An Application Protocol to Integrate a Small Size Helicopter into an IP based Ad-Hoc Network

Florian Zeiger, Christian Selbach, Benjamin Ruderisch and Klaus Schilling
University of Würzburg
Department of Computer Science
Robotics and Telematics
97074 Würzburg, Germany
Email: {zeiger,schi}@informatik.uni-wuerzburg.de

***Abstract*—In modern multi robot systems, wireless any-to-any communication in combination with highly dynamical network topologies are prerequisites for mobile robots to accomplish complex tasks. This work presents a protocol for steering and controlling a small size helicopter. It fully supports the easy integration into an internet protocol based mobile ad-hoc wireless network, as well as several security and safety issues as it is required for a reliable and robust teleoperation of an unmanned aerial vehicle. The requirements of a helicopter control protocol, as well as implementation and design details on a safe and robust teleoperation are described. Besides these implementation details, also real hardware tests are evaluated to show the seamless functionality and performance of the helicopter control protocol in combination with ad-hoc on demand distance vector routing (AODV) inside the mobile ad-hoc network. It could be proved, that the proposed application protocol for steering a small size helicopter could be used in combination with AODV and relevant parameters for a reliable communication are identified.**

I. Introduction

During the last years, advances in the fields of communication, miniaturization, and computer science enabled the robotics community to formulate more and more complex tasks and scenarios for multi robot systems with real mobile robot hardware. Important features of current multi robot systems are a reliable any-to-any communication in combination with a highly dynamical network topology in terms of changing the location of nodes as well as a variable number of nodes inside the network. Several systems of rovers with autonomous functionalities [1], groups of unmanned aerial vehicles [2], as well as heterogeneous multi robot systems were proposed. Often, standard IEEE 802.11 wireless LAN (WLAN) is used as underlying technology. For ground based systems Chung [3] presented a testbed for a network of mobile robots. With respect to unmanned aerial vehicles (UAVs), [4] presented a system using an access point running in WLAN infrastructure mode onboard the UAV. [5] presented a system for communication between a ground station and a UAV using WLAN in combination with a high-gain antenna and radio modem. Nevertheless, not only WLAN is used as radio technology. MARVIN, a helicopter with certain autonomous functionalities uses the DECT technology for communication [6]. As these multi robot systems aim to enable a reliable communication between all network nodes, several approaches are used to provide connectivity. In [7] and [8], a distributed shared memory is implemented. Each robot is connected to its dedicated ground station and the ground stations are connected via network.

The mentioned research realized a reliable radio communication within robot and control PC, but often ad-hoc capabilities were not present. The progress in available technologies supporting ad-hoc network functionalities provides researchers a fast networking of several mobile robots with different capabilities. Each robot is a mobile node of this network and thus the network topology can change very fast. Therefore, reliable mechanisms for routing inside these ad-hoc networks are a prerequisite for robust communication among the robots. Research in the sector of telecommunication evolved many (more than 70) different routing protocols for ad-hoc networks. In general there are two popular protocol families providing the majority of the existing approaches: pro-active and reactive ad-hoc routing protocols. In addition, there exist numerous routing protocol families like hybrid, hierarchical, power aware, and many more.

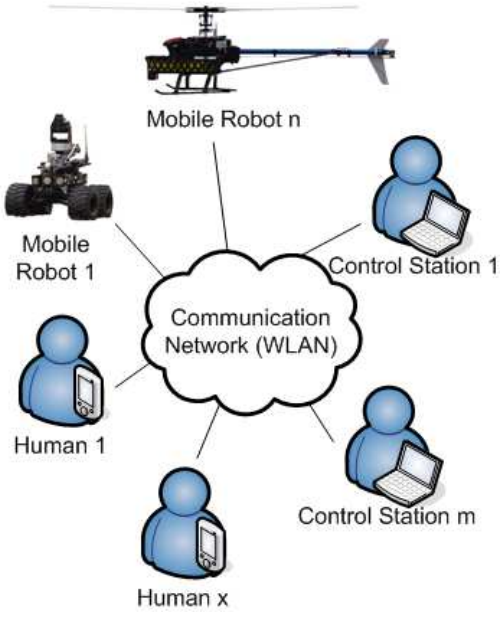

Fig. 1. Any-toany communication between all network nodes.

ROBOCOMM 2007, 14th – 16th Oct 2007, Athens, Greece.

