Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 6, pp. 5714–5721, DOI 10.12694/scpe.v25i6.2602

REDUNDANT TELEMETRY LINK SYSTEM FOR UNCREWED AERIAL VEHICLES*

ALI ELBASHIR† AND AHMET SAYAR‡

Abstract. Thanks to their flexibility in research, prototyping, and development, Uncrewed Aerial Vehicles (UAV) are seeing increasing use in agriculture, healthcare, logistics, and other industries. With the increase in the use of UAV in civilian sectors, comes an increase in the possibility of errors in telemetry connections. Issues with the telemetry connection may be catastrophic and cause accidents and crashes and may also adversely affect the performance of the subsystems connected to the telemetry. In this study, a low-cost software-based redundant telemetry link system for UAV is proposed. The proposed system minimises the received packet loss using packet deduplication and is compared with a dynamic link switching method. In the proposed system, both fault tolerance and robustness are introduced into the UAV telemetry link. The analysis is performed using the ArduPilot UAV Simulation environment as a producer and a Golang implementation of the proposed methods.

Key words: Uncrewed Aerial Vehicle, MAVLink protocol, autopilot, packet loss, Ground Control Station

1. Introduction. UAVs have been deployed in many sectors due to rapid technological advances and low production costs. Defence, agriculture, healthcare, logistics, and various other industries have turned to UAV technology to automate their business processes and make them more efficient. Furthermore, the development of open source autopilot software such as ArduPilot and PX4 has encouraged research and development in this field, allowing UAV technology to be adopted by a wider user base.

However, with the proliferation of UAVs, several challenges have emerged that can affect the safety and efficiency of their operations. The most important of these challenges is the reliability and stability of telemetry links. In particular, among US military UAV accidents, "In more than a quarter of the accidents examined by The Post, links were lost around the time of the crash" [1]. In the absence of hostile interference, which is also a big contributor to military UAV accidents, the rate of loss of telemetry links is expected to be even higher in the civilian sector. To overcome these difficulties, it is important to maintain continuous and error-free telemetry links. However, a variety of environmental factors and technical errors frequently disrupt the communication of telemetry data. Therefore, this paper presents a review of how to minimise the effects of unreliable UAV telemetry links and how these links can be optimised.

In this context, a study is conducted on the effectiveness and efficiency of redundant telemetry link systems. The goal is to ensure that the telemetry data from the UAV reaches the ground control station (GCS) quickly and that the link can be changed in the event of a link failure. As a result of our research, our telemetry link system is both fault tolerant and performant. A performant link significantly increases the probability of mission success for the UAV. Within this framework, we investigate the use and effectiveness of dynamic link switching and packet deduplication methods.

Key Contributions. Our work dives deep into the nuances of UAV telemetry links, striving for enhancements in both reliability and performance. The key contributions of this study are the following:

1. An in-depth examination of redundant telemetry link systems, spotlighting their potential to augment UAV mission success rates.
2. Introduction of an application layer fault-tolerant and high-performance redundant telemetry link system
3. Comparative analysis of dynamic link switching versus packet deduplication within a redundant telemetry link system

*This work was funded by the TUBITAK 2209-A Undergraduate Research Support Program
†Department of Computer Science and Engineering, Kocaeli University, Kocaeli, Turkey (alielbashiir@gmail.com)
‡Department of Computer Science and Engineering, Kocaeli University, Kocaeli, Turkey (a.sayar@kocaeli.edu.tr).

After this introduction, the paper subsequently delves into basic concepts in Sect. 2, reviews related works in Sect. 3, outlines the proposed methods in Sect. 4, examines the experimental results in Sect. 5, and culminates in Sect. 6 with conclusions and prospects for future research.

2. Basic Concepts.

2.1. Redundant Systems. In the field of computer science, redundant systems are critical in their role in providing data protection and service continuity. Redundant systems are systems that take over functionality when there is an error or failure in the primary system [2]. It is important to define the importance of redundant systems and the various types of redundant systems.

Redundant systems are designed to maintain functionality in the event of a failure. They ensure the continued functionality of the system in situations where the primary system is repaired or replaced. Redundant systems are especially critical in environments with high safety requirements, such as hospitals and aeroplanes.

