synchronization can be jeopardized in an unsuccess role switch, but also the frequency hopping tunning capability. In DECOH-WIP no synchronization based in clocks is used, leading to a much relaxed and robust mechanism. Inquirying and paging activities in Bluetooth are performed differently depending on whether the device has a *master* or *slave* role. The specification generally assumes that a Bluetooth device is capable of acting both as a *master* and as a *slave*, but certainly, devices might be built to operate only as *masters* or only as *slaves*. This will further complicate a possible switch of role scenario.

Bluetooth limits the maximum number of *active slaves* to 7. The maximum number of *working slaves*, in DECOH-WIP's nomeclature, that could be addressed are 254. ACL (Asynchronous Connection-Oriented link) traffic in Bluetooth is acknowledged in a per packet basis that might lead to heavy traffic increase, while DECOH-WIP uses selective negative acknowledgements in a per frame basis, which still assure reliable delivery but at a lower throughput cost. On the other hand, Bluetooth *broadcasted* ACL transmissions are not acklowledged at all, although they may be repeated several times for increased reliability, in anycase, that won't give delivery guarantees. Bluetooth assumes the correct functioning of all devices, whereas DECOH-WIP assumes unreliable robots. This means, under any problems, e.g. a *master* or *slave* break-down, that Bluetooth will stop till the devices are repaired. In cooperating robots environments, where human intervention might be difficult, e.g. planetary robots, and where tasks should be performed and completed eventhough some stations stop working, Bluetooth might pose weak self-recovering features, whereas DECOH-WIP is much more robust. And finally, if for some reason frequency hopping is disabled, Bluetooth cannot be used. It might be advisable to have the possibility of using a protocol that still will be working in a sudden single frequency scenario.

V. FIELD OF APPLICATIONS

The use of DECOH-WIP would permit, although the distributed nature, that independent robotic systems, physically separated and movable, would interact with each other almost as if they all were integrated within the same embedded architecture. Therefore, allowing the transformation of a single robot technology into a distributed multiple robotic system with the same performance responsiveness as in a single system, but with the empowering capabilities of distribution. Distribution permits to increase the quality and reach of the sensor data acquired as well as the complexity and accuracy of performable tasks. On the other hand, distribution also provides an enormous flexibility, easiness and quickness regarding the change or re-adaptation of applications, since it is only needed to change the characteristics of one or more robots in the group, leaving the rest of the members unchanged, to be able to met a whole new set of requirements, reverting in time and costs savings. More over, DECOH-WIP adapts itself to the communication velocity requirements, i.e. whereas collective robots cooperating in autonomous mode can interchange data within micro- to mili-seconds; when a tele-operator intervenes the transfer rate is automatically accommodated to the user quickness, while at the same time inter-machine interactions can proceed at normal higher rates.

The field of applications susceptible of beneficiating from DECOH-WIP's capabilities is quite broad, for instance in the construction of vehicles, machinery, devices, structures, buildings, bridges or roads; in the exploration of unknown or dangerous areas; in robotic surgery; in the implementation of intelligent houses, buildings and vehicles; in personal navigation systems for handicapped in hospitals and elderly residences; in the rescue of people, animals or goods in case of floods, earthquakes, avalanches or hurricanes; in the detection and dismantling of bombs in terrorist attacks; in the detection and dismantling of mines; in space robotics applications; in security mechanisms against unauthorized access to buildings and industrial complexes; in tele-maintenance and tele-diagnosis; in tele-education within Engineering diploma; in quality control and monitoring; in robotic entertainment applications etc. Also, DECOH-WIP conformance with the OSI layered architecture will permit its integration in the Internet protocol stack by simply interfacing DECOH-WIP to the Network Layer being used. The PC or Laptop in its role of *PC-user* will provide the access point to remote users. To the user, the interchange interaction at the DECOH-WIP layer is transparent. Nevertheless, its capabilities are transferred to higher layers, i.e. at the application layer the user or group of users through appropriate User-Interfaces can control and command a variable number of robots. This would permit the design of interesting Internet-based experiments for collective robots both in education as well as in industrial applications.

VI. CONCLUSIONS

Some of the most important characteristics of the DECOH-WIP protocol were presented. Among its distinguishing features are its unique medium access mechanism and how the communication coordination is distributed between all the communicating parties, as well as its robustness, auto-recovery capabilities and flexibility. DECOH-WIP was designed to cover the necessities of cooperating and tele-operated robots, where seamless connectivity, besides unlimited and un-predefined range of activities might be required. In DECOH-WIP every robot can communicate with any other robot at any time, making use of one hop broadcasting, multicasting[5], point-to-multipoint or point-to-point transmissions as needed. DECOH-WIP was compared with Bluetooth's release 1.0 link controller and baseband layer, where Bluetooth's weak points in cooperating applications were analyzed.

VII. ACKNOWLEDGMENTS

The author appreciated the support given by the TEAM Project within the EU/Canada cooperation in Higher Education and Training. Special mention deserves Marc-Antoine Fortin for the enjoyable discussing hours. Also EURON [3] and the Intituto Superior Técnico, in Lisbon, for its successful summer school in Cooperative Robotics [4]. And finally, the Universidad Carlos III for its kind invitation to participate in this 11th Mediterranean Conference in Automation and Control.

VIII. REFERENCES

1. www.bluetooth.com
2. Bray, J., Ch. F. Sturman. BLUETOOTH connect without cables. Prentice Hall, N.J., 2001.
3. Euron – European Robotics Research Network. www.euron.org
4. European Summer School in Cooperating Robotics. http://www.isr.ist.utl.pt/~euron/
5. Khamis, A., M. Pérez Vernet, K. Schilling (2002). "A Remote Experiment on Motor Control of Mobile Experiments". *Proceedings of the 10th Mediterranean Conference on Control and Automation*, med2002, Lisbon, Portugal, July 2002.
6. Miller, B. A., Ch. Bisdikian. Bluetooth Reveald. Prentice-Hall, Inc., N.J., 2001.
7. Pérez Vernet, M. and K. Schilling (2002). "Virtual Reality Applications for Tele-Operated Mobile Robot Control". *Proceedings of the 12th International Symposium on Measurement and Control in Robotics – Advanced Robotics and Virtual Reality*, Bourges, France, June 2002.

[5] Distinction is made here between multicasting and point-to-multipoint wireless communications. Although in both cases the same data could be sent to the final group of addresses, with multicasting the data travels via a single "common" connection, while in point-to-multipoint the data is sent via multiple connections, one per address. On the other hand, one hop is explicitly mentioned, since for point-to-multipoint transmissions, it doesn't necessarily mean that all the connections have to co-exist simultaneously.