ISBN 978-963-9799-08-0
DOI 10.4108/ICST.ROBOCOMM2007.2180

Pro-active routing protocols always keep an updated list of the network destinations. This list is periodically broadcasted throughout the the network. But there exist some disadvantages, too. As also unused routes are maintained in the routing table and the routing table is transmitted periodically, some of the available bandwidth is wasted. In addition, some pro-active routing algorithms might not be the best choice for large networks due to problems in scalability. Some well known pro-active routing protocols are Destination-Sequenced Distance Vector (DSDV) [9] or Optimized Link State Routing Protocol (OLSR) [10]. In contrast to pro-active routing protocols, reactive protocols start the route discovery on demand. Thus, a delay in finding this route cannot be avoided. Well known reactive routing protocols are Ad-hoc On-demand Distance Vector (AODV) [11], Dynamic Source Routing (DSR) [12], and Dynamic NIx-Vector Routing [13]. With respect to the use of ad-hoc routing protocols for the networking of mobile robots, several issues are of importance. Usually, different traffic types with different characteristics and requirements are present. On one hand, the communication link is used to transmit commands of human operators from control stations to several robots and for sending sensor data from the robots back to the ground station. Usually, the traffic generated by commands consists of small packets which are often sent periodically. Traffic caused by sensors might be very different – form small data packet containing only some bytes of data up to large data sets (e.g. generated by laser scanners), video streams, or even voice communication. On the other hand, the ad-hoc network of mobile robots might be used for the transmission of control commands and sensor data feedback for controllers. In this case, the network is really a part of the control process and must fulfill a set of constraints. The highly dynamic topology of a network of mobile robots might cause delays due to route discoveries or communication losses. This work presents the use of an application protocol for controlling or steering a small size helicopter used in an ad-hoc network which connects several mobile robots. Several effects of multi-hop communication (e.g. delays and packet loss during reestablishing a route) and their effects on the robustness of the helicopter control/steering protocol and the UAV tele-operation are investigated.

The work is organized as follows. Section II describes the requirements for a communication protocol for controlling/steering a small size helicopter. In Section III, implementation details of the helicopter which are relevant for the protocol design and the features of the implemented protocol are are presented. Section IV gives details on the integration of the helicopter into the ad-hoc network. In Sections V and VI tests related to safety and robustness in different scenarios and the results of these tests are presented. A conclusion is given in Section VII.

II. Requirements of a Communication Protocol for Controlling and Steering a Helicopter

Remote controlling a helicopter is a very difficult issue concerning security aspects and becomes even more complex if an ad-hoc network connects the remote pilot or the ground control station to the helicopter. The communication protocol for controlling a helicopter should provide a reliable mechanism for the transmission of sensor data and command packets. To allow a safe operation of the UAV, several requirements onboard the helicopter, at the PC of the ground control station, as well as for the communication itself must be fulfilled.

A. Security Issues

The most important requirement of the control protocol is related to security issues. While using a radio link for communication, a loss of the connection is always possible. Thus, a communication dropout has to be detected fast and reliably, and a secure reestablishing of the communication between control PC and helicopter must be possible. In addition, the user must be informed and the helicopter must stay in a stable and safe condition – e.g. stationary hovering. After reestablishing the communication link, the status of the helicopter and the control PC must be synchronized to provide valid data to the user.

Besides these aspects of operational security, also security aspects with respect to user management, prevention of misuse or intrusions, and encryption of connections should be possible.

B. Integration into existing Computer Networks

Besides the above mentioned security issue, the helicopter and the control PC should be integrated into a network of mobile robots. In current research projects, several network topologies – starting form wireless LANs using infrastructure mode and several access points up to wireless ad-hoc networks using different ad-hoc routing protocols – were used [3], [4], [5], [6], [8]. Often, these networks uses standard IP based communication together with WLAN. An advantage of WLAN is the availability of a relatively high bandwidth and the high flexibility in integrating new protocols or extend features of available protocol implementations. As currently the cooperation of several vehicles is very important, challenging problems like nodes acting autonomously as communication relay, a highly dynamic and variable network topology (some network nodes may leave of join the network at any time), routing problems, and several data streams and sources with different bandwidth requirements have to be solved. Often, ad-hoc capabilities must be present.

C. Hardware Compatibility and Availability

Furthermore, aspects like cost efficient hardware components, independence from a single manufacturer, compatibility of new and old hardware (e.g. IEEE 802.11g vs. 802.11b), availability of development libraries, open protocol stacks for low level implementations, and easy integration in existing PC networks must be considered. Also physical aspects like power consumption and the range of the radio link is an important decision criterion and closely related to the planned scenario.

D. Support Features of the Helicopter

In general, each protocol for controlling and steering of a specific robot is a specially designed interface to provide a set of basic functionalities, and in addition support special features of the used mobile robot. The presented protocol is mainly used to send sensor data from the helicopter to the ground control station and for transporting commands from the ground control station PC to the helicopter. As the helicopter provides two different operation modes – the joystick mode, and the task mode (see Section III-A) – the command data generated by the ground station has different characteristics in packet size and the frequency of the packets sent. In addition, several ground control stations should be able to connect to the helicopter for retrieving data but only one dedicated control station should be in command of the helicopter.

