Performance Evaluation in Ethernet and WiFi Scenarios

Andrea Botticella*
andrea.botticella@studenti.polito.it

Elia Innocenti*
elia.innocenti@studenti.polito.it

Renato Mignone*
renato.mignone@studenti.polito.it

Simone Romano*
simone.romano2@studenti.polito.it

ABSTRACT

This report examines the performance of wireless and device-to-device communication by comparing theoretical predictions with experimental results in three scenarios: WiFi-only, Ethernet-only, and a mixed configuration. Using iperf3 and Wireshark, we measured goodput and analyzed its variability under different conditions. The experimental data were contrasted with theoretical estimates based on protocol efficiencies and network overheads. Our findings underscore Ethernet's stability and highlight the challenges of WiFi's shared medium and half-duplex constraints.

1 BACKGROUND AND OBJECTIVES

This laboratory evaluates and compares the performance of wired and wireless communication in a local area network by setting up three scenarios: both devices on WiFi, both on Ethernet, and a mixed configuration with one on each.

The main objectives of the lab are to:

- Measure goodput using iperf3.
- Analyze the variability and stability of the connection in each scenario by collecting data over multiple test runs.
- Compare the experimental results against theoretical predictions based on protocol efficiencies and network overhead.
- Investigate potential sources of performance degradation in wireless communication, such as interference, half-duplex operation, and shared medium limitations.

These experiments provide practical insights into the strengths and limitations of both Ethernet and WiFi, crucial for optimizing mixed network performance.

2 METHODOLOGY AND CONCEPTS

This section outlines the experimental setup, the tools employed for the measurements, and the theoretical basis for estimating goodput.

2.1 Selected Tools

To evaluate the performance of both Ethernet and WiFi connections, we utilized several specialized tools:

- **iperf3**: Used to generate traffic and measure goodput in both TCP and UDP modes. By executing repeated tests, iperf3 provides key metrics such as minimum, maximum, average, and standard deviation of the throughput.
- **Wireshark**: Employed to capture and analyze network traffic, Wireshark enabled us to inspect data flows, identify control and data frames, and validate experimental results.
- **Automation Script**: A Python script was developed to automate the entire measurement process. This script manages both server and client modes of iperf3, logs output in JSON, CSV, and plain text formats, and computes summary statistics. The script accepts several command-line flags, as detailed in the Appendix.

2.2 Goodput Estimation

Goodput represents the rate at which useful data is delivered to the application layer, excluding protocol overheads and retransmitted packets. The theoretical estimation of goodput is based on the efficiency of the protocol and the capacity of the network link:

$$G \leq \eta_{\text{protocol}} \times C,$$

where C is the capacity of the bottleneck link and η_{protocol} is the protocol efficiency.

1. For **Ethernet**, the efficiency for TCP is computed as:

   $$\eta_{TCP}^{Eth} = \frac{MSS}{MSS + \text{TCP headers} + \text{IP headers} + \text{Eth. overhead}},$$

   with the Maximum Segment Size (MSS) defined as the MTU minus the headers. For a standard MTU of 1500 bytes, we obtain:
   - MSS ≈ 1460 bytes (after subtracting 20 bytes for the IP header and 20 bytes for the TCP header),
   - An additional Ethernet overhead of approximately 38 bytes.

   Thus, the efficiency for TCP over Ethernet is approximately:

   $$\eta_{TCP}^{Eth} \approx \frac{1460}{1460 + 20 + 20 + 38} \approx 94.9\%.$$

   Similarly, the efficiency for UDP is computed as follows. Since UDP has an 8-byte header, its MSS is given by:
   - MSS ≈ 1472 bytes (after subtracting 20 bytes for the IP header and 8 bytes for the UDP header).

   Thus, the efficiency for UDP over Ethernet is given by:

   $$\eta_{UDP}^{Eth} \approx \frac{1472}{1472 + 20 + 8 + 38} \approx 95.7\%.$$

2. For **WiFi**, additional factors must be considered due to its half-duplex nature and the inherent overhead of the 802.11 protocol (e.g., control frames, retransmissions, and channel contention). Consequently, the effective efficiency is reduced by a WiFi-specific factor (η_{WiFi}). The adjusted efficiency for TCP over WiFi can be expressed as:

   $$\eta_{TCP}^{WiFi} = \eta_{TCP}^{Eth} \times \eta_{WiFi},$$

   with η_{WiFi} typically around 80% in optimal conditions. A similar adjustment applies for UDP.

These theoretical estimates set an upper bound on the achievable goodput, against which our experimental results are compared. Discrepancies between theory and practice are primarily due to

*The authors collaborated closely in developing this project.
All the authors are affiliated with Politecnico di Torino, Turin, Italy.

dynamic environmental factors, such as interference, channel variability, and the inherent limitations of wireless communication.

3 EXPERIMENTAL SETUP AND TEST CASES

3.1 Equipment and Configuration

In this section, we describe the hardware and software configuration used to perform our network performance measurements. Table 1 summarizes the main devices, their interfaces, and relevant specifications.