System redundancy generally comes in two main forms: passive and active. In passive redundancy, the backup system is activated only when the primary system fails. This type of redundancy is typically used for data protection and recovery. In active redundancy, both the primary and backup systems are running at the same time, and both share the workload. This ensures higher system performance and service continuity.

2.2. MAVLink. Micro Air Vehicle Link (MAVLink) is a protocol for communication with small uncrewed vehicles. MAVLink is designed as a Marshalling library, which means that it serialises the messages of system states and the commands it needs to execute in a specific sequence of bytes, hence being platform independent. The binary serialisation structure of the MAVLink protocol makes it lightweight as it has minimal overhead compared to other serialisation techniques (XML or JSON).

It is mostly used for communication between a Ground Control Station (GCS) and uncrewed vehicles, and for communication between the vehicle's subsystems themselves. It is used to transmit data such as the vehicle's heading, GPS position, speed, etc. This protocol is used in widespread autopilot systems such as ArduPilot and PX4, and is used not only to control uncrewed aerial vehicles, but also to control autonomous systems such as Remotely Operated Underwater Vehicles and Uncrewed Ground Vehicles. [3]

MAVLink Message Format

Communication using MAVLink is bidirectional between the ground station and the uncrewed vehicle. The MAVLink protocol defines the mechanism for the structure of messages and how they are serialized at the application layer. These messages are then forwarded to the lower layers (i.e., the transport layer, physical layer) for transmission to the network. The advantage of the MAVLink protocol is that its lightweight nature allows it to support different types of transport layers and environments. MAVLink can be transmitted using sub-GHz (433 MHz, 868 MHz, 915 MHz, etc.), WiFi, Ethernet, and serial connections.

MAVLink messages are described in XML files. Each XML file defines the set of messages supported by a particular MAVLink system, also called a "dialect". The reference set of messages implemented by most MAVLinks and autopilots is defined in common.xml (most dialects are built on top of this definition). An example of a MAVLink dialect XML definition file is given in Fig. 2.1.

Code generators parse the XML definitions and create codec libraries for supported programming languages, which can then be used to communicate with UAVs, GCSs, and other MAVLink systems. MAVLink messages can be defined as two types:

1. Status messages: These messages are sent from the UAV to the ground station and contain information about the status of the system, such as its identity, position, speed and altitude.

2. Command messages: They are sent by the GCS (or other client programs) to the UAV to make the autopilot execute some action or task. For example, the GCS can send a command to a UAV to take off, land, navigate to a way-point, or even execute a mission with several linked actions.

These messages are encoded into the payload of the larger MAVLink frame, which is shown in Fig. 2.2