III. Implementation Details

The presented helicopter control protocol is used in an environment which allows any to any communication between all network nodes like mobile robots, UAVs, or humans with a PC or PDA (cf. Figure 1). Therefore, a WLAN running in ad-hoc mode is used. Each robot and UAV is equipped with a PC architecture and a standard TCP/IP and UDP/IP protocol stack.

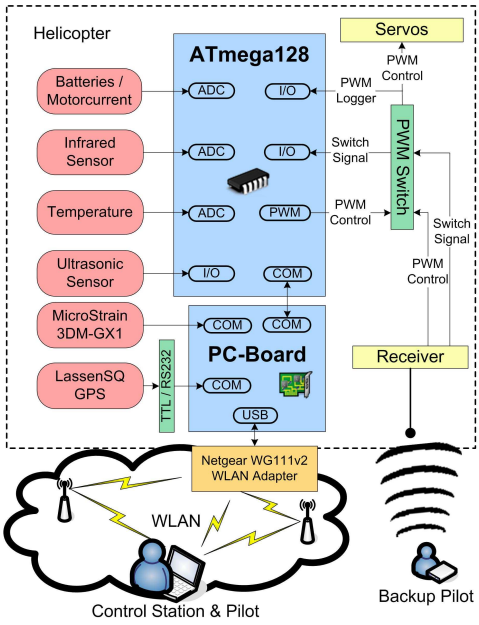

Fig. 2. Hardware of the helicopter.

A. Helicopter

The Helicopter is a modified version of a commercial remote controlled helicopter. It is equipped with several sensors for determining its position, velocity and orientation [14] (cf. Figure 2). The onboard data processing is done by a low power PC with a SUSE Linux as operating system. Communication to the control PC is realized via standard WLAN components. Furthermore, a radio link in the 35 MHz band based on standard remote controlled model helicopters is available for a manual flight in emergency situations. The control PC runs a helicopter control software which displays the transmitted sensor data and sends control commands to the helicopter. The communication between helicopter and ground station is based on standard 802.11 WLAN and is described more detailed in the following sections.

As the presented protocol and the status of the communication has a major impact on the behavior of the helicopter, at first some aspects of the helicopter control software must be explained. The helicopter provides two flight modes for the user. The joystick mode provides a simple steering of the helicopter velocities in all directions. Here, the user is supported by a flight assistance system which keeps the helicopter in a stable condition. The second operation mode (task mode) supports the transmission of a mission with a list of waypoints which the helicopter visits on its flight path autonomously.

The helicopter flight control has a simple finite state machine (FSM) implemented which provides the features mentioned in Section II-D. The following paragraph gives a short overview of this FSM onboard the helicopter. The FSM of the control software has five states the helicopter can use: *Not Initialized, Locked Hover, Unlocked Hover, Joystick Mode*, and*Task Mode* (see Figure 3). After starting the onboard software, the helicopter enters the *Not Initialized* state. In this state, no commands for the helicopter will be accepted due to a missing connection to a control PC. After the initialization procedure, the helicopter enters the *Locked Hover* mode and a connection to an available control station is established. In the *Locked Hover* mode, the helicopter keeps its position and altitude and waits for the *unlock-command* from the control PC to enter the *Unlocked Hover* mode. During the *Unlocked Hover* mode, the helicopter holds its pose and waits for commands to enter either the *Joystick Mode* or the *Task Mode*. If the helicopter is in *Joystick Mode*, *Task Mode*, or *Unlocked Hover* mode and the connection to the control PC is lost, it immediately switches to the *Locked Hover* mode.

B. Control and Steering Protocol

As the integration of the helicopter communication into existing computer networks is a main objective, the helicopter control protocol is based on the ISO/OSI layer model. To provide a maximum of compatibility to communication via Ethernet and WLAN and for also supporting the special features of the helicopter (e.g. *Joystick* or *Task* mode), the helicopter control protocol is located at the application layer of the ISO/OSI model. At layer four, the possibilities of using UDP or TCP were analyzed. For the following aspects, the implemented helicopter control protocol uses UDP as transport protocol:

- **Packet retransmissions of sensor data is often not useful:** As the sensor data should be transmitted periodically, the retransmission of a lost packed will result in transmitting old sensor data which is of course not important for the tele-operator.

