Profiling IEEE 802.11 Performance on Linux-based Networked Aerial Robots

Robert Kuschnig[1], Evsen Yanmaz[2], Ingo Kofler[1], Bernhard Rinner[2], and Hermann Hellwagner[1]

[1] Institute of Information Technology (ITEC)
[2] Institute of Networked and Embedded Systems (NES),
Klagenfurt University, AUSTRIA
<firstname>.<lastname>@uni-klu.ac.at

Abstract. Continuous innovation in wireless technology, such as adaptive rate control and new antenna systems (beam forming, antenna diversity, and multi-input multi-output systems), enhances the performance of wireless transmission systems at a cost of increased complexity. While sensor networks operating at low data rates can do without these new technologies, they are essential for aerial robot networks, where visual sensor data (like high resolution images or video) are expected to be transmitted. When the data rate requirements are coupled with the sensitivity of the wireless links between aerial robots to the position or the orientation of the robots and as well as their motion, development of high-capacity links can become a great challenge. An intuitive way to achieve better link quality is to design the wireless transmission system via profiling the wireless links in real-world measurements. This paper shows how to realize and analyze IEEE 802.11 performance measurements on Linux-based platforms.

Keywords: 802.11, measurements, Linux, unmanned aerial vehicles

1 Introduction

In the last couple of years, unmanned aerial vehicles (UAVs) have gained great interest. While the main driving force behind UAV development has been the military, small-scale UAVs have recently become also available for the civilian market. Typical applications include aerial photography or surveillance, live video streaming, which may be used to support the navigation of the UAVs, or still image transmission for generating high resolution overview images of areas of interest. All of these applications require a data downlink of the sensed data from the vehicles to other vehicles or a ground control station.

While ground robots usually communicate on the same plane with other robots or the base station, networked aerial robots (like UAVs) move in the three dimensional space and are expected to require more sophisticated transmission systems. The wireless links need not only support high data rates required by the application, but overcome the limitations due to the positioning, orientation, and motion of the aerial robots. The IEEE 802.11 standard, which provides high

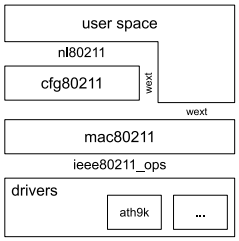


Fig. 1. Architecture of the Linux wireless subsystem [6].

data rates, is a typical interface deployed in wireless networks and, as an initial step, we assume the UAVs are equipped with such wireless interfaces. Modeling as well as handling the limitations due to the motion of the vehicles on the other hand is a challenging task. As illustrated in [3, 7], collecting real-world measurement data over an experimental testbed of networked aerial vehicles can prove very useful to achieve this goal.

One possibility to gather information on the wireless transmission quality is the use of the Linux wireless subsystem [6]. Even more detailed information can be extracted from packet traces by utilizing the monitor mode in the Linux wireless subsystem [4]. In this paper, we provide a methodology to profile the wireless links for Linux-based networked aerial vehicles. We present two options to collect link quality information and monitor the 802.11 wireless interface. We illustrate the use of our approach via some real-world measurements between a UAV and a ground control station.

The rest of the paper is organized as follows. Section 2 provides the architecture of the Linux wireless subsystem. In Section 3, two methods for Linux-based 802.11 measurements are introduced. The measurement results for a 802.11a-based network are presented in Section 4. Section 5 concludes the paper.

2 Architecture of the Linux Wireless Subsystem

The Linux wireless subsystem [2] consists of several modules, which handle the configuration of the IEEE 802.11 wireless hardware and manage the transmission of the data packets [6]. Figure 1 depicts the architecture of the subsystem. In the wireless subsystem (as of early 2012) the configuration is made in the *user space* (e.g., associating to an access point or setting the transmit power) and the Linux Netlink [5] interface `nl80211`[3] is used to access the `cfg80211`[4] module in the kernel. The `cfg80211` module implements configuration options which are common in IEEE 802.11 platforms. It also performs active tasks like scanning the network for access points (APs) and manages the encryption of the wireless transmission channel.

The `mac80211`[5] module implements the medium access layer in software, which can be used by wireless devices that expect the MAC-layer to be imple-

[3] http://linuxwireless.org/en/developers/Documentation/nl80211
[4] http://linuxwireless.org/en/developers/Documentation/cfg80211
[5] http://linuxwireless.org/en/developers/Documentation/mac80211

mented in the device driver (also called SoftMAC devices). Nowadays nearly every chipset uses this SoftMAC approach, because it lowers the hardware costs. Therefore, `mac80211` handles the construction of the 802.11 frames and implements also the rate control algorithms (see `mac80211` documentation). The `mac80211` module directly interacts with the IEEE 802.11 device drivers (e.g., `ath9k`) to transmit or receive data packets. The drivers have to implement the `ieee80211_ops` interface for that purpose. For legacy reasons, the interfaces of the old wireless driver framework are still supported by means of the wireless-extensions (`wext`).