3. Related Works. To strengthen the communication system of UAVs, Lau & Ang worked on a link switching algorithm [5]. In their system, the "heartbeat" message defined in Fig. 2.1 in common.xml is used to determine the state of the link. In the event that a connection is broken, the system automatically switches to a backup connection. This provides fault tolerance, and thus maintains the communication continuity of the UAV.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<mavlink>
  <version>3.0</version>
  <system_id>1</system_id>
  <component_id>1</component_id>
  <messages>
    <message id="0" name="HEARTBEAT">
      <field type="uint32_t" name="custom_mode" units="" enum="MAV_MODE_FLAG"/>
      <field type="uint8_t" name="type" units="" enum="MAV_TYPE"/>
      <field type="uint8_t" name="autopilot" units="" enum="MAV_AUTOPILOT"/>
      <field type="uint8_t" name="base_mode" units="" enum="MAV_MODE_FLAG"/>
      <field type="uint32_t" name="system_status" units="" enum="MAV_STATE"/>
      <field type="uint8_t" name="mavlink_version" units=""/>
    </message>
    <message id="1" name="SYS_STATUS">
      <field type="uint32_t" name="onboard_control_sensors_present" units=""/>
      <field type="uint32_t" name="onboard_control_sensors_enabled" units=""/>
      <field type="uint32_t" name="onboard_control_sensors_health" units=""/>
      <field type="uint16_t" name="load" units=""/>
      <field type="uint16_t" name="voltage_battery" units=""/>
      <field type="int16_t" name="current_battery" units=""/>
      <field type="int8_t" name="battery_remaining" units=""/>
      <field type="uint16_t[10]" name="drop_rate_comm" units=""/>
      <field type="uint16_t[4]" name="errors_comm" units=""/>
      <field type="uint16_t[6]" name="errors_count1" units=""/>
      <field type="uint16_t[6]" name="errors_count2" units=""/>
      <field type="uint16_t[6]" name="errors_count3" units=""/>
      <field type="uint16_t[4]" name="errors_count4" units=""/>
      <field type="int8_t" name="battery_remaining2" units=""/>
    </message>
  </messages>
</mavlink>
```

Fig. 2.1: MAVLink message definition example

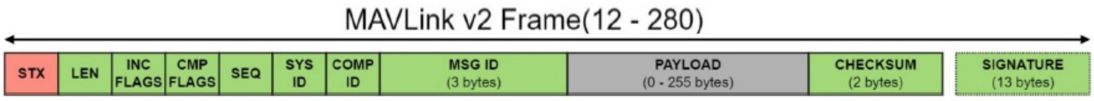

Fig. 2.2: MAVLink V2 packet format. Source:[4]

In addition, projects such as mavproxy, mavp2p, and mavlink-router support the use of parallel connections [6]. Parallel connections allow both incoming telemetry and outgoing commands to be exchanged between multiple connections. This provides a backup connection in case one link drops or fails. However, since packet deduplication is not implemented in these projects, this creates the effect of ”rubberbanding” in the case there is a latency difference between the links.

For experimental studies, most previous work has used the IEEE 802.11b (Wi-Fi) standard. Brown et al. [7, 8] implemented a wireless mobile ad hoc network on radio nodes in fixed locations, off-road vehicles, and UAVs. One of their scenarios is to extend the operational scope and range of small UAVs. They measured network efficiency, delay, range, and connectivity in detail under different operating conditions and showed that the UAV has longer range and better connectivity. Jimenez-Pacheco et al. [9] built a lightweight ad hoc mobile network (MANET). The system implemented line-of-sight dynamic routing in the network. Morgenthaler et al. described the implementation and characterisation of a mobile ad hoc network for lightweight flying robots. The UAVNet prototype was able to automatically interconnect two end systems through a flying relay.

Asadpour et al. [10, 11], conducted a simulation and experimental study of image relay transmission for UAVs. Yanmaz et al. proposed an antenna extension to a simple 802.11 device for aerial UAV nodes. They tested the performance of their system and showed that 12 Mbps communication can be achieved at a communication distance of about 300 m. Yanmaz et al. [12] conducted a tracking study on two UAV networks using 802.11a. In this study, they showed that high throughput can be achieved using two-hop communication. Johansen et al. [13] described a field experiment in which a fixed-wing UAV acts as a wireless relay for an underwater vehicle. In that study only download data rates were tested, in particular, while Wi-Fi networking is suitable for creating multiple access points, the communication range is very limited, and many nodes are required for long-range applications. An advantage of the Wi-Fi network is its high throughput. However, for UAV telemetry communication, distance is a more important requirement than data transfer rate. This paper presents a UAV communication system that is different from previous research. Our system is based on a Ground Control Station (GCS) with two links to the UAV. In this way, a structure that can establish both relay connections and provide direct connections is obtained. Our system is flexible and can support different communication protocols such as serial protocols and wireless technologies such as Wi-Fi for data transmission. Because our system works at the application level, it is independent of the transport and data link layers. These features offer the ability to better adapt to various scenarios and communication needs. This work is differentiated from previous work by its new approach and more flexible structure in the UAV-GCS communication.

Our research goes even deeper into these issues. In particular, we investigate how link switching and deduplication algorithms can be used dynamically effectively.

4. Method.

4.1. Link Switching. The system calculates the packet loss in real time throughout the flight and shows which link has the least packet loss. To ensure the command packets sent from the GCS reach the UAV as they are sent over all the links, not just the active one.

If a link is detected to be completely down, it is disabled, and telemetry is received on the remaining active links. When the number of active links drops to 1, all telemetry data is received from that link.

The link switching algorithm's pseudocode is shown in Algorithm 1.

Algorithm 1 Link Switching Algorithm