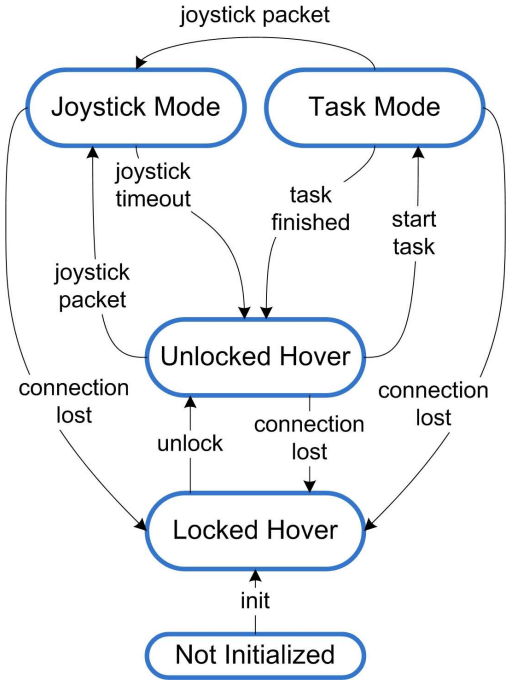

Fig. 3. State diagram of the helicopter.

- **Distributed network architecture with several control PCs:** Using a UDP socket, packets from different stations can be received. Thus, several network nodes can request sensor data from the helicopter to this specified UDP socket. Furthermore, the helicopter can join the existing network via broadcast or multicast. Thus, UDP allows that several stations are listening to sensor data while only one station is in command of the helicopter.
- **Only a small amount of data which has to be transmitted reliably:** As only some packets have to be transmitted reliable, UDP is the easiest implementation approach. For the small amount of data being sensitive on packet loss, a separated *Reliable Channel* (cf. Section III-B1) with a very simple functionality is implemented.
- **Blocking of internal buffers by data which is sent periodically:** As a disturbed TCP connection will reduce the throughput, internal sending buffers will be filled with old sensor data. These old packets cannot be removed from the queues and they are blocking and delaying newer data.
- **Joystick Flight Mode:** The Joystick Flight Mode enables a user to control the helicopter's velocities in all directions via a normal joystick. Onboard systems provide the required flight assistance so that the user has no need to keep the helicopter in a stable condition. The complete system can be considered as a control loop for the helicopter's velocities in all three directions, whereas the user is the controller giving commands based on sensor data provided through the control PC. As the control PC continuously sends commands to the helicopter, a packet loss is not as dangerous as repeating old packets with obsolete control commands. For this reason, a TCP connection must not be used as delay caused by packet retransmissions may result in a non controllable situation for the remote user.
- **Loosing the WLAN connection causes long delays due to timeouts:** In case a TCP connection is lost and cannot be reestablished, the system will detect the loss of connection after a certain timeout (more than 5 seconds). During this time, it is not possible to connect again to this TCP socket.

As a real flow control and reliable transmission of all packets (like it is implemented in TCP) is not required in the current design, the advantages of using UDP preponderate the disadvantages. In case an acknowledged transmission of packets is necessary, the mechanism of the implemented *Reliable Channel* (see Section III-B1) can be used. Nevertheless, using standard UDP provides full compatibility towards lower layers of the ISO/OSI model (e.g. internet protocol (IP) at layer three). This allows encryption mechanisms like the use of a virtual private network (VPN) to fulfill security aspects. Additionally, already existing ad-hoc routing protocols like AODV [11], or DSDV [9] can be used in combination with the presented helicopter control protocol. Sections V and IV will give more details on the performance of the presented protocol in combined use with the ad-hoc routing protocol AODV.

The presented helicopter control protocol uses packets of variable length. The header consists of two parts, a mandatory section with transport information (cf. Table I) and an optional part in case the *Reliable Channel* is used (cf. Section III-B1 and Table III).

Bit	Name	Description
0-3	VER	version of the protocol
4-7	PRI	priority of the packet
8	R_CH	set if the simple reliable channel should be used
9-13	-	reserved
14-23	LEN	size of packet including header and payload
24-31	TX_ID	unique ID of the sender
32-39	RX_ID	unique ID of the recipient
40-...		data

TABLE I
HEADER WITH TRANSPORT INFORMATION.

The payload consists of twenty different command packets of variable size (cf. Table II).

The KEEP_ALIVE_REQ and KEEP_ALIVE_RESP packets are used to detect a communication loss (see Section III-B2. Packet types 2-8, 12, 13, 15, and 16 are used for exchanging status information between helicopter and ground control station PC, whereas STATUS_RES_TRI (packet type 5) is sent periodically by the helicopter. While the helicopter is in the *Joystick Mode*, JSTK_CMD packets are accepted. In the *Task Mode*, TASK_UPLOAD and TASK_STAT_INF packets are exchanged. For the initialization between helicopter and control PC, INIT packets are used (cf. Section III-B3). The HELI_HELLO packets are sent from the helicopter via broad-

ID	NAME	SIZE (BYTES)	RELIABLE CHANNEL
0	KEEP_ALIVE_REQ	11	no
1	KEEP_ALIVE_RESP	11	no
2	TIME_REQ	11	no
3	TIME_RES	17	no
4	STATUS_REQ	9	both
5	STATUS_RES_TRI	15	no
6	STATUS_RES_REQ	16	both
7	UNLOCK_REQ	11	yes
8	UNLOCK_RES	19	no
9	JSTK_CMD	28	no
10	TASK_UPLOAD	>7	yes
11	TASK_STAT_INF	1	yes
12	GS_INF_REQ	1	no
13	GS_INF_RES	2	no
14	INIT	4-20	no
15	CONF_REQ	1	no
16	CONF_RES	1	no
17	HELI_HELLO	1	no
18	WARNING	10	no
19	ERROR	10	no

TABLE II
COMMAND PACKET TYPES.

