Network Traffic Analysis of a Small Quadcopter

Stefano Munari, Claudio E. Palazzi, Giacomo Quadrio, Daniele Ronzani
University of Padua
Padua, Italy
stefano.munari.1@studenti.unipd.it, cpalazzi@math.unipd.it,
giacomo.quadrio@studenti.unipd.it, dronzani@math.unipd.it

ABSTRACT

In our work we analyze the network traffic generated by a small low-cost quadcopter. In particular, we consider the IEEE 802.11n protocol focusing on the live video streaming communication between the UAV and its controller in different outdoor scenarios. First we describe the network protocols of the drone and the testing methodology, then we analyze and evaluate the characteristics and the performances of the generated network. A clear distinction arises by comparing an uncongested network against a congested one. Different frequencies highlight quite distinct network profiles while various altitudes show to have a significant impact only on the signal degradation and not on the packet size distribution.

Keywords

UAV, network traffic analysis, QoS, video streaming

1. INTRODUCTION

Drones are an emerging trend in the technology industry. Faster deliveries, better traffic reports, and more effective search and rescue operations are just a few of the ways that UAVs could improve our lives. More recently, with the advent of low-cost drones, UAVs have acquired a recreational perspective.

In this work we test *Parrot Bebop*, an off-the-shelf quadcopter of 390g alimented by a 1200mAh lithium-ion polymer battery (11.1V). It can fly for ∼10min up to 250m from the controller, reaching the maximum speed of ∼47 km/h. Thanks to its built in Full HD camera, it is a perfect tool for making aerial footages and photos.

Parrot Bebop can be piloted with its specific remote controller (*Parrot Skycontroller*) or with any device which implements the 802.11 protocol through a dedicated application. Also, Parrot Bebop implements the 802.11n protocol, thus supporting an outdoor transmission range $\leq 250m$ both for 2.4GHz and 5GHz frequencies while providing a maximum nominal data rate of ∼300Mbps.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DroNet'17, June 23, 2017, Niagara Falls, NY, USA

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-4960-4/17/06…$15.00

DOI: http://dx.doi.org/10.1145/3086439.3086446

In our work an *outdoor WLAN* is considered. The UAV acts both as base station and server for the devices connected to it. Indeed, the network interface of the Parrot is configured in *master mode*, while the client interfaces are in *managed mode*, thus they can directly connect to the drone.

1.1 Contribution

This work analyses the aforementioned scenario by answering the following questions:

1. *What are the characteristics of the network traffic generated by an off-the-shelf drone?*
   Characterizing the data rates, packet sizes and inter-packet times will help to define and classify this kind of network.
2. *How does the network performance vary in different environments?*
   We experimentally select a set of parameters and test the drone in different parametrized environments. We analyze and compare them to figure out if and how the network change its behavior and which parameters affect most the network performances.
3. *How much traffic can the drone handle?*
   We incrementally increase the traffic load of the network to find its break point. This can directly affect the capacity of the network generated by the drone resulting in network congestion or affect the server resources resulting in a system crash.

The outcome of this analysis can be useful, for instance, to include realistic drone network traffic in models and simulations.

1.2 Organization

The rest of this paper is organized as follows. In Section 2 we explain the used terminology and concepts. In Section 3 we describe our methodology along with the test environments. In Section 4 we discuss the results of the analysis. To conclude, we summarized our work in Section 5.

2. BACKGROUND

UDP is the preferred transport layer protocol for multimedia real-time applications such as streaming services, online gaming, and any other application that requires quick delivery of every packet [1] [2] [3] [4]. For these kinds of services the transport layer does not handle congestion or flow control problems simply because it cannot detect them due to the nature of UDP [5] [6].

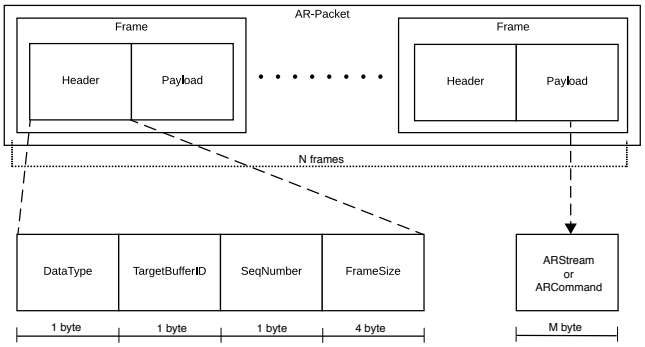


Figure 1: ARpacket format

However, these responsibilities are delegated to other specific protocols implementations at the application layer. RTP [7] is a paradigmatic example of the aforementioned kind of protocol. However, we are more interested in ARSDK protocols [8], the specific application layer solutions provided by Parrot Bebop.