```
for each packet do
    if packet is coming from the active link then
        route the packet to the GCS
        update last received packet time
    else if packet is coming from the GCS then
        route the packet to the UAV through all links
    end if
    calculate time difference from the last received packet.
    if packet received time difference ≥ link switching duration then
        switch active link with inactive link
        update last received packet time
    end if
end for
```

The dynamic link switching algorithm monitors the packet loss rate and delay on the main link and switches to the backup link if these values exceed the set thresholds.

Detection time is the time period between the end of the transmission of data packets over the main link and the detection of the end of the transmission. Ideally, the detection time should be as short as possible so that outage situations can be quickly identified and responded to.

Response times refer to the completion times of the operations performed after detection. When a link outage is detected, the system must react quickly to enable the backup link to be activated. Response times include the time required to activate the backup connection and ensure a smooth transition of telemetry data. These times may vary depending on the method of connection switching and the complexity of the system infrastructure.

The continuous monitoring of link quality process is used to determine the difference in performance between the main link and the backup link and to make a decision to switch links when necessary. Various metrics can be used to monitor link quality; for example, parameters such as latency, bandwidth, packet loss rate, and signal strength can be evaluated.

Latency measures the time delay in the transmission of data packets from the main link to the backup link. In many cases, lower latency means better connection quality and indicates faster data transfer. The bandwidth determines the amount of data transmitted simultaneously, with a larger bandwidth providing higher data transfer rates. The packet loss rate refers to the rate at which data packets are lost. A low packet loss rate indicates reliable data transmission [14].

Various algorithms can be used for continuous monitoring of connection quality. For example, network protocols such as ping and traceroute can be used to measure connection performance and detect faulty points on the network. Monitoring software can also be used to track statistics on the sending and receiving of data packets.

The thresholds set during this monitoring process are used to determine whether the quality of the connection is good or bad. For example, when the packet loss rate exceeds a certain threshold, the link quality is considered poor, and a link-switching process can be triggered. These thresholds are predetermined depending on the requirements of the telemetry system and the application scenario.

Consequently, continuous monitoring of link quality is a critical step for the reliability and performance of the backup telemetry system in UAVs. This monitoring process is accomplished through the use of various metrics, and link replacement is initiated when thresholds are exceeded.

The step of determining the packet loss rate and setting thresholds is performed to ensure accurate and reliable collection and transmission of telemetry data. Various statistical methods can be used to determine the lost packet rate. For example, the packet loss rate can be calculated as a percentage of the difference between the number of packets sent and received. The determination of threshold values is important to identify situations where the packet loss rate exceeds acceptable levels. These values should be determined according to the needs of the UAVs and the telemetry system used [15].

