information

MDPI

Article

On the Use of Affordable COTS Hardware for Network Measurements: Limits and Good Practices †

Eduardo Miravalls-Sierra, David Muelas *, Jorge E. López de Vergara, Javier Ramos and Javier Aracil

High Performance Computing and Networking Research Group, Departamento de Tecnología Electrónica y de las Comunicaciones, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Fco. Tomás y Valiente, 11, 28049 Madrid, Spain; eduardo.miravalls@uam.es (E.M.-S.); jorge.lopez_vergara@uam.es (J.E.L.d.V.); javier.ramos@uam.es (J.R.); javier.aracil@uam.es (J.A.)

* Correspondence: dav.muelas@uam.es

† This paper is an extended version of our paper published in XIII Jornadas de Ingeniería Telemática (JITEL 2017), "Evaluación de equipamiento de bajo coste para realizar medidas de red en entornos domésticos".

Received: 15 January 2018; Accepted: 19 February 2018; Published: 22 February 2018

Abstract: Wireless access technologies are widespread in domestic scenarios, and end users extensively use mobile phones or tablets to browse the Web. Therefore, methods and platforms for the measurement of network key performance indicators must be adapted to the peculiarities of this environment. In this light, the experiments should capture the true conditions of such connections, particularly in terms of the hardware and multi-device interactions that are present in real networks. On the basis of this, this paper presents an evaluation of the capabilities of several affordable commercial off-the-shelf (COTS) devices as network measuring probes, for example, computers-on-module or domestic routers with software measurement tools. Our main goal is to detect the limits of such devices and define a guide of good practices to optimize them. Hence, our work paves the way for the development of fair measurement systems in domestic networks with low expenditures. The obtained experimental results show that these types of devices are suitable as network measuring probes, if they are adequately configured and minimal accuracy losses are assumable.

Keywords: wireless networks; network monitoring; performance evaluation; quality of service; COTS

1. Introduction

Wireless Internet access technologies have completely revolutionized how end users connect with each other. In this regard, WiFi technology has emerged as one of the most popular options to provide low-cost and high-bandwidth (BW) wireless connectivity to end users, particularly in environments that highly depend on mobility. This has awakened interest in studying the factors that negatively affect wireless communications to design and deploy protocols and technologies that perform better towards connecting everything.

Particularly, wireless technologies pose several security risks and performance challenges. In fact, intermittent performance issues are common and usually diminish the quality of service (QoS) and quality of experience (QoE) of many end users. Moreover, users employ a broad variety of devices to connect to the Internet, for example, mobile phones or laptops, or perhaps a small board ARM computer such as a Raspberry Pi or ODROID. Such devices may present further hardware or software restrictions, which may limit the kind of tasks that can be performed.

In this light, Internet Service Providers (ISPs), public institutions and end users are increasingly concerned about the QoS in these scenarios. Therefore, the development of measurement mechanisms

that gather fair QoS estimations deserves much greater attention, and at the same time, they must not entail high deployment and operation costs to result as useful.

Hence, multiple initiatives have proposed the deployment of measurement frameworks that follow these principles—see, for instance, the architectures of Project Bismark [1], RIPE Atlas or SamKnows [2]. Their main objective is to model how Internet connections perform from the standpoint of domestic end users. To do so, these projects typically rely on small, cheap hardware probes implemented with commercial off-the-shelf (COTS) routers flashed with OpenWRT (https://openwrt.org/) or ARM devices such as Raspberry Pi or ODROID. These initiatives aim to provide a standard measurement framework that is affordable to deploy in a wide variety of scenarios and is able to perform both active and passive network measurements.

However, the empirical evaluation of network deployments is severely constrained when testing with few clients. Furthermore, WiFi deployments suffer from important performance degradation issues when many clients are present, as a consequence of how low-level protocols regulate the access to the shared medium [3]. Thus, this matter is another key factor alongside capacity planning and quality assurance processes in wireless scenarios. Consequently, the consideration of testbeds that represent the interactions among the expected number of clients and the hardware that is present in operational access networks is a necessity.

To sum up, these facts expose that the performance evaluation of wireless scenarios must consider how access networks are used. Keeping in mind the practical issues that this entails, we present a systematic performance evaluation of some such low-end devices for network monitoring purposes. Our main contributions are twofold. First, we identify the main characteristics of low-end devices that affect measurements and how they interact with standard software tools that are commonly used to evaluate network performance. Second, we provide a set of figures that conform an empirical assessment of the expected practical bounds of COTS devices during QoS evaluation in domestic environments. To do so, we select some typical cases of network measurements using common software tools on top of affordable COTS platforms.

Our findings define a guide of good practices that must be taken into account by both researchers and practitioners if small low-cost hardware probes are used. Specifically, we detect how such equipment can limit network measurements and give some corrective strategies able to overcome those limits. As a result, we pave the way for the development of testbeds that can empirically evaluate how deployments will perform when many clients are present while keeping down the cost, that is, by adding affordable hardware probes that emulate active users.

The rest of the article has the following structure. Section 2 reviews previous works that describe how network key performance indicators (KPIs) should be measured and defined. With this, we justify the typical cases our empirical evaluation includes. Then, in Section 3, we describe the experimental setup (i.e., scenarios under test and methods) and report our empirical results. Afterwards, in Section 4, we extract the main findings of our empirical study and establish a guide to improve the outcomes of network measurement systems making use of low-cost COTS platforms. Finally, in Section 5, we draw the conclusions of our work and outline some future lines of work.

2. Network Performance Measurements

We now discuss some fundamentals related to the measurement of network KPIs. We start with a review of several works that analyzed methodological and practical aspects that affect network measurements. Although these aspects apply when measuring any kind of network, whether it is wireless or not, our discussion links them to some specific issues of the latter case.