3 Methods for Profiling the 802.11 Performance in Linux

The wireless subsystem of Linux offers two different options to gather information on the wireless transmission system. First, a snapshot of the wireless connection status can be retrieved via the `nl80211` interface. Second, by using the monitor mode of the wireless subsystem, it is possible to get detailed information for each received and transmitted network packet.

3.1 Using Wireless Connection Status

The status of a wireless connection can be monitored by using the `nl80211` interface to the wireless subsystems. The `nl80211` interface describes all commands which can be issued and information which can be retrieved. An example is the `iw` tool[6], which is an `nl80211`-based command-line interface configuration utility for wireless devices. On an access point the "station dump" option can be used to print all information about the connected client. The information includes average *signal strength* (SS) of the last received data packets and the sending and receiving *data rate*. At the client, the "link status" option of `iw` can be used to get information about the connectivity to the access point.

The standard `nl80211` interface only reports the signal strength of the antenna used for receiving. But usually more than one antenna is used in the wireless cards to enable diversity. To get an idea how the differnt antennas perform and which antenna is selected in the transmission process, the OpenWrt project[7] extended the Linux wireless subsystem and specifically the `ath9k` 802.11 driver[8] to report the signal strength of individual antennas.

While gathering information on the wireless transmission quality by means of the `nl80211` interface is straightforward, there are still some open issues. To continuously monitor the transmission channel, polling has to be used, which is computationally expensive if the time interval between the samples is short. The reported values are usually averaged, but the method how the values for the single packet transmissions are aggregated is not documented. Also, the update frequency of the data is unknown. For that reason, the `nl80211` interface is usually only used to monitor the performance in static environments. In the

[6] http://linuxwireless.org/en/users/Documentation/iw
[7] https://openwrt.org/
[8] https://dev.openwrt.org/browser/trunk/package/mac80211?rev=30753 and https://dev.openwrt.org/browser/trunk/package/iw?rev=30753

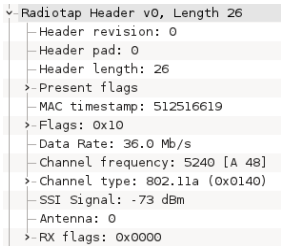


Fig. 2. Sample output of wireshark showing the radiotop header.

next section, a method for gathering information on the individual packet transmissions will be presented.

3.2 Using Wireless Packet Capture

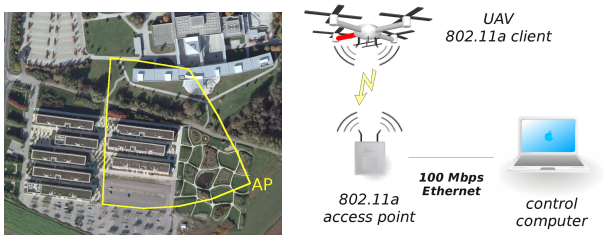
In addition to the usual operation modes of a wireless interface (i.e., infrastructure or ad-hoc mode), the Linux wireless (IEEE-802.11) subsystem also provides a *monitor mode*[9]. In this mode, all packets received by the wireless device are handed over to the operating system, which is very useful when debugging a wireless system. This monitor can be used in parallel to a normal operation mode (like infrastructure mode), which helps to gather detailed information on the wireless transmission.

In the monitoring mode, the wireless subsystem creates a new wireless device, which can be used to capture the wireless network packets. While this is also possible with non-monitor devices, the monitor device adds a special header to the IEEE 802.11 frames, which carries the information on the wireless transmission. This so called `radiotap header`[10] includes detailed information on each packet, like the signal strength or the data rate of the received packet. For sent packets, the header includes the number of retransmissions needed and whether the packet was successfully transmitted or not. Detailed information on the fields of the `radiotap header` is listed in [1]. Since each captured packet also includes a very accurate time stamp, other performance values such as throughput, packet delay, and packet loss rate, can also be derived.

Because the Linux wireless subsystem is still work in progress and not all drivers supply the same information, not all reported values are valid. For instance, in our measurements, we experienced that with the current `ath9k` driver and the used hardware (see Section 4), the antenna used for receiving the packets is not reported correctly. Another issue when capturing at packet level is the high data volume, especially when using MIMO (802.11n). To mitigate this problem, the packet capture interface allows to capture only packet headers.

Apart from using the capture interface directly, also established software can be used to capture and analyze the wireless packets. The most prominent tools

[9] http://linuxwireless.org/en/users/Documentation/modes
[10] http://linuxwireless.org/en/developers/Documentation/radiotap



(a) Test area (b) Test setup

Fig. 3. Test area (waypoints over campus, 150 m × 150 m) and test setup.