For example, during a UAV mission, a threshold value should be set on the packet loss rate that occurs on the main link when the backup telemetry system is activated. This threshold represents a point at which packet loss exceeds acceptable levels. When the threshold is exceeded, a switchover from the active link to the backup link is performed. To properly perform this switchover, the packet loss rate and the threshold value must be taken into account. For example, if the threshold value is set at 5%, the connection switching process must be initiated if the packet loss rate exceeds this value.

4.2. Paket Deduplication. Packet deduplication is a method of checking whether a packet from a source has arrived before. It is used in TCP/IP routers, data storage systems, and systems with parallel connections [16].

Packet deduplication can be implemented in many different ways, but since our use case is real-time and sensitive, we use a temporary buffer that keeps the packets of the last 5 seconds.

Deduplication starts by checking whether packets from different sources are arriving. For this, a performant and suitable duplicate detection algorithm needs to work. There are 2 options here:

1. Store the incoming packet itself and compare every byte of each incoming message with the stored packets
2. Get the hash of each incoming packet, store it in a hashmap, and check if the hash of each incoming packet exists in the hashmap

The first option has two obvious problems. The first one is that it requires us to store all messages. We might need to store about 124 MAVLink messages per second (for the Ardupilot simulation environment), so in the worst case 263 * 124 bytes. If we use an 8 byte hash, that is 8 * 124 bytes, which is 32 times less memory space, which can be even more significant depending on the hardware the switch is running on and the message rates it receives.

As a hashing algorithm, we use the 4-byte version of MurmurHash, which is used in high-speed real-time applications such as Hadoop [17], ElasticSearch [18] and Kafka [19].

5. Experimental Results. In the experimental studies carried out during this work, the impact and performance of packet deduplication were evaluated. The results obtained are shown in Fig. 5.2

Data collection was performed using the simulation environment of the open source MAVLink-based Ardupilot autopilot software. In the experimental environment, a UAV was left in ”Loiter” mode and a packet loss of 1% per second was inserted using the tc command available in Linux distributions. This scenario was designed to mimic the challenging situations that can be encountered in real world conditions. UDP was used as the transport layer protocol during the experiment.

First, data reflecting the state before deduplication was applied was recorded. Then, using a 4 byte version of the MurmurHash algorithm, packet deduplication was performed and the data were recorded as such.

The analysis of the experimental data was carried out by measuring the packet loss rate. Data recorded after deduplication was considered as the control group, while data recorded without deduplication was used as the comparison group.

The results show that packet deduplication is an effective method. It is observed that after deduplication, the packet loss rate is reduced from 10% to 30%. This shows that the redundant telemetry link system provides more reliable and consistent communication.

We also evaluated the impact of deduplication on transmission times. Although a buffer is used in deduplication, packets are not kept in that buffer before being sent. Therefore transmission times are not affected in any way.

Another part of our experimental work focused on the study of the mechanism of dynamic switching of telemetry links. In this process, we studied a system that forwards packets from the active link. However, in this case, the loss rate of the filtered packet is not lower than the loss rate of the active link packet, as shown in Fig. 5.1.

In some scenarios, this may cause the loss rate to be higher than that of the active link, even if the packet loss rate of the inactive link is lower. This is also shown graphically in our experimental results. When the packet loss rate of the active link reaches 100% and a threshold time (e.g. 1 second) is exceeded, the link is automatically switched.

