WiFi Performance Lab

Niccolò Bedini, Fabrizio Perna, Francesco Rosucci

1 INTRODUCTION

This report evaluates WiFi performance using various tools across different scenarios with two hosts connected to an access point. Measured data will assess the accuracy of expected goodput for each use case. We estimate expected goodput considering the test configuration and relevant literature. For each scenario, TCP and UDP protocols will be used to compare their performance in handling network traffic. This analysis will highlight the strengths and weaknesses of each protocol under various conditions, providing a comprehensive understanding of WiFi behavior.

2 TOOLS AND THEORY

2.1 Expected goodput

To estimate the expected goodput between hosts connected to the same access point, we consider various scenarios: Wi-Fi to Wi-Fi, Ethernet to Wi-Fi, and Ethernet to Ethernet connections. In our tests, since the capacity of Wi-Fi is always lower than that of Ethernet (considering that Wi-Fi operates in half-duplex mode), Wi-Fi is consistently identified as the bottleneck when both technologies are present. The calculations for expected goodput differ based on whether the protocol used is TCP or UDP, each with its own characteristics and implications:

- **TCP** connections over Wi-Fi, the expected goodput can be estimated by applying a coefficient to the bottleneck bandwidth. Due to TCP's inherent overhead, such as acknowledgment packets and retransmissions, we use a coefficient of 0.5. This means that the goodput is typically half of the bottleneck bandwidth. TCP's overhead for reliable data transfer, including congestion control and error correction, impacts throughput, especially in half-duplex scenarios.
- **UDP**, on the other hand, has a slightly higher expected goodput compared to TCP, due to its lack of acknowledgment packets and retransmissions. For UDP over Wi-Fi, we use a coefficient of 0.55. However, the simplicity of UDP comes with its own set of challenges. UDP segment size is critical for throughput. Unlike TCP, which optimizes segment size using MSS (Maximum Segment Size), UDP sends packets regardless of size. This can significantly reduce the goodput because small payloads become dwarfed by header overhead. Additionally, high volumes of data divided into tiny segments increase packet loss, especially in wireless environments prone to interference and collisions. Conversely, overly large segments can be fragmented at the IP level. Excessive fragmentation increases packet loss because losing a single fragment discards the entire original IP packet, even if other fragments arrived successfully. Consequently, while UDP has the potential for higher goodput than TCP, this advantage hinges on optimal segment size and favorable network conditions.

Wireless Security Report, Torino, 2024
2024. ACM ISBN .
https://doi.org/

In summary, when estimating the expected goodput for network tests involving Wi-Fi connections, for TCP, the goodput, in optimal conditions, is approximately 50% of the Wi-Fi link's bottleneck bandwidth while for UDP, the goodput is around 55% of the Wi-Fi link's bottleneck bandwidth, but this is highly sensitive to the segment size used. These estimates help in understanding the performance trade-offs between TCP and UDP in wireless network environments and guide the configuration for optimal data transmission.

2.2 Tools

For our analysis, we utilized two tools:

- **Iperf3** is a widely-used network testing tool designed to measure the maximum achievable bandwidth with both TCP and UDP. It allows users to assess the throughput, latency, and packet loss of a network connection. With its client-server architecture, iperf3 facilitates conducting performance tests between two endpoints, enabling network administrators and engineers to optimize network configurations and troubleshoot potential bottlenecks. Crucially for our purposes, iperf3 offers flexibility with features like variable packet sizes and cross-platform compatibility, along with support for multiple protocols.
- **Wireshark** is a powerful network protocol analyzer widely used for network troubleshooting, and analysis. It allows users to capture and interactively browse the traffic running on a computer network in real-time. With its extensive protocol support, Wireshark can decode and display data from hundreds of protocols , ranging from the commonly used TCP/IP suite to more specialized protocols such as iperf3. This tool provides detailed insights into network communications, including packet headers, payloads, and timing information, enabling network administrators, security professionals, and developers to diagnose network issues, detect anomalies, and investigate security incidents.

3 LAB SETUP AND SCENARIOS

The analysis were conducted across various scenarios where two devices are connected to an access point. These scenarios differ based on the type of connection between the hosts and the access point, as illustrated in the figures in the following subsections[**3.1**][**3.2**][**3.3**]. For each scenario four tests were conducted. The four tests are divided into two using TCP and two using UDP. For each protocol, one test involves the client sending data to the server, while the other test involves the client receiving data from the server. Each test consists of running 10 trials using iperf3 and analyzing statistical information such as minimum, maximum, standard deviation, and average.