2.1 ARSDK Protocols

These protocols are used by all Parrot drones. More generally, ARSDK is a framework to build Parrot compliant applications.

2.1.1 ARDiscovery

The drone uses Multicast DNS (mDNS) [9] during the discovery phase: it periodically sends beacon frames to inform of its presence other devices in its Wi-Fi range. When a controller (e.g. tablet, smartphone, etc.) wants to connect to the drone, the association phase starts. Since the connection is not protected, no authentication phase exists. The controller sends a connection request to the service port advertised in the mDNS packet. The request contains the identifiers and the port that the controller will use to receive the data. The drone replies specifying the max number of stream fragments for each video frame, the timeout for each stream fragment ack and other communication details.

2.1.2 ARNetworkAL & ARNetwork

Congestion control and flow control are handled at the application layer through the ARNetworkAL & ARNetwork protocol. The full block of data sent by this protocol is an application level packet (ARpacket). An ARpacket is composed of one or more frames (as depicted in Fig. 1).
Each frame is composed of a header and a payload and each payload can contain ARCommand (command) or ARStream (stream) data. Any frame header contains one of the following application-level data types:

- `Ack`;
- `Data`: normal data with no ack needed;
- `Low Latency Data`: high priority data (i.e. video streaming frames);
- `Data with Ack`: data with mandatory ack.

The live video streaming is enabled by default and disabled only in particular cases. However, during all the tests the live streaming has always been enabled. For Parrot Bebop each ARStream payload is *a single lossy compressed video frame* encoded with the H.264 codec [10]. Moreover, each ARStream can be identified in its header by the `Low Latency Data` type.

3. METHODOLOGY

3.1 Experiments

We performed 24 tests, *2 test sessions of 3 minutes* for each combination of the following parameters:

- Frequency - 2.4GHz, 5GHz
- Altitude - 17m, 34m, 51m
- Network load - uncongested, congested

Since 802.11n supports 2.4GHz and 5GHz we tested both of them expecting different results.
We know from the *inverse-square law* that the radio signal power decreases with the square of the distance. So, we calculated three different altitudes considering 51m as the threshold value for the absolute altitude. Also, we defined 30m as the threshold horizontal distance from the controller of the drone. As a result, in each test the UAV moves on a 2D plane at fixed altitude.

3.2 Measurement Testbed

In the following, we describe the hardware configuration of our testbed. It supports the PHY modes: 802.11 a/b/g/n.

- *Parrot Bebop Drone*
  - Processor: Parrot P7 dual-core, quad core GPU;
  - Wi-Fi antennas: MIMO dual-band with 2 double-sets of dipole antennas for 2.4 and 5 GHz;
  - Firmware: 1.98.10;
  - Sending power: Up to 21 dBm;
  - Signal range: Up to 250 meters;
  - OS: Linux Busybox.
- *iPad 3*
  - Processor: ARM Cortex-A9 dual-core @ 1.0 GHz with quad-core GPU;
  - RAM: 1GB;
  - WNIC: 2.4 and 5 GHz support;
  - Firmware modem: 6.1.00;
  - OS: iOS 9.3.5;
  - Application: FreeFlight Pro 4.1.14.
- *MacBook Pro 8.1*
  - Processor: Intel i5 dual-core @ 2.3 GHz;
  - RAM: 8GB;
  - WNIC: AirPort Extreme with 2.4 and 5 GHz support;
  - Firmware: Broadcom BCM43xx 1.0;
  - OS: macOS 10.12.3;
  - Application: Wireshark 2.2.4.