Compared to packet deduplication, link switching is less efficient. Two main reasons explain this:

1. If the active link is broken during the threshold time, even if the inactive link is active, no packets will be forwarded to the ground control station (GCS) until the link is replaced.

2. The filtered packet loss can be as low as, but not lower than, the packet loss of the lowest link. This shows that the link switching mechanism has more limitations than packet deduplication in terms of communication continuity and efficiency in certain situations.

These results should be taken into account when determining optimisation strategies. In particular, under harsh environmental conditions and high packet loss rates, packet deduplication seems to provide a more effective solution than link switching.

6. Conclusion. In this article, a study on a redundant telemetry connectivity system for UAVs is analysed in detail. The purpose of the study is to evaluate the feasibility of using MAVLink packet forwarding methods for transmitting telemetry data to UAVs by duplicating MAVLink packets at an application layer level.

The packet deduplication method was found to be suitable for a real-time and accurate use. A copy detection algorithm with MurmurHash was used. This method is based on the principle of taking the hash value of each incoming packet and checking whether it has been received before.

The experimental results confirm the effectiveness of the packet deduplication method. In simulations with an additional packet loss of 1% per second, it is observed that the loss of deduplicated packets is on average 10% to 30% less. This shows that packet deduplication is an effective solution to improve the reliability of UAV telemetry links.

This paper also investigates the mechanism for dynamic switching of telemetry links. This approach allows the use of a backup link in case the active link is down. However, our experimental studies have shown that this mechanism has certain limitations. In particular, if the threshold time is exceeded and the link is down, the inactive link is not used until the active link is replaced. Furthermore, the link-switching method cannot guarantee the link that can offer the lowest packet loss rate.

These observations suggest that the link switching mechanism may be less efficient than packet deduplication in certain scenarios. Therefore, additional optimisation work needs to be done in order to make more efficient use of the link switching mechanism.

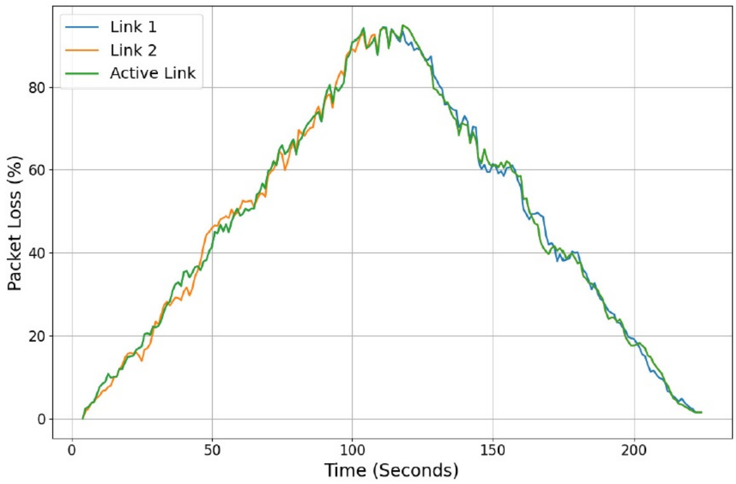

Fig. 5.1: Packet loss using dynamic link switching

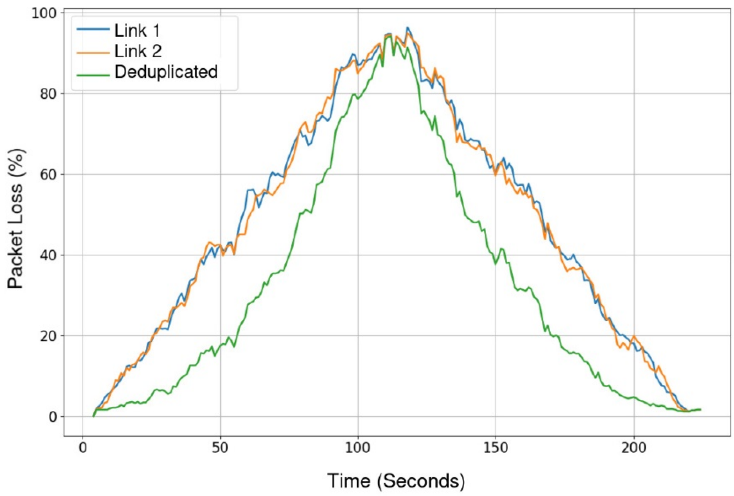

Fig. 5.2: Packet loss using packet deduplication

Furthermore, the scope of the study is limited to MAVLink packets. It is important to apply similar methods for different protocols or data types and evaluate the results. Further research on different types of telemetry data and protocols is recommended for future work.

The results highlight the importance of a redundant telemetry connectivity system for UAVs and show that packet deduplication is an effective solution to improve the reliability of this system. It is hoped that experts in the field of UAV technologies and telemetry will be inspired by this study for further evaluation and implementation.

In conclusion, this study has addressed the concept of a redundant telemetry connectivity system for UAVs and demonstrated that packet deduplication is an effective solution to improve the reliability of this system. Future work should support progress in this area with similar studies on different protocols and data types.

Acknowledgement. The authors of this paper would like to thank Mr. Alessandro Ros – developer of the MAVLink implementation for the Go language (GOMAVLIB) – for his support and code review.

REFERENCES