To perform the network tests, we utilized a Bash script shown in section [**A**] to automate the entire process. The script runs 10 tests consecutively, allowing us to pass a parameter to choose the direction of the data, from sender to receiver or vice versa. Additionally, the script extracts statistical results such as minimum, maximum,

average, and standard deviation, which were used to populate the tables in figures **4** and **5**. These statistics provide a comprehensive overview of the different scenarios, aiding us in drawing more informed conclusions.

When conducting tests with UDP, it's important to make several considerations. Firstly, in the testing script, the *-b* (bandwidth) parameter is crucial. If this parameter is omitted, the script generates packets that are too small since the default value for the bandwidth is 1 Mb/s, which is far inferior to the actual capacity of the link, resulting in significantly reduced throughput. Additionally, it's important to consider that UDP does not automatically use the maximum Transmission Unit (MTU) size for segment generation. This can lead to inefficiencies discussed in the previous section if not properly configured. In iperf3, the *-l* (length) parameter can be used to specify the packet size. If this parameter is not provided, iperf3 automatically detects and uses the maximum transmission unit (MTU) size. In our case, we experimented with various packet sizes using the *-l* parameter, but we didn't see significant improvements in performance compared to the optimal value detected by iperf3. Therefore, we decided to let iperf3 automatically detect the optimal packet size for us.

3.1 WiFi-WiFi

Figure 1

In this configuration, both hosts use Arch Linux as their operating system and are connected to the access point via Wi-Fi. The two devices have different capabilities:

- **Host A:** rx: 1813.8 Mb/s, tx: 1729.6 Mb/s
- **Host B:** rx: 390 Mb/s, tx: 395 Mb/s

3.2 WiFi-Ethernet

Figure 2

In this configuration, both hosts use Arch Linux as their operating system and are connected to the access point one via WiFi and the other one via Ethernet. The two devices have different capabilities:

- **Host A:** rx: 1.813 Gb/s, tx: 1.729 Gb/s
- **Host B:** rx: 1 Gb/s, tx: 1 Gb/s

3.3 Ethernet-Ethernet

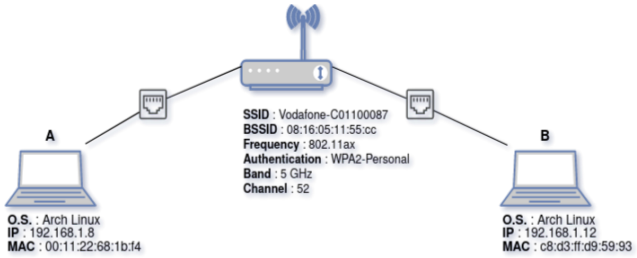


Figure 3

In this configuration, both hosts use Arch Linux as their operating system and are connected to the access point via Ethernet.

- **Host A:** rx: 1 Gb/s, tx: 1 Gb/s
- **Host B:** rx 1 Gb/s, tx: 1 Gb/s

4 RESULTS

	Test	TCP: Goodput per flow					Comment
		Prediction	Average	Min	Max	Std	
Both Ethernet	A	~949 Mb/s	887.9 Mb/s	886 Mb/s	889 Mb/s	1.044	The TCP tests with both hosts connected via Ethernet showed the best results in terms of stability and throughput. The standard deviation was minimal, indicating consistent performance, and the throughput rates were the highest observed
	B	~ 949Mb/s	941 Mb/s	941 Mb/s	941 Mb/s	0	
Both WiFi	D	~195 Mb/s	182 Mb/s	141 Mb/s	208 Mb/s	23.2809	In the WiFi-WiFi configuration, the throughput was significantly lower due to the bottleneck created by the WiFi host's limitations, which were below the channel's capacity. The standard deviation was within normal ranges for WiFi connections, reflecting typical variability due to interference and other WiFi-specific issues.
	E	~185 Mb/s	172.11 Mb/s	131 Mb/s	193 Mb/s	18.705	
Mixed mode	G	~865 Mb/s	892.3 Mb/s	872 Mb/s	903 Mb/s	8.78692	This configuration achieved bit rates comparable to the both Ethernet setup and exhibited instability similar to the both WiFi configuration. The improved bit rate is due to the WiFi-connected host in this setup having a greater throughput capacity than the host used in the WiFi-WiFi tests. However, the instability is still attributed to the wireless portion of the communication
	H	~905 Mb/s	828.3 Mb/s	775 Mb/s	880 Mb/s	28.7508	