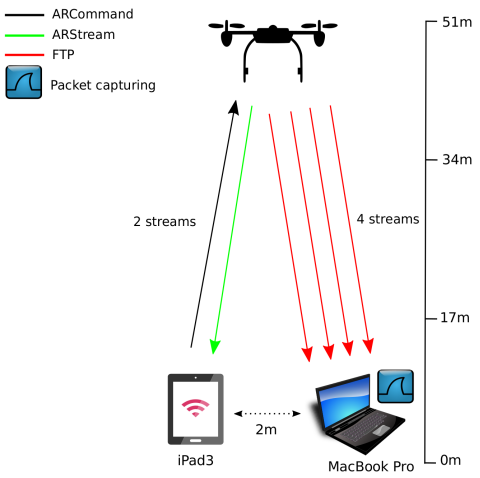


Figure 2: Measurement testbed - Congested environment

In each test environment the iPad 3 pilots the drone while the MacBook Pro passively captures network traffic through Wireshark [11], which is configured in RFMON (Radio Frequency MONitor) mode to sniff also packets sent from other clients to the UAV. Only in the congested environment the MacBook Pro is actively connected to the drone (Fig. 2). Basically, we deploy a single testbed configuration within which each test environment differs in the parameters. For example, the traffic load of the network clearly distinguishes the congested environment from the uncongested one.

Before performing the tests, we scanned the ports of the drone using nmap [12] and found several of them open. Surprisingly, the Parrot does not support multiple ARStream connections. So, we need another kind of traffic to generate a congested environment. Thus, we telnet into the UAV and create a binary file of 500MB which we use to test the QoS of the network. Also, with iwconfig [13] we were able to assess 65Mbps as the effective maximum bitrate of the network. Since other test environments are pretty similar and straightforward we will describe only the congested one.

3.3 Congested Environment

In this scenario the MacBook Pro is actively connected to the UAV (Fig. 2). Indeed, the wireless adapter of the MacBook Pro enables to transmit packets also in monitor mode.
The tests are performed using the Unix FTP client (macOS) and server (Linux Busybox) implementations as following: each 30s a new FTP connection is opened on the Parrot to download a 500MB file. Since the Parrot cannot handle more than 4 parallel FTP connections without causing piloting problems such as lost of control or complete streaming degradation for the iPad 3, we limit them to four (Fig. 3).

3.4 Limitations

All the tests have been performed outdoor, in open country, on sunny and windless days. We tried our best to limit known signal problems such as shadowing, reflection, diffraction and scattering by avoiding areas with obstacles (e.g. trees, houses, etc.). Since the wind speed increases with the altitude, it has not been possible to fly the drone over 51m

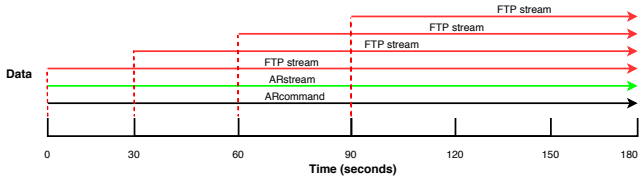


Figure 3: Congested environment - Data over time

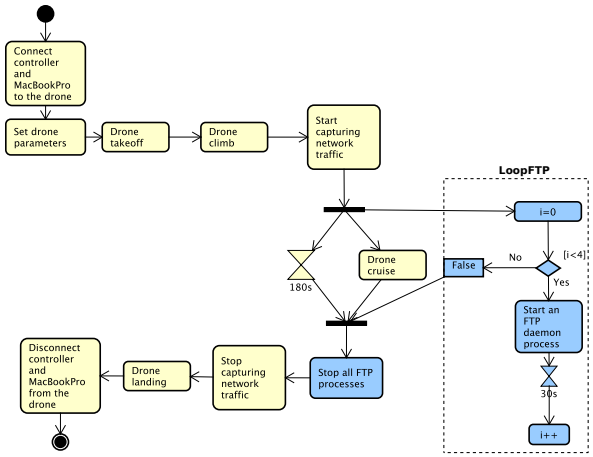


Figure 4: Testing process - *uncongested environment steps* in yellow, *congested environment steps* in yellow and blue

because of the serious stability and piloting problems that we experienced at higher altitudes. Moreover, each session lasted 3 minutes because of the limited battery autonomy of the drone (∼4min).