Next, we present several usual definitions of network KPIs, which include those that we considered in our experiments—namely, BW, delay, jitter and packet losses. The selection of these KPIs is rooted in their typical use by network administrators or ISPs to gauge whether their network is working properly or not.

We note that this section aims to provide a general view of the variety of methodological constraints, definitions and parameters that are present in network measurement processes. Hence, and for the sake of brevity, we have restricted this review, aiming to achieve a good trade-off between completeness and utility for the empirical study in the following sections.

2.1. Measuring Methodology

A recurrent problem when trying to measure network performance parameters is the reliability of available measurement tools, as these widely differ in their reported results. On the one hand, the obtained results may depend on the data collection technique used. On the other hand, the specific definition of the estimation of metrics can be somehow ambiguous, which may lead to confusion among different computations for a given metric.

The ways in which to measure network performance have resulted from considerable efforts by both the research community and industry. Initiatives such as *Internet Protocol performance metrics* (*IPPM*) [4] or standards such as the *two-way active measurement protocol* (TWAMP) [5] aim to establish a set of good practices and define clear metrics and methodologies for network measurements. Their final goal is to provide users with methodologies to obtain clear, objective and reproducible results. In this light, they try to minimize the non-reproducibility of experiments due to, among other aspects, factors not accounted for in statistical models, errors in experimental design, and so forth.

However, recent studies have shown that important deviations on the estimated KPIs can appear when using different measurement tools. For instance, the authors of [6] showed the differences in the statistics reported by several popular BW measurement tools in an extensive crowdsourced dataset. Furthermore, most experiments seem to disregard factors that have been proven to severely affect the measurements, such as CPU load or CPU scheduling policies [7]. Hence, the results that they provide may lack accuracy or may even be unreproducible, with the consequent limitation of the conclusions that they offer.

Moreover, wireless environments exhibit other factors that introduce further uncertainty in the measurements. Wireless links typically tend to produce random losses and increments in latencies, which are hard to predict and reproduce because of their multifactorial nature. Some recent studies have tackled the task of modeling random delay produced by wireless links, such as in [8,9]. To do so, these proposals usually rely on simple models such as decision trees or random forests, as these are easily understandable and generalizable.

Additionally, shared channels often produce performance problems once multiple users compete for them [10]. In fact, there are hidden interactions between the upper-layer protocols and the medium access protocols that may cause performance degradation, as a result of typical access policies that they consider. For instance, a Transmission Control Protocol (TCP) implementation may start retransmitting a large amount of data because of jitter caused by collisions at the wireless physical layer. Such retransmissions may be unnecessary (spurious retransmissions), as data is not really lost but is only delayed above the typical TCP timeouts.

As a result, the performance evaluation of a network that is shared by a several users should consider all these methodological factors to provide accurate conclusions. This motivates the utilization of (i) low-cost measurement platforms, to alleviate the cost of data gathering in diverse locations and with several clients; (ii) low-level tuning, to control the effects of software–hardware interactions in measurement elements; and (iii) standard definitions of metrics, instead of ambiguous ad hoc implementations.

2.2. Bandwidth

The BW of a channel measures the amount of information per unit of time that is capable of transmitting (i.e., in bits per second). However, BW can be understood in different manners, depending on the specific underlying parameter [11,12].

On the one hand, it may be used as a synonym for the *capacity* of a path. Given an end-to-end path consisting of a series of n ordered links $i = 1, ..., n$, we define the capacity of link i as the maximum transmission rate at the IP level, C_i. Thus, the capacity of the path, C^*, is defined as the minimum of the capacities $\{B_i\}_1^n$ of each of the individual links:

$$C^* = \min_{i=1,...,n} \{C_i\} \tag{1}$$

The links i_k such that $C_{i_k} = C^*$ are called the *narrow links* of the path. We note that more than one link may be the bottleneck.

On the other hand, the BW of a path may refer to the *available BW* of a path, which is the unused capacity of a channel in a specific moment in time. This metric is complementary to the current used BW: given a utilization factor $u_i^t \in [0, 1]$, the available BW at time t of link i is $A_i^t = C_i(1 - u_i^t)$, and thus

$$A_t^* = \min_{i=1,...,n} C_i(1 - u_i^t) \tag{2}$$

Thus the available BW of a path depends on C^*, the amount of traffic passing through it, and the number of competing clients—which is particularly notorious in wireless scenarios. The link i_k such that $A_{i_k} = A^*$ is called the *tight link*. This instantaneous measure is usually reported as averaged over a time interval $[t, t + \tau]$:

$$\overline{A}^*(t, t + \tau) = \min_{i=1,...,n} C_i \left(1 - \overline{u}_i(t, t + \tau)\right) \tag{3}$$

Finally, the *bulk transfer capacity* (BTC) is the maximum amount of data per unit of time that a protocol that implements a congestion control algorithm, for example, TCP, can send. This metric is affected by several factors [7], such as the number of concurrent TCP sessions or the amount of User Datagram Protocol (UDP) interfering traffic, among others. To carry out BW measurements, we can follow either an active or passive approach. The active approaches are affected by the transport protocol, and depending on which is used, the measurements may report different parameters. For example, while the packet train technique [7], which uses UDP, can accurately measure the capacity C^* of a path, measurements with TCP traffic provide estimations of the BTC. Passive approaches rely on monitoring the BW usage by applications or hosts, thus accounting for the transmitted bytes per unit of time.

2.3. Delay

Delay is the amount of time that it takes a byte to travel the distance from the moment it left a host until it reaches its destination. Delay may refer either to *one-way delay* (OWD) [13], or to *round-trip time* (RTT)—the sum of both OWDs between a pair of hosts.

OWD estimation is challenging, as it requires very accurate and synchronized clocks at both ends of the connection. Therefore, RTT measurements are usually preferred, as they bypass this problem—we note that in this case, the involved times can be gathered from the same clock. Assuming that both latencies are of the same order (symmetric paths), the OWD can be estimated as half the RTT between a request and a reply, as suggested in protocols such as Q4S [14].