Figure 4

	Test	UDP: Goodput per flow					Comment
		Prediction	Average	Min	Max	Std	
Both Ethernet	A	~957 Mb/s	914.6 Mb/s	910 Mb/s	917 Mb/s	2.2	The results for UDP with both hosts connected via Ethernet were the best among all combinations. Notably, only in this configuration was the performance improvement of UDP over TCP evident. This is attributed to the reliability and stability of the Ethernet channel, which allows UDP to fully utilize its lower overhead and achieve higher throughput.
	B	~957 Mb/s	956 Mb/s	956 Mb/s	956 Mb/s	0	
Both WiFi	D	~214 Mb/s	108.963 Mb/s	97.7 Mb/s	117 Mb/s	5.83822	This is one of the two scenarios (along with Mixed Mode) where our predictions were less accurate. However, this revelation is not surprising and is natural given the characteristics of WiFi communications. The throughput was lower and more variable due to the inherent instability and interference typical of WiFi networks.
	E	~214 Mb/s	178.8 Mb/s	142 Mb/s	196 Mb/s	15.2302	
Mixed mode	G	~950 Mb/s	668.6 Mb/s	611 Mb/s	710 Mb/s	31.0941	In addition to the considerations made for the TCP Mixed Mode case, we observed that even in short periods, the UDP throughput could vary by hundreds of Mb/s depending on the momentary network conditions. This further highlights the impact of the wireless segment on overall performance stability and throughput.
	H	~995 Mb/s	915.3 Mb/s	894 Mb/s	933 Mb/s	10.64	

Figure 5

Comparison beetween different setups During our network tests to evaluate goodput between different hosts connected via an access point, we noticed significant variability in speed when comparing WiFi-WiFi[**1**] and Ethernet-WiFi[**2**] configurations. Although Ethernet technology generally provides superior goodput, this variability is primarily due to the use of different computers, as the devices used in the first scenario lacked Ethernet ports. With the mixed mode configuration[**2**], the previous WiFi bottleneck is removed, which causes a significant increase in goodput. When we used Ethernet to connect both devices[**3**], they could transmit and receive data at the maximum rate allowed by their network

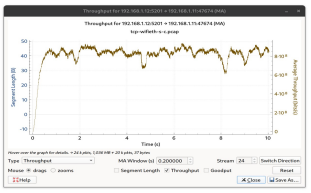
cards and Ethernet cables. This raised the goodput of the bottleneck compared to the previous scenarios, highlighting the superior efficiency of Ethernet technology. Moreover, as we will see in the following section[**4.1**], the stability and quality of the connection are significantly improved as well.

Differences beetween TCP and UDP In the Ethernet-Ethernet scenario[**3**], our tests clearly showed that UDP provides greater goodput compared to TCP. This result underscores that the previously unobserved superiority of UDP was due to the variability of the WiFi network. With the more stable Ethernet technology, the performance difference between UDP and TCP becomes evident. We initially predicted that UDP would achieve higher throughput than TCP due to its lower overhead and lack of concern for network conditions or receiver capacity. However, our observations did not consistently support this prediction in WiFi environments. This is primarily due to the fact that wireless networks are too influenced by external factors such as collisions, network congestion, interference, etc., to allow for a difference of around 5% in efficiency to be highlighted. Finally, inverting the roles of sender and receiver in our tests, we did not find any significant difference, as the bottleneck remained the same.

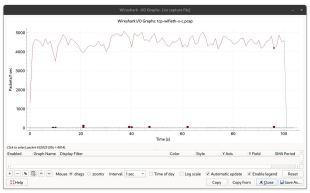
Tables' results summary the predictions of the expected goodput did not always match the actual values observed, especially where a wireless connection was involved, due to the characteristics of wireless communications. In these scenarios, we also observed that during the tests, the goodput varied significantly over time, with differences between the minimum and maximum registered goodput reaching up to hundreds of megabits and the standard deviation reaching up to 30. In the Ethernet-Ethernet scenario, the predictions were more accurate, with a much lower standard deviation, highlighting the inherent stability of this technology compared to WiFi. A more in-depth comparison of the stability is presented in the following section.