BYTE	BIT	NAME	DESCRIPTION
0	0	ACK	indicates an acknowledgment nevertheless, it can contain data
0	1	ACK_REP	indicates a packet loss after the packet with the sequence number given in SEQ_LRX
0	2	NO_DATA	in case NO_DATA = 1, this packet is only an acknowledgment without data
0	3-7	-	reserved
1-4		SEQ_LRX	contains the last sequence number of a correctly received packet
5-8		SEQ_RX	sequence number of this packet
9-10		RX_WND	shows the size of the free memory of the remote station

TABLE III
HEADER WITH INFORMATION FOR THE RELIABLE CHANNEL.

cast or multicast into the network and they are used to disclose the helicopter details to all network nodes.

1) The Reliable Channel: As mentioned already above, UDP is not designed for reliably transmitting packets. Usually, the sender receives no feedback from the receiver whether a packet was transmitted sucessfully or lost. With respect to the requirements of the helicopter control protocol, a reliable transmission of packets is only necessary for a few packet types (cf. Table II) in special situations. For this reason, a special mechanism called *Reliable Channel* is implemented in the helicopter control protocol. The mechanism is designed very simple. In case sender A transmits a packet to station B via the reliable channel, receiver B sends an acknowledgment packet (the ACK-flag of the *Reliable Channel* header is set) with the sequence number of the received packet in the SEQ_LRX field. Station A now knows the sequence number of the last packet which was transmitted correctly and can initiate retransmissions if required. The *Reliable Channel* assures successful transmissions of packets and has no functionalities implemented to replace the complete traffic flow control of TCP.

2) Detection of a communication loss: The possibility of a communication loss is always present and the communication can also be lost asymmetrically. This means, station A can send packets to station B successfully, whereas the return link from station B to station A is broken. To recognize a communication loss reliably, both stations – in our case the helicopter and the ground control PC – monitor the status of the communication link. Helicopter and the ground control station periodically send HELI_HELLO packets every $80ms$ to keep the communication status updated. In case the helicopter receives no new HELI_HELLO packets in an interval of 2 seconds, the helicopter considers the communication link broken. In this case, the *Locked Hover* state is entered and the *unlock* process is required (refer to Subsection III-B3) to leave this state. In case only the ground station recognizes a time out of the HELI_HELLO packets received from the helicopter, the user is notified. In this case, the helicopter will not necessarily change its own status as the HELI_HELLO packets to the helicopter might be submitted successfully.

3) Initialization and Unlock process: In general, there exist two different connection procedures for the synchronization between helicopter and ground control station – a complete initialization and a reestablishing of a broken connection. A complete initialization is necessary for the first connection between helicopter and control PC (cf. the *init* transition from state *Not Initialized* to *Locked Hover* in Figure 3). This *init* process is used to exchange and update the sequence numbers of the status data and the reliable channel. The initialization is always started by a control station. It is also necessary to provide a reliable initialization in case more than one control station started the *init* procedure at the same time. As, due to simultaneously started *init* procedure attempts, the packets of these attempts can be mixed, the packets of each *init* process have to be associated to the corresponding attempt. Therefore, the ground control station selects a unique value and includes it to the *init* packet. The corresponding reply packet of the helicopter then also includes this value and the ground station can identify the *init* packets of the corresponding initialization attempt.

The *init* process itself has four simple steps:

1) The helicopter has a counter which is incremented by one for each successful initialization. The ground station requests this counter from the helicopter.
2) The helicopter sends a reply containing this counter value.
3) The ground station now requests the initialization which corresponds to the counter value exchanged in the step before and sends its own sequence numbers of the status data and the reliable channel.
4) In case the initialization procedure is successful, the helicopter replies by sending its own sequence numbers for status transmission and reliable channel. Furthermore, the initialization counter will be incremented by one and the state of the FSM is changed to *Locked Hover*. In case

the *init* procedure fails, an error message is sent to the control PC.

If a broken connection to an already initialized helicopter should be reestablished, an *unlock* procedure (cf. transition from *Locked Hover* to *Unlocked Hover* in Figure 3) is performed. Therefore, the *Reliable Channel* is used as its sequence numbers are still synchronized. The control station sends only its current sequence numbers of its joystick and status packets. The helicopter itself replies by sending its corresponding sequence numbers.