A common way of estimating the RTT of a path is by actively measuring it using mechanisms such as `ping` [15]. Measuring the RTT passively is harder, as it requires knowledge of the upper-layer protocols on top of TCP to correctly pair requests with replies. However, Nagle's algorithm or the *delayed acknowledgment (ACK)* implemented in TCP make this task difficult, as replies may have additional delays, which leads to overestimations. Despite these limitations, we can still measure the RTT by looking at the timestamps of the packets sent during the three-way handshake of a TCP session. As depicted in Figure 1, we can estimate the RTT by subtracting the times between the segments with the TCP synchronize (SYN) and ACK control flags set:

$$RTT = t_{ACK} - t_{SYN} \tag{4}$$

or between the SYN, ACK and the client SYN:

$$RTT = t_{SYN,ACK} - t_{SYN} \tag{5}$$

Depending on the point in the path our network probe is measuring, we can chose between one or the other. If we are closer to the server, we should use Equation (4). If we are closer to the client, we should use Equation (5) instead.

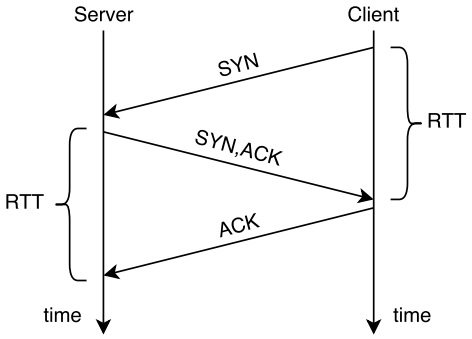

Figure 1. Measuring round-trip time (RTT) in a three-way handshake of the Transmission Control Protocol (TCP).

2.4. Variability of Latency or Jitter

Jitter is a critical performance indicator because of the impact it exerts on the quality of multimedia applications such as Voice over IP (VoIP). As this KPI provides an estimation of how network latency varies, there are several definitions depending on how this variability is measured. The definition we have used in our experiments follows that in [16] and equivalently in [14]. Given $N+1$ OWD measurements, $\{l_i\}_{i=0}^{N}$, we compute the N pairwise differences $\{\Delta_j\}_{j=1}^{N}$:

$$\Delta_j = |l_j - l_{j-1}| \tag{6}$$

and define jitter using a statistic of the centrality of the $\{\Delta_j\}$, such as the mean or the median.

Other definitions of jitter are given as the result of an exponential filter [16] of the $\{\Delta_j\}$ with parameter 1/16, or by computing the standard deviation of the $\{l_i\}$.

2.5. Packet Loss

Packet loss is an indicator of network saturation, as both routers and hosts drop packets when they receive packet rates above their processing capabilities. Additionally, packets can be dropped when bit errors occur, as a consequence of either hardware errors or random noise—the later is a frequent issue in wireless communications. Measurement protocols such as Q4S or IPPM [17] use sequence numbers to estimate packet loss with UDP traffic, the same approach that uses TCP to implement its reliable transfer capability. Packet loss is computed as the ratio of non-received packets over the total expected number.

3. Experimental Evaluation

We now describe the main aspects of our experimental evaluation. First, we describe the hardware that we used, which was representative of common low-cost COTS platforms. After that, we comprehensively describe the network KPIs' measurement methodology, which aims to detect the shortcomings that may arise in general scenarios making use of standard tools and metric definitions. Finally, we report the measured values for each scenario under test.

3.1. Testbed Description

The following equipment was used in our experiments:

- Two TP-Link Archer C7 AC 1750 routers with a multiple-input and multiple-output (MIMO) 3 × 3:3 configuration (three transmitting antennae, three receiving antennae, and three spatial streams). One of these kept the stock firmware (3.15.1 Build 160616 Rel.44182n), and we refer to it as "TP-Link". The second was flashed with a dd-wrt (http://dd-wrt.com/site/index) Linux distribution, and we refer to it as "dd-wrt". The estimated price of these routers was 80 euros each.
- PC1: Desktop PC with an Intel Core i7 CPU at 2.80 GHz, 8 GiByte of DDR3 RAM at 1333 MHz and two network interfaces: the integrated interface was connected to the Internet, and a second interface was used to connect to the tested network, a PCI Broadcom Corporation NetXtreme II BCM5709 Gigabit Ethernet (rev 20) network card. The operating system was an Ubuntu 14.04 Mate. The estimated price of this hardware was 600 euros.
- PC2: Desktop PC with an Intel Core i7-2600 CPU at 3.40 GHz, 8 GiByte of RAM DDR3 at 1333 MHz, and one (integrated) Intel Gigabit Ethernet network card. The operating system was a CentOS 7, and the estimated price of the hardware was 550 euros.
- ODROID C2 [18] with a USB Edimax EW-7811UN 802.11n WiFi adapter with a maximum throughput of 150 Mbit/s. This kit had an approximate price of 100 euros.

Our wired scenarios were as follows:

- Scenario 1: As shown in Figure 2a, we connected two hosts directly with an ethernet cable, as a baseline of an ideal direct connection to the server without wireless hops.
- Scenario 2: As shown in Figure 2b, the purpose of this scenario was to obtain an idea of the capabilities the dd-wrt router has as a measuring device, keeping in mind that it has limited resources versus a PC.
- Scenario 3: As mentioned before, small form-factor ARM-based computers such as ODROID have made it to the mass market. Thus, it is interesting to find out whether they are suitable as cheap measuring devices. As illustrated in Figure 2c, in this scenario, we connected PC1 to an ODROID C2 and benchmarked the communication performance.