[1] C. Whitlock, *When drones fall from the sky*, The Washington Post, https://www.washingtonpost.com/sf/investigative/2014/06/20/when-drones-fall-from-the-sky/. [Accessed: 24 May 2023].

[2] D. Systems, *Redundant systems: Definition, important, working method*. https://dexonsystems.com/news/redundant-systems, 2022.

[3] MAVLink Development Team, *MAVLink Repository*. https://github.com/mavlink/mavlink. [Accessed: 25 May 2023].

[4] Dronecode Foundation, *MAVLink Serialization Guide*. https://mavlink.io/en/guide/serialization.html. [Accessed: 25 May 2023].

[5] Y. H. Lau and M. H. Ang, *A novel link failure detection and switching algorithm for dissimilar redundant uav communication*, Drones, 5 (2021), https://doi.org/10.3390/drones5020048, https://www.mdpi.com/2504-446X/5/2/48.

[6] P. Kancir, *Mavlink routing with a router software*, 2022, https://discuss.ardupilot.org/t/mavlink-routing-with-a-router-software/82138. [Accessed: 24 May 2023].

[7] T. Brown, B. Argrow, C. Dixon, S. Doshi, R.-G. Thekkekunnel, and D. Henkel, *Ad hoc uav ground network (augnet)*, in Proceedings of the AIAA 3rd Unmanned Unlimited Technical Conference, Chicago, IL, USA, September 2004, pp. 1–11.

[8] T. Brown, S. Doshi, S. Jadhav, and J. Himmelstein, *Test bed for a wireless network on small uavs*, in Proceedings of the AIAA 3rd Unmanned Unlimited Technical Conference, Chicago, IL, USA, September 2004, pp. 20–23.

[9] A. Jimenez-Pacheco, D. Bouhired, Y. Gasser, J. Zufferey, D. Floreano, and B. Rimoldi, *Implementation of a wireless mesh network of ultra light mavs with dynamic routing*, in Proceedings of the IEEE Globe Work, Anaheim, CA, USA, December 2012, pp. 1591–1596.

[10] M. Asadpour, D. Giustiniano, and K. Hummel, *From ground to aerial communication: Dissecting wlan 802.11n for the drones*, in Proceedings of the WiNTECH 13 8th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization, Miami, FL, USA, October 2013, pp. 25–32.

[11] M. Asadpour, D. Giustiniano, K. Hummel, S. Heimlicher, and S. Egli, *Now or later?—delaying data transfer in time-critical aerial communication*, in Proceedings of the 2013 ACM International Conference on Emerging Networking Experiments and Technologies (Conext '13), New York, NY, USA, December 2013, pp. 127–132.

[12] E. Yanmaz, S. Hayat, J. Scherer, and C. Bettstetter, *Experimental performance analysis of two-hop aerial 802.11 networks*, in Proceedings of the 2014 IEEE Wireless Communications and Networking Conference (WCNC), Istanbul, Turkey, April 2014, pp. 3118–3123.

[13] T. Johansen, A. Zolich, T. Hansen, and A. Sørense, *Unmanned aerial vehicle as communication relay for autonomous underwater vehicle—field tests*, in Proceedings of the Globecom 2014 Workshop—Wireless Networking and Control for Unmanned Autonomous Vehicles, Austin, TX, USA, December 2014.

[14] S. Wang, J. Li, and R. Zheng, *Active and passive control algorithm for an exoskeleton with bowden cable transmission for hand rehabilitation*, in 2010 IEEE International Conference on Robotics and Biomimetics, 2010, pp. 75–79, https://doi.org/10.1109/ROBIO.2010.5723306.

[15] S. James, R. Raheb, and A. Hudak, *Impact of packet loss to the motion of autonomous uav swarms*, in 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC), 2020, pp. 1–9, https://doi.org/10.1109/DASC50938.2020.9256561.

[16] N. Wilkins, *Packet deduplication explained*, 2020, https://www.garlandtechnology.com/blog/packet-deduplication-explained.

[17] HBase, *Hadoop in Java*. HBase, 2011, https://web.archive.org/web/20120112023407/http://hbase.apache.org/docs/current/api/org/apache/hadoop/hbase/util/MurmurHash.html. Archived on 12 January 2012. Retrieved 13 January 2012.

[18] *Elasticsearch 2.0 - CRUD and routing changes*. Elastic, 2022, https://www.elastic.co/guide/en/elasticsearch/reference/2.0/breaking $_{2}0_{c}rud_{a}nd_{r}outing_{c}hanges.html_{r}outing_{h}ash_{f}unction.$

[19] *DefaultPartitioner.java*. Apache Foundation, 2022, https://github.com/apache/kafka/blob/trunk/clients/src/main/java/org/apache/kafka/clients/producer/internals/DefaultPartitioner.java.