To detect disturbing 802.11 signals in 2.4GHz and 5GHz frequencies we first measured the interference using inSSIDer [14] and WiFiAnalyzer [15]. As expected, we found the most perturbed areas near the city centers.
Finally we have considered *congested environment* rather than a perturbed one for three reasons:

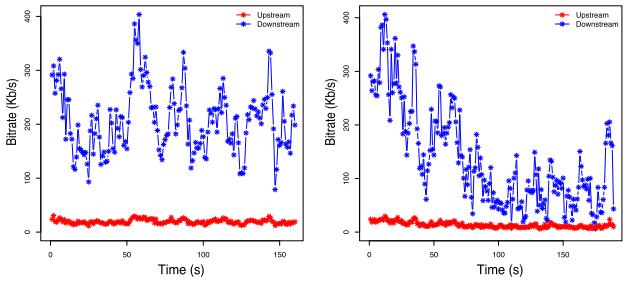
1. *Fine-grained control on each parameter* - since we directly manipulate the sources of congestion we can analyze and tune them, thus being able to reproduce the same conditions as needed.

2. *Network analysis* - we perform an *experimental* network stress test in a manner similar but less formal than the one described by Rowe [16]. Thus, we hope to obtain interesting results to clearly define how the traffic evolves over time and what are its break points (if any exists).

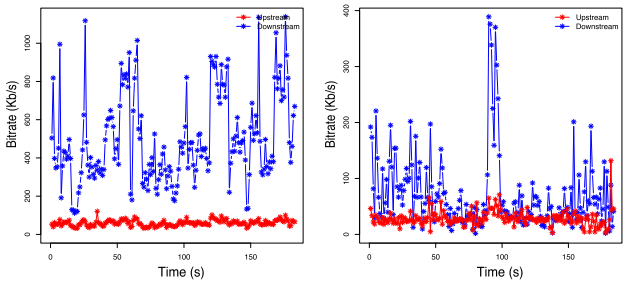
3. *Instrumentation* - a simple perturbed environment could lead to meaningless results because of the lack of instrumentation required to clearly define its interference ratio.

3.5 Testing Process

In the activity diagram depicted in Fig. 4 we summarize the steps involved in the testing process of the uncongested and congested environments.



(a) 2.4GHz - Uncongested (b) 2.4GHz - Congested



(c) 5GHz - Uncongested (d) 5GHz - Congested

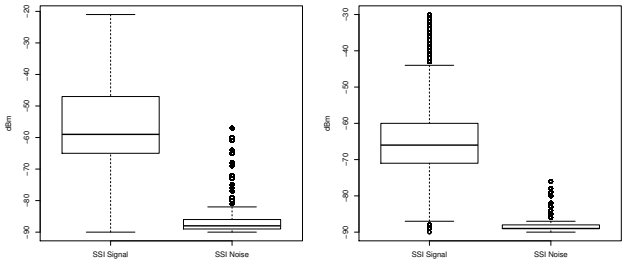
Figure 5: Bitrate - upstream (red) and downstream (blue) comparison between uncongested and congested environments. The graphs (a) and (b) refer to 2.4GHz while (c) and (d) refer to 5GHz.

4. ANALYSIS

Since each ARpacket can contain a different number of data frames, considering packets/s in this scenario could lead to imprecise results. Thus, in our analysis, we consider Kbps to measure the throughput of the link. Also, we mainly focus on the communication between the drone and the controller (iPad 3). The analysis includes UDP and IP headers, unless otherwise noted.

4.1 Network Traffic Characteristics

In Fig. 5 we compare the bitrate for the 2.4GHz and 5GHz frequencies (computed every second) in an *uncongested* and *congested* environment. The results show that most of the network traffic has been generated by the UAV (downstream). Indeed, the bandwidth usage is more limited but similar to thin-client based games systems [17]. If we consider that the behavior is specular, then this result is not surprising; a thin client (iPad 3) sends small packets (commands) to modify the state of the server (Parrot Bebop) while the latter handles the computation of the scenario (live video) and sends its results to the former (downstream).
Moreover, using Wireshark display filters, we were able to verify that nearly all the downstream traffic consists of ARStream packets (not shown in Fig. 5 because it overlaps with the downstream traffic making it less readable).
Since we set the WNIC in monitor mode, we were able to dissect the radiotap headers [18] injected in each frame.
These headers provide useful additional information on the communication such as data rate, antenna signal strength and the related noise.



(a) 2.4GHz (b) 5GHz

Figure 6: Radio frequency - comparison between signal and noise. The graphs (a) refers to 2.4GHz while (b) refers to 5GHz.