The wireless scenarios were as follows:

- Scenario 4: In this scenario, we connected both PCs with a WiFi link, as shown in Figure 2d. The TP-Link router acted as an access point (AP), while the dd-wrt acted only as a WiFi client that directly connected PC2 to PC1. In this scenario, we used the 2.4 GHz band with a 40 MHz channel with a 22 modulation and coding scheme (MCS) index in transmission and a 23 MCS index in reception, measured with `iw`.
- Scenario 5: In this scenario, the network topology was the same as in the previous. However, in this scenario, the dd-wrt took the active role as the measuring device, and PC2 acted only as a mere terminal to run commands through SSH. Our rationale was that because the commands went through a different link, they should have interfered less with our measurements.
- Scenario 6: In this scenario, we measured what an average user would experience if they used a cheap USB WiFi dongle with a 7 MCS index connection.
- Scenario 7: In this scenario, we connected the PC2 directly to the AP using the USB WiFi dongle, as illustrated in Figure 2f.
- Scenario 8: In this scenario, we used the same network topology as is depicted in Figure 2d, but this time with a 5 GHz WiFi link, with an 80 MHz channel width and a MCS index of 8.
- Scenario 9: In this scenario, we maintained the topology of Scenario 8, but measured from the dd-wrt instead. Again, we used the PC2 only as a terminal to issue commands that interfered with the measurements as little as possible.

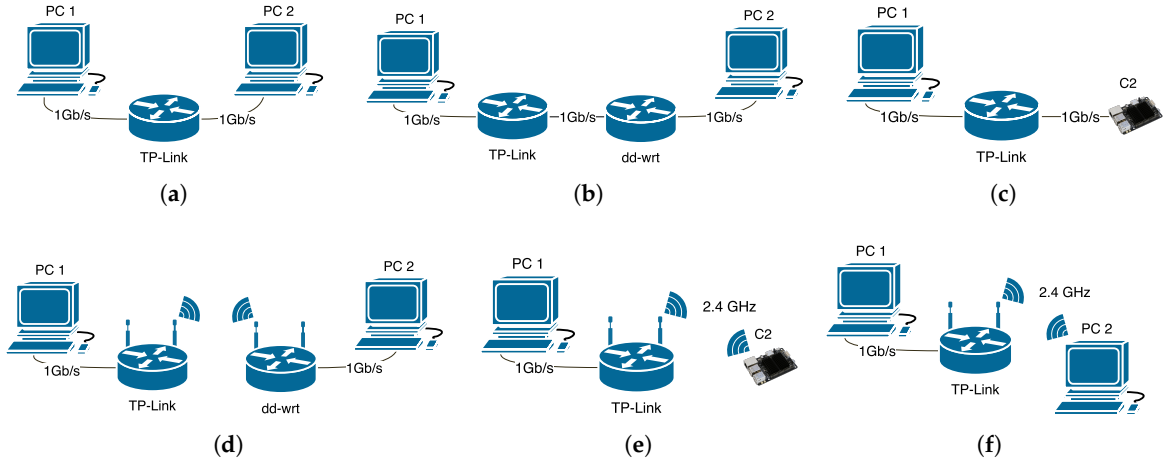

Figure 2. Network topologies for the considered scenarios. (**a**) Scenario 1; (**b**) Scenario 2; (**c**) Scenario 3; (**d**) Scenarios 4 and 5 with a 2.4GHz link, 8 and 9 with a 5 GHz; (**e**) Scenario 6; (**f**) Scenario 7.

In all our experimental scenarios described before, the PC1 and the router TP-Link acted as the "server", and the other hardware involved was the "client". In our discussion, we denote the server-to-client data flow as "downlink" and the reverse data flow as "uplink". We remark that our experiments were intended to measure how a single client performed in these scenarios, and thus all communication channels were completely available for the client.

Hence, these results reflect the maximum performance figures that a single client could achieve, as an initial estimation of how these setups perform. However, further testing using multiple concurrent clients is required in order to account for the effects of competition for the same channel.

In this light, novel wireless technologies are under development, improving the performance characteristics they can offer. Furthermore, new protocols to regulate access to shared media are continuously under development [19], and their effects on upper-layer protocols are also attracting the attention of the research community [20].

3.2. Method Description

To perform our experiments, we used widespread software tools to measure network KPIs: `iperf3` to measure BW, packet loss and jitter; and ping to measure delay, as `iperf3` does not report it.

Specifically, `iperf3` is one of the de facto standard tools to actively measure network KPIs. We note that `iperf3` works differently depending on the selected transport protocol—see the discussion on BW measurements in Section 2. We recall that measurements on top of UDP provide an estimation of path capacity by sending packets at a configured rate. The implications of this method are twofold. First, this entails that the packet rate must be adjusted to a rate near the capacity of the path. Otherwise, BW estimations will suffer from the effect of high packet-loss ratios. Coherently, we estimated the maximum transmission rate of `iperf3` such that packet losses were kept as close as possible to zero, before recording the measurements of a scenario. Second, estimations in the client side do not reflect the true capacity of the path, as the sending rate remains constant. Hence, only server side measurements could be used to characterize the network performance. Finally, another critical factor of UDP-based measurements is that the packet size must be adjusted to prevent fragmentation at the network layer; otherwise, the loss of a single fragment causes the loss of the whole UDP datagram, which degrades the estimated KPIs.

Additionally, `iperf3` offers the possibility of sending the data in multiple streams. However, the multiple streams do not offer any performance improvement if the limiting factor is the CPU, as it is a single-threaded application—in fact, the measured BW is divided equally among the streams.

Preliminary experiments showed that this functionality did not affect the measured BW in our testbed; thus we executed all our trials with a single stream.