IV. Integration into Ad-Hoc Networks

The above presented helicopter steering protocol is placed at the application layer of the ISO/OSI reference model and thus can use all available features provided by the layers below. In a wireless network with a static topology, an existing route between helicopter and ground station will not be changed and the helicopter control protocol works reliably. In worst case, a connection is lost and the procedures described in Section III-B3 have to be performed. Considering a high mobility of the network nodes (cf. Section II), possible negative effects caused by the changing topology must be considered. In this case, a loss of an existing communication to a network node not necessarily means a completely unreachable destination. It might be possible that another route through the wireless network still exists. Ad-hoc routing protocols provide a solution for these scenarios and also imply some potential sources for integration problems. The most important aspects with respect to the described helicopter control protocol are the required time for establishing an alternative route, the end-to-end delay between helicopter and control station, and the packet loss during the route reestablishment.

For the investigated test scenarios, AODV is used as ad-hoc routing protocol. AODV uses routing tables containing information about a destination, the next hop to reach this destination, the number of nodes to reach this destination and some more information about the status of a route. It uses three different types of messages Route Requests (RREQ), Route Replies (RREP), and Route Errors (RERR) for distributing routing information. As soon as a route to an unknown destination is requested, the source sends a request RREQ via broadcast and the route is discovered after the RREQ reached the destination or a node which knows the destination. The destination sends a RREP back to the requesting source node via the same route used from source to destination. A loop is prevented by using sequence numbers. Each node monitors the status to its neighbors. As soon as a link is lost, RERR is sent as error message containing the unreachable node.

With respect to the combined use of AODV and the helicopter steering protocol, two important aspects will be analyzed while increasing or decreasing the hop count between source and destination. First, the packet loss and second, the packet loss probability and the change in the delay during this event. Considering the rerouting, it may happen that the connection will be lost for a period longer than 2 seconds (which is always possible while using a radio link) or a number of packets will be lost for a certain time. In these cases the helicopter control protocol must react robust and keep the FSM onboard the helicopter in a defined state. The helicopter control protocol has a timeout Δt_{cl} of 2 seconds before a link is considered broken and the helicopter enters the *Locked Hover* mode. Single packet losses and communication gaps shorter than 2 seconds should not affect the helicopter steering protocol.

V. Test Scenarios

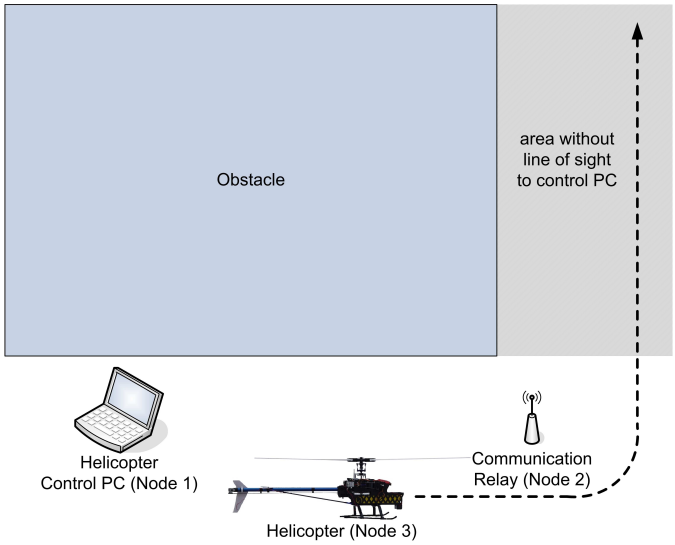

Fig. 4. Test-scenario for communication via relay.

The presented test scenario is used to show the behavior of the helicopter steering protocol in combination with AODV. Therefore, three network nodes are used. Node 1 is the control station PC, node 2 is a communication relay, and node 3 is the helicopter. All nodes have AODV-UU version 0.9.3 [15] [16] running which is RFC3561 compliant. For all tests, AODV is used with the standard parameter setting.

The test setup is shown in Figure 4. The helicopter will move from the control PC into the area without a direct communication to the control station. The direct communication between helicopter and control PC must be changed to a relay communication via node 2. Measurements for determination of the packet loss rate and the delay behavior will be carried out.

VI. Results

The following figures show the measured round trip times (rtt) of packets and the packet loss during a rerouting process. The round trip times are measured for ICMP packets, as well as for helicopter control protocol packets. For the helicopter control protocol, also the packet loss is shown. To calculate the packet loss, a sliding window over the past time and the ratio of sent packets versus received packets within this time interval are used. The length of this sliding window is set to the length of the timeout Δt_{cl} which is used by the helicopter control protocol to discover a communication loss. For the following

pictures, the y-axis either displays the round trip time of ICMP or helicopter control protocol packets in milliseconds, or the packet loss probability. The x-axis always shows the current time of the test in seconds.