Thus, we observe mostly background noise during the performed tests. The boxplot (Fig. 6) shows that the signal of the 2.4GHz band is stronger than the 5GHz one. Indeed, the interquartile range (IQR) of the 2.4GHz band is higher than the 5GHz one. Also during the 51m tests, better responses were experienced from the 2.4GHz band due to its stronger radio signal. However, we can see that the background noise slightly perturbs the 2.4GHz while this is not the case for the 5GHz. Indeed, the SINR is better for the 5GHz band:

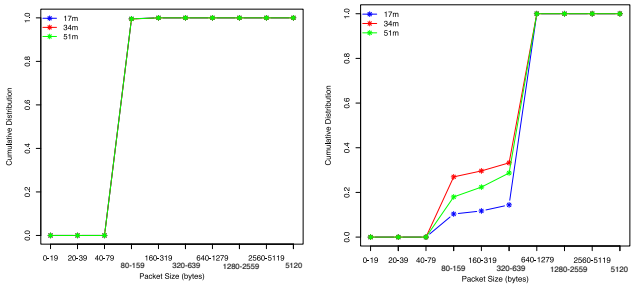
- $SINR(2.4GHz) = 0.65$
- $SINR(5GHz) = 0.74$

Also, the IQR is smaller for 5GHz frequencies which have a symmetric distribution around the median value. Note that the outliers shown in Fig. 6 are values which exceed the following bounds:

- upper bound = 3rd quartile+1.5×IQR
- lower bound = 1st quartile−1.5×IQR

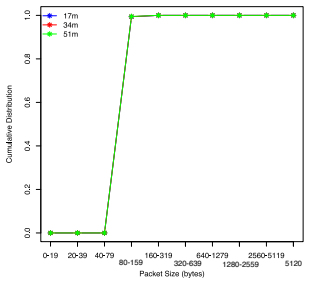
4.2 Network Congestion

In Fig. 5b and 5d we notice a degradation of the live streaming service over time. In both scenarios the ARNetwork protocol has not been able to recover from the congestion during the test time.
We know that the 802.11n protocol does not support EDCA and HCCA at the MAC Layer. Therefore, it cannot distinguish between the priority flows, thus depriving the video streaming of the needed bandwidth [19]. Indeed, during the tests, the controller starts slightly lagging after the second FTP connection while, in two of the performed tests, we completely lost the live video streaming after the fourth FTP connection (∼90s-100s clearly depicted in Fig. 5b). Moreover, by the end of each test, none of the FTP download has been completed. Also, we consider all the congested tests for the 51m altitude as failed because we temporarily lost the control of the drone (controller disconnected) after ∼100s. We notice that the application layer protocol (ARNetwork) uses an ack-based communication [8]. The downstream spikes in Fig. 5, which correspond to short bursts of ARStream fragments, unveil a very aggressive congestion control policy.

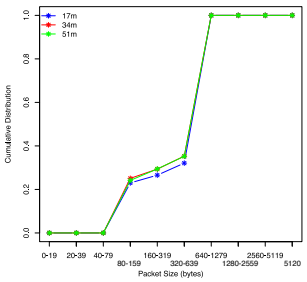


(a) 2.4GHz - Upstream

(b) 2.4GHz - Downstream

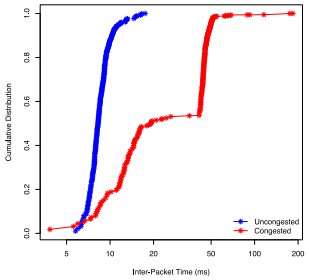


(c) 5GHz - Upstream

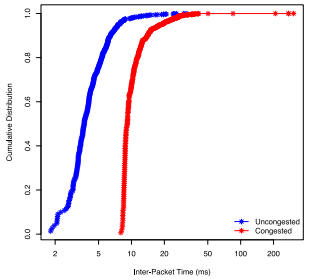


(d) 5GHz - Downstream

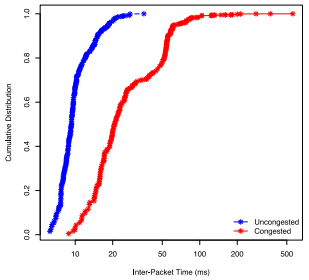
Figure 7: Packet size - comparison between different altitudes. The graphs (a) and (b) refer to 2.4GHz while (c) and (d) refer to 5GHz.