In order to minimize CPU scheduling effects in the measurements, we fixed the affinity of `iperf3` to a single CPU core in both the client and the server whenever it was possible—this optimization can be applied in multi-core systems, although the TP-Link Archer C7 AC 1750 featured only one core. Furthermore, these CPU cores were isolated at boot, as explained in [21]. Our commodity hardware only had a single Non-Uniform Access Memory (NUMA) node per device, but in complex machines, the selected CPU core should be that directly connected to the network interface card (NIC) [21].

With the default configuration, each execution of `iperf3` transmitted over 10 s, trying to achieve as much throughput as possible. Then, `iperf3` reported its statistics for each second. We executed `iperf3` 10 times per scenario, for a total of a 100 samples. Afterwards, we sent 100 consecutive `pings` per direction to estimate the delay of the measured link. The CPU usage was low (below 20%), and thus it was not a limiting factor in our experiments. In all our tests, we tried to minimize our lost packet rate, and we aimed to keep this below 1%.

In Table 1, we summarize the different metrics we computed and which tool was used in each case. We note that we also reproduced `iperf3` TCP measurements with those provided by `nttcp`. As we found that none of the results reported by either tool presented significant deviations, we have omitted the latter for the sake of brevity. We note that although `ping` would seem a valid estimator of lost packets, the lost packets it reports are different than those of `iperf3`; `iperf3` continuously sends a high volume of packets, while `ping` sends them sporadically, to the default value of 1 probe/s—the `busybox` that comes with the dd-wrt did not allow us to configure it; thus we kept it as it was in all our tests for consistency. Thus `ping` can be used as an instantaneous measure of connectivity [22], and in order to estimate the link's approximate loss rate, a large number of probes must be considered [14], as random sporadic losses are expected.

Table 1. Measured key performance indicators (KPIs) and tools used in each case.

Tool	Protocol	BW	Delay	Jitter	Losses
`iperf3`	Transmission Control Protocol (TCP)	✓	✗	✗	✓
`iperf3`	User Datagram Protocol (UDP)	✓	✗	✓	✓
`ping`	Internet Control Message Protocol (ICMP)	✗	✓	✗	∼

In our experiments, we connected two devices to the network under test, where one acted as a server and the other acted as a client for the measurements. The client transmitted the data, and the server gathered the estimations of the network KPIs. Then, both devices switched roles, and we measured the opposite direction. We measured both wired links (as the baseline) and WiFi links, both in the 2.4 and 5 GHz bands. WiFi measurements may be affected by any other wireless sources in the vicinity, and in our tests, the building had other WiFi networks in the same bands—we performed the evaluation in a university laboratory. As this factor cannot (generally) be controlled in domestic environments, we did not isolate our tests from these exogenous effects. We note that this context may be somewhat optimistic, but it still represents the typical operation of real probe deployments.

Finally, we remark that we report the values obtained after applying all the aforementioned configurations. Hence, the results that we report represent the bounds for network KPI measurements that may be expected when using this type of equipment.

3.3. Results

In Figure 3, we show the results obtained for each scenario. Overall, we note that the available BW for a TCP client in the downlink direction was higher than in the uplink direction.

In Scenario 2, we note that `iperf3` in dd-wrt generated TCP traffic twice as fast as it could process it. We also note that it could not receive more than 10 Mbit/s of UDP traffic. However, this is a known bug of version 3.1.x [23,24]. This bug also affected Scenarios 5 and 9, which showed the same limitation.

Furthermore, more performance problems using UDP could be seen in the ODROID results, which suggests that UDP traffic is handled more inefficiently than TCP traffic. Regarding UDP measurements, special care was taken so that no IP fragmentation occurred, as this greatly increased the number of lost packets in the preliminary experiments.

Comparing the BW results for Scenarios 6 and 7, there was a noticeable drop in performance. We suspect that some driver or hardware issues were affecting the experiments of the PC2, as the adapter was the same in both scenarios.

Looking at Scenarios 8 and 4, we notice a 50% increase in all the TCP measurements. However, TCP could not achieve maximum throughput due to random losses, even if the 5 GHz channel could reach up to a 440 Mbit/s rate, as seen in Scenario 9.

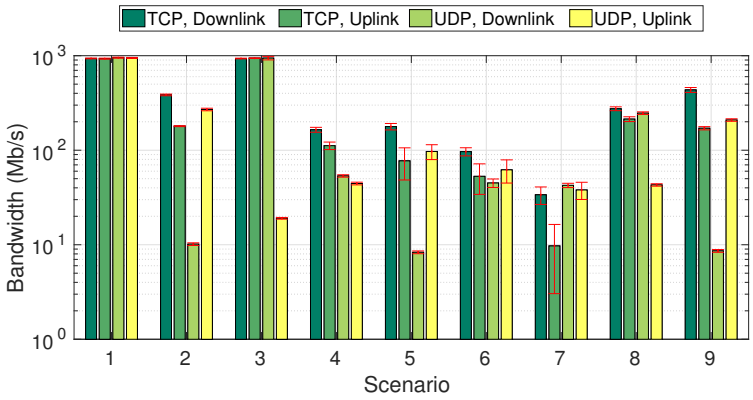

Figure 3. Bandwidth measurements in megabits per second. Bars show average results, with whiskers representing standard deviations.

In Figure 4, we show the latency measurements obtained using `ping`. We detected random streaks of lost packets in Scenario 4, with an average of 10% losses. Thus, the results shown for this scenario only used the echo requests that received a reply.