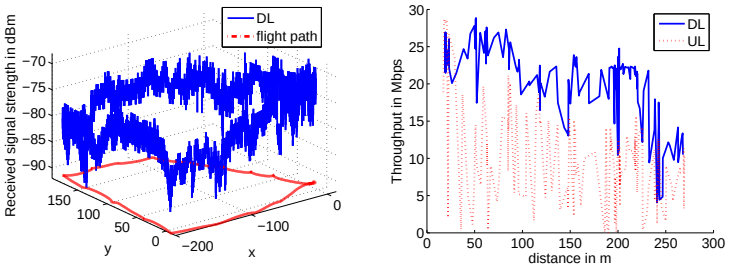
are `tcpdump`[11] and `wireshark`[12] [4]. `Tcpdump` is a command-line program used to capture network packets and runs also fast on embedded devices. It offers powerful filter capabilities to reduce the data volume. `Wireshark` is a graphical tool, which can capture and analyze network packets. In addition, it can also parse and decode the `radiotap header` (see Figure 2).

4 Measurement Results for 802.11a-based UAV-to-Ground Links

To illustrate how the methods for measuring the 802.11 performance can be used for profiling the wireless links in real-world flights, we constructed a flight route around our campus (as depicted in Figure 3). The test setup consists of one UAV (AscTec Pelican), which is equipped with a board featuring an Intel Atom processor (Ubuntu Linux 10.04) and a SparkLAN WPEA-110N wireless card (Atheros AR9280). The access point is a Netgear WNDR3700 version 2 with two Atheros AR9280-based wireless cards running the Linux-based OpenWRT Backfire 10.03.1-RC5. On the access point and the UAV, two WiMo 18720.11 antennas are mounted, one being vertically and one horizontally mounted. For this test we used profiling based on wireless packet capture.

Using the presented methods, we were able to measure the received signal strength (RSS) and the achieved throughput in the described scenario. The distance and height values are obtained from the recorded GPS and IMU readings. As shown in Figure 4, RSS for the downlink (DL) decreases with the distance to the AP, but remains at an acceptable level. The high variance of the RSS is mainly because of the UAV's movements. The throughput follows the RSS and decreases with the distance. Throughput drops can be noticed during the UAV's acceleration phases. Note that the different receiver sensitivity of the wireless cards used on the AP and the UAV leads to a different baseline performance in the uplink and the downlink. While this paper focuses on how to acquire information about the wireless transmission channel, more details and additional results of real-world flights are presented in [7].

[11] http://www.tcpdump.org
[12] http://www.wireshark.org



(a) Received signal strength (b) Throughput versus distance

Fig. 4. Received signal strength and throughput.

5 Conclusion

Profiling the communication performance of UAVs connected via an 802.11 wireless network is an ongoing research topic. The main issues are that the UAVs are able to move arbitrarily in the 3D space and can change the orientation with respect to the ground station very fast. With the presented methods for measuring the 802.11 performance, we can get new insights into the factors influencing the communication performance of UAVs, as they enable profiling the performance in a fine grained manner.

Acknowledgments. This work was performed in the project Collaborative Microdrones of Lakeside Labs and was partly funded by the ERDF, the KWF, and the state of Austria under grant 20214/17095/24772.

References

1. Radiotap standard for 802.11 frame injection and reception. `http://www.radiotap.org`. last visited: March 2012.
2. The Linux wireless (IEEE-802.11) subsystem. `http://linuxwireless.org`. last visited: March 2012.
3. Chen-Mou Cheng, Pai-Hsiang Hsiao, H.T. Kung, and D. Vlah. Performance Measurement of 802.11a Wireless Links from UAV to Ground Nodes with Various Antenna Orientations. In *Proceedings of the 15th International Conference on Computer Communications and Networks (ICCCN 2006)*, pages 303–308, Oct 2006.
4. S. Crandall and H. Jasani. ProMix: Linux Promiscuous Wireless Packet Analysis. In *Proceedings of the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2011)*, pages 1–4, Sep 2011.
5. J. Salim, H. Khosravi, A. Kleen, and A. Kuznetsov. Linux Netlink as an IP Services Protocol. RFC 3549 (Informational), July 2003.
6. M. Vipin and S. Srikanth. Analysis of open source drivers for IEEE 802.11 WLANs. In *International Conference on Wireless Communication and Sensor Computing (ICWCSC 2010)*, pages 1–5, Jan 2010.
7. Evsen Yanmaz, Robert Kuschnig, and Christian Bettstetter. Channel measurements over 802.11a-based UAV-to-ground links. In *Proceedings of the GlobeCom 2010 (Wi-UAV Workshop)*, pages 1–5, Dec 2011.