4.1 Stability Comparison

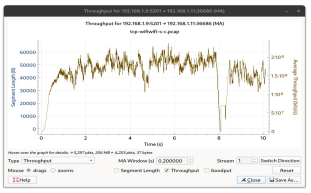
Based on the numerical results obtained from the various tests, we decided to delve deeper into the most interesting cases using different graphs provided by Wireshark. We specifically focused on the stability of the different scenarios. We chose to use three different types of graphs to provide three distinct perspectives, emphasizing the same topic from various angles while arriving at the same conclusion. The first set of graphs illustrates the variance in goodput for a single flow. From these graphs, we can observe significant instability in the Wi-Fi to Wi-Fi scenario[**6c**][**6d**], which decreases in the subsequent cases[**6a**][**6b**], particularly in the Ethernet to Ethernet scenario[**6e**][**6f**].
To compare the entirety of a test rather than a single flow, we analyzed the packets per second graph. This graph also highlights how various TCP errors present in the Wi-Fi to Wi-Fi case disappear in the Ethernet to Ethernet case, reinforcing our earlier results. Finally, to further visualize the differences in stability across the various scenarios, we included graphs of sequence numbers and retransmissions. These provide a clear and visual confirmation of our analysis, showcasing the differences in stability between the scenarios.



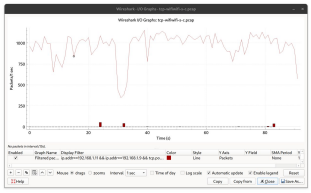
(a) WiFi-Ethernet



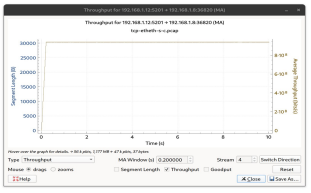
(b) WiFi-Ethernet



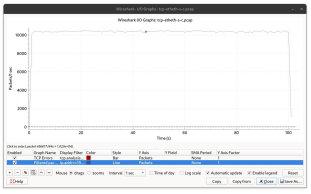
(c) WiFi-WiFi



(d) WiFi-WiFi

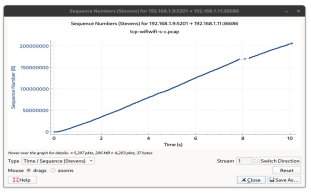


(e) Ethernet-Ethernet

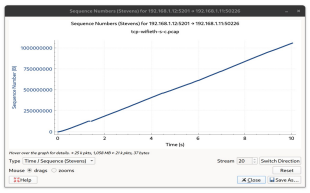


(f) Ethernet-Ethernet

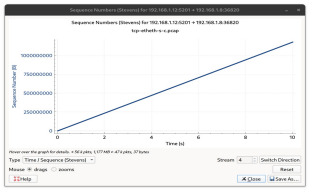
Comparing the sequence numbers across the three scenarios (both hosts on WiFi[**7a**], both hosts on Ethernet[**7c**], and mixed mode[**7b**]), we observe that the presence of WiFi renders the transmission less stable and more susceptible to retransmissions. Specifically, in the scenario where both hosts are on WiFi, there are numerous retransmissions. However, even introducing just one host on Ethernet significantly enhances the quality despite sending a higher volume of packets, thus potentially increasing the chances of having TCP errors. In the ideal scenario where both hosts are connected via Ethernet, retransmissions are entirely eliminated, resulting in the optimal transmission conditions.



(a) WiFi-WiFi



(b) WiFi-Ethernet



(c) Ethernet-Ethernet

5 CONCLUSION

Our evaluation of WiFi performance across various scenarios, using tools such as iperf3 and Wireshark, provided significant insights into the behavior and efficiency of TCP and UDP protocols under different network configurations. Several key findings and considerations emerged from our extensive testing.

Firstly, the expected goodput for TCP and UDP was estimated at 50% and 55% of the WiFi link's bottleneck bandwidth, respectively, based on their inherent characteristics. However, real-world performance showed significant variability, particularly in wireless environments. Factors such as collisions, network congestion, and interference greatly influenced the goodput, demonstrating the limitations of theoretical predictions in practical scenarios.