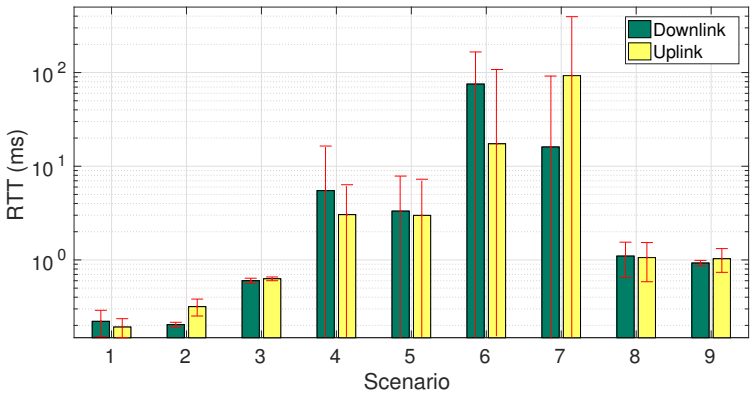

Figure 4. Measured round-trip time (RTT) in milliseconds. Bars show average results, with whiskers representing standard deviations.

We had two jitter measurements. On the one hand, in Figure 5, we show the jitter reported by `iperf3`. On the other hand, in Figures 6 and 7, we have computed the jitter using Equation (6) with the `ping` probe times.

Specifically, Figure 6 includes the average and standard deviations of punctual values, while Figure 7 represents their median and interquartile range. We note that the latter statistics were robust against outliers, and thus noticeable differences between a statistical metric and its robust counterpart may indicate the presence of such atypical values; for example, see the divergent values for Scenarios

4 and 6, as a result of random losses previously commented on. The significant differences among the jitter values reported by `iperf3` and `ping` were expected, as these applications generate different traffic patterns. While `iperf3` generates constant traffic with large back-to-back packets to fully saturate the link, `ping` transmits only periodic small packets.

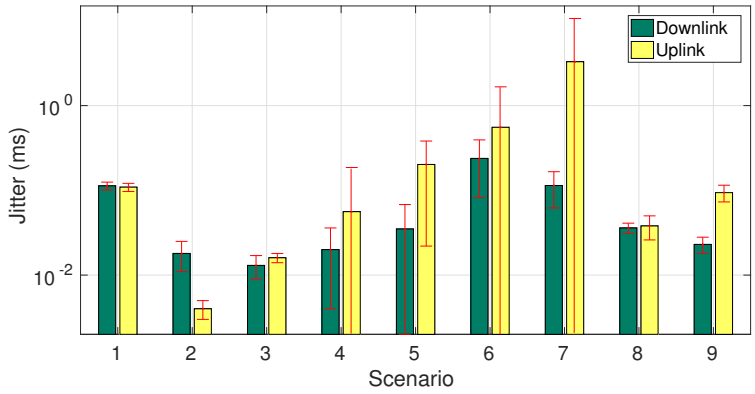

Figure 5. Measured jitter using `iperf3`, in milliseconds. Bars show average results, with whiskers representing standard deviations.

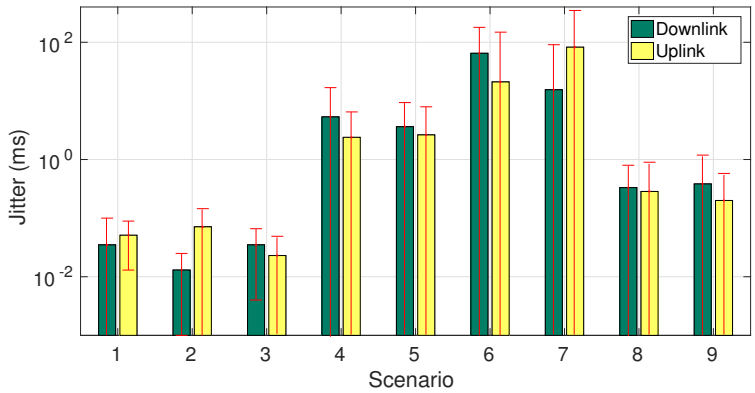

Figure 6. Measured jitter using `ping`, in milliseconds. Bars show average results, with whiskers representing standard deviations.

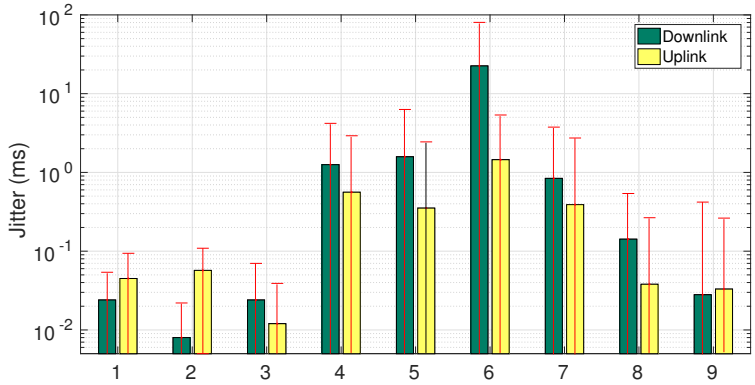

Figure 7. Measured jitter using `ping`, in milliseconds. Bars show median results, with whiskers representing interquartile range.

As a result, the jitter is expected to be lower if there is no *cross-traffic* (traffic that traverses the link that is not part of our active measurement that negatively affects it) [7], but the second situation illustrates that applications that send small blocks of data periodically (e.g., real-time multimedia

applications such as VoIP) may present higher latency variations. Finally, it is worth noticing that while `iperf3` reported higher jitter values for the uplink direction of the wireless tests, in Figure 6, we observed the opposite trend in Scenarios 4, 5 and 6.

4. Discussion and Lessons Learned

The aforementioned results expose several factors that deserve careful consideration when measuring network capabilities with low-end hardware devices. First of all, hardware limitations can appear at different layers: CPU, network interface, or internal connections among communication devices.

CPU-limited devices may not be able to transmit or receive network traffic, which fairly represents the true network capacity. Similarly, the use of these devices may also lead to overestimations of latency or packet loss. The same limitations hold if limitations appear in the network interface.

Internal connections among communication devices turn out to be a critical matter when using external appliances to extend the connectivity of devices. Paradoxically, our testbed illustrated that the performance of an external USB WiFi adapter exhibited better performance when used in the most limited hardware device.