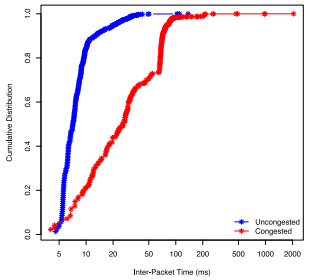
(a) 2.4GHz - Upstream



(b) 2.4GHz - Downstream



(c) 5GHz - Upstream



(d) 5GHz - Downstream

Figure 8: Inter-packet time - comparison between congested and uncongested environments. The graphs (a) and (b) refer to 2.4GHz while (c) and (d) refer to 5GHz.

Finally, comparing the two frequencies, the average downstream bitrate measured for 5GHz band is 1.66x of the one measured for the 2.4GHz band (Table 1) while for the upstream the ratio is 2.69:1.

4.3 Altitude

The packet size analysis (Fig. 7) shows the same upstream behavior for all the altitudes considering both 2.4GHz and 5GHz frequencies. These packets are really small compared to the downstream packets: 100% of the upstream ones cover the range of $[80, 159]$ bytes. For the downstream network traffic (Fig. 7b and Fig. 7d) the CDF displays more than 50% of the packets within the range of $[640, 1279]$ bytes while less than 40% of them are in the range of $[320, 639]$ bytes. Thus, we can infer that the *altitudes are not a significant parameter for the packet size distribution*. Only the Fig. 7b shows a slight variation between these parameters in the range of $[80, 639]$ for less than 40% of the total downstream traffic considering the 2.4GHz frequency.

4.4 Inter-packet time

Fig. 8 compares the inter-packet time (Δ) CDFs for uncongested and congested environments. We define Δ as the time between each UDP packet arrival. As expected, the Δ of congested traffic is larger than the uncongested one.

Considering 2.4GHz frequencies (Fig. 8a and Fig. 8b), less than 80% of the uncongested upstream traffic shows a Δ of 10ms while, for the same percentage, the downstream one registers 5ms. The gap is even larger if we consider the congested traffic, for the same percentage the ratio is greater than 3:1 (50ms for upstream against 15ms for downstream).

Also a Δ of 20ms is observed for less than 50% of the upstream traffic while the same value holds for 90% of the downstream one.

Not considering the outliers, in Fig. 8c and Fig. 8d the trendlines look similar with a slightly bigger gap for the downstream traffic: for less than 60% of the upstream traffic the difference between congested and uncongested is $\sim$10ms while for the downstream traffic it is $\geq$15ms. Finally, we note that only the congested streams are affected by the outliers. This could indicate periods of lagging video.

Table 1 provides summary statistics of the network analysis, showing both upstream and downstream traffic grouped by frequency. The overall traffic is quite asymmetric, with downstream having at least 6x higher bitrates and 8x larger packets. For 2.4GHz there are more packets per second than the 5GHz. Also, the gap between the average inter-packet time values is more emphasized on the 2.4GHz band.

4.5 Security concerns

Although security is not the main focus of this work, nonetheless our analysis revealed some vulnerabilities; some of them could affect even other off-the-shelf drones.
In essence, the drone's network interface has several open ports, which could be used by a malicious user to telnet into the UAV and shutdown the system or use FTP to transfer malicious code to be executed [20]. Enforcing network authentication along with proper OS hardening [21] like closing open ports, avoid running services if not needed and setting firewall rules, will improve the drone's security.

Table 1: Network metrics - Summary

	2.4GHz		5GHz	
Metrics	UP	DOWN	UP	DOWN
Avg bitrate (Kb/s)	16.37	171.65	44.06	284.57
Pkt size (bytes)	[80,159]	[640,1279]	[80,159]	[640,1279]
Avg inter-pkt time (ms)	13.62	6.42	15.00	16.36

5. CONCLUSION

In this work we classify the network traffic generated by a small quadcopter (Parrot Bebop). We test the drone in environments which differ on three parameters: *frequency*, *altitude* and *network load*. The resulting network traffic is *highly asymmetric* and *interference sensitive* because of the live video streaming in the drone-controller communication and the wireless outdoor scenario.
Among all, we recognize the *network load* as the most significant parameter. However, the comparison between 5GHz and 2.4GHz arises the following considerations:

- not perturbed areas are probably an unrealistic scenario for drones;
- experiments gave better results for 5GHz (Table 1).