A. Delays

In Figure 5 the round trip time of ICMP packets between the helicopter and the control PC during a rerouting by AODV is shown. From 0 up to 345 seconds on the x-axis, the helicopter communicated via two relay nodes tho the control PC. The rtt is about 4 milliseconds. Beginning from 345 seconds until 500 seconds, the number of communication hops was and the communication link was routed via the relay node 2. In this case, the rtt is about 3 milliseconds. All round trip times over 10 milliseconds can be considered as packet losses or timeouts. After 500 seconds, the link changed to a direct communication with a round trip time of about 1 millisecond. No timeouts occurred during this rerouting.

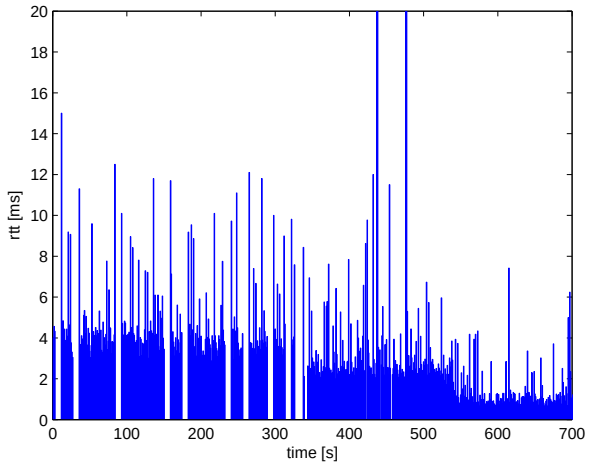

Fig. 5. Round trip time (rtt) of ICMP messages during decrease of hop count.

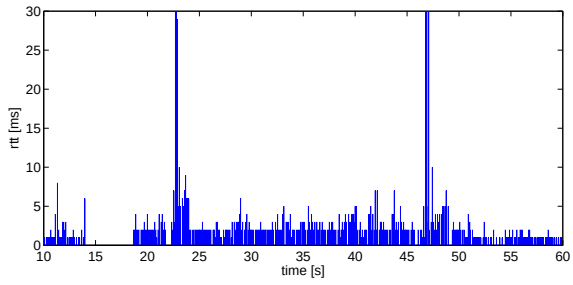

Fig. 6. Round trip times (rtt) of helicopter steering packets during rerouting.

The rtt of helicopter control packets is displayed by the graph in Figure 6. In the beginning, a direct communication link was available. The rtt of the helicopter control protocol of about 1 millisecond is not significantly higher than the rtt of the ICMP messages of the previous test. Between 14 and 18 seconds on the x-axis, the communication was lost. During this time interval, the rerouting process took place. From 18 to 51 seconds, the communication link is routed via the relay node. The rerouting to direct communication happens at 51 seconds. The communication dropout for about 4 seconds while establishing the route via the relay node forced the helicopter to switch to *Unlocked Hover* mode. The high round trip times at about 23 seconds and 43 seconds have a value of about 1 second which will not cause the helicopter tho change its state to *Locked Hover*. However, the delay alone cannot be used to characterize whether the helicopter control protocol will recognize a complete loss of communication or not. Therefore, the packet loss for the duration of Δt_{cl} must be investigated.

B. Packet Loss

In Figure 5, also the packet loss is visible. While communication is routed via two relay nodes, missing round trip times indicate lost packets. After the number of communication hops is decreased, the packet loss is also reduced. A more detailed view gives the following test result of plotting the packet loss of the helicopter control protocol. During the tests for measuring the packet loss for the helicopter control protocols, the packets are transmitted all 20 milliseconds. The helicopter steering protocol will recognize a communication dropout if the packet loss during timeout duration Δt_{cl} is 100%. In Figure 7, the packet loss of helicopter control packets is displayed. The x-axis scale corresponds to the x-axis of Figure 6 and has the same timestamps for the rerouting events. During the rerouting form direct communication to relay communication between 14 seconds and 18 seconds, the communication dropout (packet loss of 100% for more than 2 seconds) forced the helicopter to switch to the *Locked Hover* mode. While switching back to direct communication at 51 seconds a packet loss of less than 20% occurred which does not affect the helicopter's FSM.

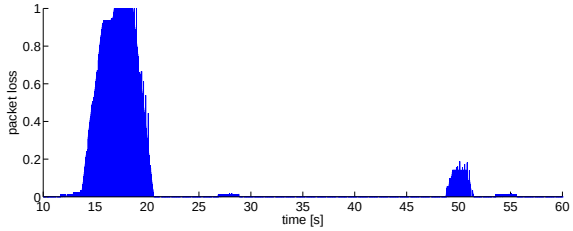

Fig. 7. Packet loss of helicopter steering packets during rerouting.