Coherently, the following actions must be accomplished to obtain robust estimations of network KPIs, whenever possible:

- Check cores and NUMA architecture of the device, to adapt core isolation and affinity. That is, to prevent artificial performance bottlenecks, a meticulous assignation of threads has to be done.
- Calibrate network generation capabilities. To do so, first check single-stream measurements, and, after that, include multiple streams. Bottlenecks may indicate limitations of network interfaces or CPU limitations. Additionally, we note that despite having the multi-stream functionality, measurement tools must employ multiple threads to overcome CPU limitations.
- Ensure that the capabilities of the internal connection among communication devices suffice the use case, instead of assuming that higher-end equipment should provide better performance. External appliances may be limited by the design or sharing of these resources.

Additionally, band selection (i.e., 2.4 or 5 GHz) and channel occupancy exert a noticeable effect on network performance estimation. Thus, while these factors do not truly depend on the measurement system but on the specific deployment under test, they are factors that can bias the obtained results.

We summarize the relation of such factors with the measured values of KPIs in Table 2—see the official documentation for further details about `isolcpus` [25], `taskset` [26] and the NUMA configuration [27]. In this table, we link the network KPIs, the possible issues and the solutions that we applied during our experiments. We remark that, although the obtained measurements may not have been totally precise (particularly if compared with high-end hardware-assisted systems [28]), they suffice to measure the network performance in common domestic environments if a careful configuration is applied—see the results in the wired scenarios.

Table 2. Summary of Network Key Performance Indicators (KPIs), performance issues and tested solutions.

Network KPIs	Available bandwidth	Round trip time (RTT) and jitter	Packet loss
Performance Issues	• Low performance as a result of packet loss.	• High values caused either by limited resources, Media Access Control (MAC) or physical level.	• High values as a result of computational and memory-access constraints.
Tested Solutions	• `isolcpus` and `taskset` in GNU/Linux systems. • Memory affinity. • Tuning of transport-layer parameters, i.e., buffers and timeouts.	• Select adequate band to represent the true operational conditions. • `isolcpus` and `taskset` in GNU/Linux systems. • Memory affinity.	• `isolcpus` and `taskset` in GNU/Linux systems. • Mapping to Non-uniform memory access (NUMA) nodes with memory affinity.

5. Conclusions

In this work, we have conducted an empirical evaluation of critical factors for network measurement systems that use low-cost COTS platforms. We have used several standard definitions of common network KPIs and state-of-the-art tools, to characterize the bounds of such systems.

We have shown that BW estimations depend on the protocol, the computational resources of the device and how they connect to the AP. In this light, we have illustrated that low-cost ARM boards such as ODROID can achieve 100 Mbit/s throughput with TCP in a 2.4 GHz network with a cheap wireless adapter if measurement threads are carefully placed and isolated. Better results can be achieved when using PCs or routers to accomplish such measurements.

Delay measurements exhibited very different ranges depending on the network setup that we used. In the 5 GHz scenarios, we measured about 1 ms of delay, while with the 2.4 GHz WiFi, the delay had a greater variability. These challenges present a trade-off between the quality of network measurements and the capabilities of the available network measurement probes.

These findings have been used to define a guide of good practices to alleviate the shortcomings of measuring with hardware-limited equipment. As a result, our work can be useful for the improvement and development of systems to fairly characterize operational networks while keeping expenditures low.

Future lines of work include analyzing how the addition of multiple concurrent clients with this type of hardware affects the estimations of network KPIs. This is one of the underlying motivations of our work (needless to say, the capabilities of clients will exert a direct effect on the testbed configuration and operation), and our findings can help to develop these experimental scenarios. Although a systematic evaluation of such scenarios is beyond the scope of this work, we are currently working in this direction. Additionally, we are studying the effects of having mixed traffic in the measurements, for example, if some clients are uploading data while others are downloading, or the effect of more complex traffic patterns.

Acknowledgments: This work was partially funded by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund through the research projects TRAFICA (MINECO/FEDER TEC2015-69417-C2-1-R) and RACING DRONES (MINECO / FEDER RTC-2016-4744-7). The first author would like to thank the Spanish Ministry of Education, Culture and Sports for the 2016–2017 collaboration grant.

Author Contributions: All authors contributed equally to this work.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACK	TCP acknowledgment control flag
AP	Access point
BTC	Bulk transfer capacity
BW	Bandwidth
COTS	Commercial off-the-shelf
ICMP	Internet Control Message Protocol
IPPM	Internet Protocol Performance Metrics
ISP	Internet service provider
KPI	Key performance indicator
MAC	Media Access Control
MCS	Modulation and coding scheme
MIMO	Multiple-input and multiple-output
NIC	Network interface card
NUMA	Non-Uniform Access Memory
OWD	One-way delay
Q4S	Quality for Service Protocol
QoE	Quality of experience
QoS	Quality of service
RTT	Round-trip time
Rx	Reception
SYN	TCP synchronize control flag
TCP	Transmission Control Protocol
Tx	Transmission
UDP	User Datagram Protocol
VoIP	Voice over IP