Secondly, Ethernet connections demonstrated far greater stability and a lower standard deviation in goodput compared to WiFi connections. This inherent stability of Ethernet was clearly evident through various statistical measures and graphical analyses. The stability and efficiency of Ethernet technology allowed for clearer differentiation between TCP and UDP protocols, confirming the expected superiority of UDP in a stable network environment.

The detailed graphical analysis provided by Wireshark reinforced these findings. Goodput variance graphs highlighted significant instability in WiFi-WiFi scenarios, which decreased notably in Ethernet-Ethernet scenarios. Packets per second graphs underscored the disappearance of TCP errors in Ethernet-Ethernet tests, reinforcing the superior stability of wired connections. Graphs of sequence numbers and retransmissions vividly illustrated the differences in stability, with numerous retransmissions occurring in WiFi scenarios and their near-elimination in Ethernet scenarios.

These findings underscore the superiority of Ethernet technology for providing stable and reliable network performance. Wireless networks, on the other hand, are inherently variable and influenced by numerous external factors, which can significantly impact both TCP and UDP performance. This variability must be accounted for when planning and optimizing wireless network deployments.

The study also highlights the importance of proper configuration for optimal performance, particularly with UDP. Parameters such as segment size need to be carefully configured to avoid inefficiencies and reduced goodput.

In conclusion, while theoretical predictions of goodput provide a useful benchmark, actual performance is highly dependent on network conditions, especially in wireless environments. Understanding the characteristics and limitations of different network technologies and protocols is essential for network administrators and engineers to optimize configurations, troubleshoot issues, and ensure reliable and efficient network performance. This comprehensive evaluation offers valuable insights into the trade-offs between different protocols and connection types, guiding better decision-making in network design and implementation.

A SCRIPT

TCP test

```

#!/bin/bash

if [[ $# -lt 2 ]]; then
  echo "Usage: ./script.sh arg serverIp where arg =1 if you want -R option otherwise arg can be passed as 0"
  exit 0
  fi

[[ -f "/tmp/gput.dat" ]] && rm /tmp/gput.dat
[[ -f "/tmp/gputtemp.dat" ]] && rm /tmp/gputtemp.dat

reverse=$1
server=$2
for i in {1..10}
do
  echo "Test $i"
  if [[ "$reverse" -ne 1 ]]; then
    echo "Client sends to server"
    iperf3 -c "$server" -i 1 | grep "receiver" | awk '{print $7}' >> /tmp/gput.dat
  else
    echo "Client receives from server"
    iperf3 -c "$server" -i 1 -R | grep "receiver" | awk '{print $7}' >> /tmp/gput.dat
  fi
done
cat /tmp/gput.dat | awk '
BEGIN{max=-1;min=9999999}
{x+=$0;y+=$0^2;if ($0<min) {min=$0}; if ($0>max) {max=$0}}
END{print "avg=", x/NR, "\nmin=", min, "\nmax=", max, "\nstd=", sqrt(y/NR-(x/NR)^2),"\n"}' >> tcpResults.txt
```

UDP test

```

#!/bin/bash

if [[ $# -lt 2 ]]; then
  echo "Usage: ./script.sh arg serverIp where arg =1 if you want -R option otherwise arg can be passed as 0"
  exit 0
  fi

[[ -f "/tmp/gput.dat" ]] && rm /tmp/gput.dat
[[ -f "/tmp/gputtemp.dat" ]] && rm /tmp/gputtemp.dat

reverse=$1
server=$2

for i in {1..10}
do
  echo "Test $i"
  if [[ "$reverse" -ne 1 ]]; then
    echo "Client sends to server"
    iperf3 -c "$server" -i 1 -u -b 1G | grep "receiver" | awk '{print $7}' >> /tmp/gput.dat
  else
    echo "Client receives from server"
    iperf3 -c "$server" -i 1 -u -b 1G -R | grep "receiver" | awk '{print $7}' >> /tmp/gput.dat
  fi
done
cat /tmp/gput.dat | awk '
BEGIN{max=-1;min=9999999}
{x+=$0;y+=$0^2;if ($0<min) {min=$0}; if ($0>max) {max=$0}}
END{print "avg=", x/NR, "\nmin=", min, "\nmax=", max, "\nstd=", sqrt(y/NR-(x/NR)^2),"\n"}' >> udpResults.txt
```