So, 5GHz seems a reasonable solution for not perturbed areas. But despite the fact that 2.4GHz is a crowded frequency, the choice of 5GHz band is not straightforward for perturbed environments. Indeed, most urban centers have lots of buildings and obstacles which could lead to the major problem of losing the drone-controller connection.
The choice to avoid using standard application level real-time protocols such as RTP to build the streaming service has lead to undesired consequences. *The drone is not able to handle and recover from a congested environment* (4 FTP streams + 1 ARStream).

In conclusion, we suggest to improve the application level network protocol using RTP as base and enforcing network security as discussed. Hopefully, this will mitigate most of the serious problems which affect Parrot Bebop. Furthermore, these considerations can be applied also to other off-the-shelf drones.

6. REFERENCES

[1] M. Roccetti, S. Ferretti, and C. E. Palazzi, "The brave new world of multiplayer online games: Synchronization issues with smart solutions," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, May 2008, pp. 587–592.

[2] D. Maggiorini, C. Quadri, and L. A. Ripamonti, "Opportunistic mobile games using public transportation systems: a deployability study," *Multimedia systems*, vol. 20, no. 5, pp. 545–562, 2014.

[3] C. E. Palazzi, A. Bujari, G. Marfia, and M. Roccetti, "An overview of opportunistic ad hoc communication in urban scenarios," in *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, June 2014, pp. 146–149.

[4] S. Cacciaguerra, S. Mirri, P. Salomoni, and M. Pracucci, "Wandering about the city, multi-playing a game," in *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.*, vol. 2, Jan 2006, pp. 1214–1218.

[5] L. Donatiello and M. Furini, "Ad hoc networks: a protocol for supporting qos applications," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International.* IEEE, 2003, pp. 8–pp.

[6] C. E. Palazzi, A. Bujari, and S. Mirri, "Reducing queuing delays through voap," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2016, pp. 1–3.

[7] R. F. H. Schulzrinne, S. Casner and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Requests for Comments, RFC Editor, RFC 3550, July 2003. [Online]. Available: https://tools.ietf.org/html/rfc3550

[8] (2015) Parrot for developers - arsdk protocols. [Online]. Available: https://developer.parrot.com/docs/bebop/ARSDK_Protocols.pdf

[9] S. Cheshire and M. Krochmal, "Multicast DNS," Internet Requests for Comments, RFC Editor, RFC 6762, February 2013. [Online]. Available: https://tools.ietf.org/html/rfc6762

[10] ITU-T, "Advanced video coding for generic audiovisual services," International Telecommunication Union, Recommendation, October 2016.

[11] (2015) Wireshark project. [Online]. Available: https://www.wireshark.org

[12] (2017) Nmap. [Online]. Available: https://nmap.org/

[13] (2017) iwconfig - manual page. [Online]. Available: https://linux.die.net/man/8/iwconfig

[14] (2017) inssider. [Online]. Available: http://www.metageek.com/products/inssider/

[15] (2017) Wifianalyzer. [Online]. Available: https://play.google.com/store/apps/details?id=com.vrem.wifianalyzer

[16] M. Rowe, "Server stress testing using multiple concurrent client simulation," Nov. 27 2001, uS Patent 6,324,492. [Online]. Available: https://www.google.com/patents/US6324492

[17] M. Claypool, D. Finkel, A. Grant, and M. Solano, "Thin to win? network performance analysis of the onlive thin client game system," in *Proceedings of NetGames 2012*. IEEE Press, 2012, p. 1.

[18] (2017) Radiotap. [Online]. Available: http://www.radiotap.org/

[19] A. Malik, J. Qadir, B. Ahmad, K.-L. A. Yau, and U. Ullah, "Qos in ieee 802.11-based wireless networks: a contemporary review," *Journal of Network and Computer Applications*, vol. 55, pp. 24–46, 2015.

[20] J.-S. Pleban, R. Band, and R. Creutzburg, "Hacking and securing the ar. drone 2.0 quadcopter: investigations for improving the security of a toy," in *IS&T/SPIE Electronic Imaging*, 2014.

[21] W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007.