VII. Conclusion

During the tests, the communication protocol stack was forced to initiate a rerouting process as the network topology was changed. The round trip times and the packet loss of the helicopter control protocol packets were analyzed during increasing and decreasing the number of hops used for communication. As the tests and measurement results have shown, the presented helicopter steering protocol can be used together

with AODV without major difficulties. Even the used standard parameter settings of AODV allowed a safe teleoperation of the helicopter. Nevertheless, possibilities for performance improvements and further investigations are also identified. Often, the communication link is lost for more than 2 seconds while the number of hops is increased. Thus, the helicopter switches to the *Locked Hover* mode which complicates the teleoperation. Simulations proved that AODV might perform better under certain conditions. In case of the presented use, a parameter adaption of AODV also leads to better performance. Relevant parameters of AODV are investigated more detailed to reduce the required time for route discoveries and for avoiding alternating routes. In addition, the combined use of the presented helicopter control protocol together with other ad-hoc routing protocols (which are mentioned in Section I) are investigated in future.

REFERENCES

[1] L. E. Parker, "Alliance: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots," in *IROS*, 1994.

[2] A. Ollero, G. Hommel, J. Gancet, L.-G. Gutierrez, D. Viegas, P.-E. Forssén, and M. González, "COMETS: A multiple heterogeneous UAV system," in *IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR 2004)*, 2004.

[3] T. Chung, L. Cremean, W. B. Dunbar, Z. Jin, E. Klavins, D. Moore, A. Tiwari, D. van Gogh, and StephenWaydo, "A Platform for Cooperative and Coordinated Control of Multiple Vehicles," *3rd Conference on Cooperative Control and Optimization*, 2002.

[4] A. Ollero, J. Alcazar, F. Cuesta, F. Lopez-Pichaco, and C. Nogales, "Helicopter Teleoperation for Aerial Monitoring in the COMETS Multi-UAV System," in *3rd IARP Workshop on Service, Assistive and Personal Robots (IARP 2003), Madrid (Spain)*, 2003.

[5] R. Vidal, O. Shakernia, H. J. Kim, H. Shima, and S. Sastry, "Multi-Agent Probabilistic Pursuit-Evasion Games with Unmanned Ground and Aerial Vehicles," *IEEE Transactions on Robotics and Automation*, vol. Vol. 18, Number 5, pp. 662–669, 2002.

[6] M. Musial, G. Hommel, U. W. Brandenburg, E. Berg, M. Christmann, C. Fleischer, C. Reinicke, V. Remuß, S. Rönnecke, and A. Wege, "MARVIN - Technische Universität Berlin's Flying Robot Competing at the IARC'99," in *Fachgespräche Autonome Mobile Systeme AMS 99, Band 15, pp. 324-333, Springer- Verlag, München, 1999*, 1999.

[7] V. Remuß and M. Musial and U. W. Brandenburg, "BBCS – Robust Communication System for Distributed Systems," in *Proceedings of the SSRR '04 - IEEE International Workshop on Safety, Security, and Rescue Robotics, Bonn, Germany*, 2004.

[8] Volker Remuß and Marek Musial, "Communication System for Cooperative Mobile Robots using Ad-Hoc Networks," in *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[9] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance Vector (DTDV) for Mobile Computers," in *Proc. of the SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications*, 1994.

[10] P. Jaquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clausen, "Optimized Link State Routing Protocol (OLSR)," in *RFC 3626*.

[11] C. E. Perkins, E. Royer, and S. Das, "Ad hoc On-demand Distance Vector (AODV) Routing," in *RFC 3561*.

[12] D. B. Johnson, D. A. Maltz, and J. Broch, *Ad Hoc Networking*. Addison-Wesley, 2001, ch. Chapter 5: DSR - The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, pp. 139–172.

[13] Y. J. Lee and G. F. Riley, "Dynamic NIx-Vector Routing for Mobile Ad Hoc Networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2005)*, 2005.

[14] C. Herrmann, F. Zeiger, L. Ma, C. Selbach, and K. Schilling, "Design and Test of an Autonomous Helicopter for Multi-Vehicle Cooperation," in *Accepted for Proceedings of Intelligent Autonomous Vehicles (IAV 2007), September 3-5, Toulouse, France*, 2007.

[15] H. Lundgren, E. Nordström, and C. Tschudin, "Coping with Communication Gray Zones in IEEE 802.11b based Ad hoc Networks," in *Proceedings of The Fifth International Workshop on Wireless Mobile Multimedia*, 2002.

[16] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin, "A Large-scale Testbed for Reproducible Ad hoc Protocol Evaluations," in *Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC 2002*, 2002.