References

1. Sundaresan, S.; Burnett, S.; Feamster, N.; de Donato, W. BISmark: A Testbed for Deploying Measurements and Applications in Broadband Access Networks. In Proceedings of the 2014 USENIX Annual Technical Conference (USENIX ATC 14), Philadelphia, PA, USA, 19–20 June 2014; USENIX Association: Philadelphia, PA, USA, 2014; pp. 383–394.
2. Bagnulo, M.; Burbridge, T.; Crawford, S.; Eardley, P.; Schoenwaelder, J.; Trammell, B. Building a standard measurement platform. *IEEE Commun. Mag.* **2014**, *52*, 165–173.
3. Choi, S.; Park, K.; Kim, C. Performance impact of interlayer dependence in infrastructure WLANs. *IEEE Trans. Mob. Comput.* **2006**, *5*, 829–845.
4. Paxson, V.; Almes, G.; Mahdavi, J.; Mathis, M. RFC 2330: Framework for IP Performance Metrics, 1998. Available online: https://tools.ietf.org/html/rfc2330 (accessed on 20 February 2018).
5. Hedayat, H.; Krzanowski, R.; Morton, A.; Yum, K.; Babiarz, J. RFC 5357: A Two-Way Active Measurement Protocol, 2008. Available online: https://tools.ietf.org/html/rfc5357 (accessed on 20 February 2018).
6. Atxutegi, E.; Liberal, F.; Saiz, E.; Ibarrola, E. Toward standardized internet speed measurements for end users: current technical constraints. *IEEE Commun. Mag.* **2016**, *54*, 50–57.
7. Ramos, J.; Santiago del Río, P.M.; Aracil, J.; López de Vergara, J.E. On the effect of concurrent applications in bandwidth measurement speedometers. *Comput. Netw.* **2011**, *55*, 1435–1453.
8. Pei, C.; Zhao, Y.; Chen, G.; Tang, R.; Meng, Y.; Ma, M.; Ling, K.; Pei, D. WiFi can be the weakest link of round trip network latency in the wild. In Proceedings of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2016), San Francisco, CA, USA, 10–15 April 2016; pp. 1–9.
9. Hu, Z.; Chen, Y.C.; Qiu, L.; Xue, G.; Zhu, H.; Zhang, N.; He, C.; Pan, L.; He, C. An In-depth Analysis of 3G Traffic and Performance. In Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges, London, UK, 17 August 2015; pp. 1–6.

10. Maity, M.; Raman, B.; Vutukuru, M. TCP download performance in dense WiFi scenarios. In Proceedings of the 2015 7th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, 6–10 January 2015; pp. 1–8.
11. Strauss, J.; Kaashoek, M.F. Estimating Bulk Transfer Capacity. Available online: http://web.mit.edu/jastr/Public/paper.pdf (accessed on 20 February 2018).
12. Ramos de Santiago, J. Proactive Measurement Techniques for Network Monitoring in Heterogeneous Environments. Ph.D. Thesis, Universidad Autónoma de Madrid, Madrid, Spain, 2013.
13. Almes, G.; Zekauskas, M.; Kalidindi, S.; Morton, A. RFC 7679: A One-Way Delay Metric for IP Performance Metrics (IPPM), 2016. Available online: https://tools.ietf.org/html/rfc7679 (accessed on 20 February 2018).
14. García Aranda, J.J.; Pérez Lajo, J.; Díaz Vizcaino, L.M.; Muñoz Fernández, G.; Barcenilla, C.; Cortés, M.; Salvachua, J.; Quemada, J.; Martínez Sarriegui, I.; Fajardo Ibáñez, L.; et al. The Quality for Service Protocol. Internet-draft, Internet Engineering Task Force, 2017. Work in Progress.
15. Almes, G.; Zekauskas, M.J.; Kalidindi, S. RFC 2681: A Round-Trip Delay Metric for IPPM, 1999. Available online: https://tools.ietf.org/html/rfc2681 (accessed on 20 February 2018).
16. Demichelis, C.; Chimento, P. RFC 3393: IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), 2002. Available online: https://tools.ietf.org/html/rfc3393 (accessed on 20 February 2018).
17. Almes, G.; Kalidindi, S.; Zekauskas, M. RFC 2680: A One-Way Packet Loss Metric for IPPM, 1999. Available online: https://tools.ietf.org/html/rfc2680 (accessed on 20 February 2018).
18. ODROID Products. ODROID-HC2. Available online: http://www.hardkernel.com/main/products/prdt_info.php (accessed on 19 February 2018).
19. Charfi, E.; Chaari, L.; Kamoun, L. PHY/MAC Enhancements and QoS Mechanisms for Very High Throughput WLANs: A Survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1714–1735.
20. Karmakar, R.; Chakraborty, S.; Chattopadhyay, S. Impact of IEEE 802.11 n/ac PHY/MAC High Throughput Enhancements over Transport/Application Layer Protocols-A Survey. *arXiv* **2017**, arXiv:1702.03257.
21. Moreno, V.; Ramos, J.; Santiago del Río, P.M.; García-Dorado, J.L.; Gómez-Arribas, F.J.; Aracil, J. Commodity Packet Capture Engines: Tutorial, Cookbook and Applicability. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1364–1390.
22. Mahdavi, J.; Paxson, V. RFC 2678: IPPM Metrics for Measuring Connectivity, 1999. Available online: https://tools.ietf.org/html/rfc2678 (accessed on 20 February 2018).
23. UDP Performance Stuck around 10mbits Unless Using—A Option # 234. Available online: https://github.com/esnet/iperf/issues/234 (accessed on 19 February 2018).
24. Extremely High Packet Loss with UDP Test # 296. Available online: https://github.com/esnet/iperf/issues/296 (accessed on 19 February 2018).
25. The Kernel'S Command-Line Parameters. Available online: https://www.kernel.org/doc/html/v4.14/admin-guide/kernel-parameters.html (accessed on 19 February 2018).
26. Taskset(1)-Linux Man Page. Available online: https://linux.die.net/man/1/taskset (accessed on 19 February 2018).
27. Numactl(8)-Linux Man Page. Available online: https://linux.die.net/man/8/numactl (accessed on 19 February 2018).
28. Ruiz, M.; Ramos, J.; Sutter, G.; de Vergara, J.E.L.; Lopez-Buedo, S.; Aracil, J. Accurate and affordable packet-train testing systems for multi-gigabit-per-second networks. *IEEE Commun. Mag.* **2016**, *54*, 80–87.