Device	Key Specifications
PC1	**Victus 16-s1005nl Notebook** *Operating System:* Ubuntu 24.04.2 LTS *Ethernet Interface:* Realtek RTL8111/8168/8211/8411 *Wireless Interface:* Realtek RTL8852BE (802.11ax) | 2x2
PC2	**Microsoft Surface Laptop Go 3** *Operating System:* Ubuntu 24.10 *Ethernet Interface:* via Anker PowerExpand+ USB-C Hub *Wireless Interface:* Intel Alder Lake-P CNVi (802.11ax) | 2x2
Router	**Vodafone Power Station Wi-Fi 6** *Ethernet Ports:* 4 × 1 GbE ports *Wi-Fi:* Dual-band 802.11ax (2.4 GHz | 2x2, 5 GHz | 4x4)
Cables	CAT.5E (up to 1 Gbps)

Table 1: Summary of Hardware and Network Configuration

Connections	Key Specifications	
Ethernet	*Cabling:* CAT.5E *Nominal Speed:* 1 Gbps	 *Protocol:* 802.3??
Wi-Fi	*Standard:* 802.11ax *Nominal Speed:* 1200 Mbps *Bandwidth:* 80 MHz	*Security Protocol:* ? *Frequency:* 5 GHz *Channel:* ?

Table 2: Ethernet and Wi-Fi Connection Specifications

This hardware setup allows us to compare Ethernet versus Wi-Fi performance under a consistent router and cabling environment. In the next section, we detail the evaluation scenarios and the measurement methodology.

3.2 Evaluation Scenarios

We considered three distinct network configurations to assess the performance differences between wired and wireless communications. For each scenario, the theoretical goodput is computed based on the nominal link capacity and protocol efficiency.

1. **Both Ethernet:**
   In this scenario, both PC1 and PC2 are connected to the router via CAT.5E cables, providing a nominal link capacity of 1 Gbps. The efficiency for Ethernet is calculated as follows:
   $$\eta_{TCP}^{Eth} \approx \frac{1460}{1460+20+20+38} \approx 94.9\%,$$
   $$\eta_{UDP}^{Eth} \approx \frac{1472}{1472+20+8+38} \approx 95.7\%.$$
   Thus, the expected goodput is:
   $$G_{TCP}^{Eth} \leq 0.949 \times 1000\,\text{Mbps} \approx 949\,\text{Mbps},$$
   $$G_{UDP}^{Eth} \leq 0.957 \times 1000\,\text{Mbps} \approx 957\,\text{Mbps}.$$
2. **Both Wi-Fi:**
   For this configuration, both devices use their wireless interfaces (802.11ax) to connect to the router. Although the nominal Wi-Fi link speed is assumed to be approximately 867 Mbps, the half-duplex nature of Wi-Fi effectively halves the throughput available for data transfer. Assuming a Wi-Fi efficiency factor of about 80%, the expected goodput for TCP is:
   $$G_{TCP}^{WiFi} \leq 0.80 \times 867\,\text{Mbps} \times \frac{1}{2} \approx 347\,\text{Mbps},$$
   and similarly for UDP:
   $$G_{UDP}^{WiFi} \leq 0.806 \times 867\,\text{Mbps} \times \frac{1}{2} \approx 350\,\text{Mbps}.$$
3. **Mixed Scenario:**
   In this configuration, one device (PC1) is connected via Ethernet while the other (PC2) uses its Wi-Fi interface. Here, the bottleneck is the Wi-Fi link; however, since only one device is utilizing Wi-Fi, the throughput is not halved. The expected goodput is then:
   $$G_{TCP}^{Mixed} \leq 0.80 \times 867\,\text{Mbps} \approx 694\,\text{Mbps},$$
   $$G_{UDP}^{Mixed} \leq 0.806 \times 867\,\text{Mbps} \approx 698\,\text{Mbps}.$$

These calculations provide the theoretical upper bounds for goodput in each scenario. The experimental results, obtained via automated measurements using the provided Python script, are compared against these predictions to evaluate real-world performance.

4 ANALYSIS AND FINDINGS

4.1 TCP Performance

The performance tests using TCP reveal several noteworthy trends. The analysis for TCP performance in different scenarios is organized as follows:

Test	TCP: Goodput per flow (Mbps)				
	Prediction	Average	Min	Max	Std
Both WiFi	?	434.7	396.3	461.96	22.5
Both Ethernet	?	939.6	938.2	942.7	1.5
Mixed	?	663.7	619.2	698.86	26.6
Shared Capacity	?	536.8	440.5	722.2	112

Table 3: TCP Results (Client → Server)

1. **Both Ethernet:**
   Figure 1 shows the TCP throughput measured in the Ethernet scenario. The graph reveals a rapid ramp-up in throughput during the first few seconds, followed by a stable transmission rate that approaches the theoretical value.
   Figure 2 illustrates the round-trip time (RTT), which remains very low (typically within a few milliseconds), highlighting the minimal latency in wired connections.

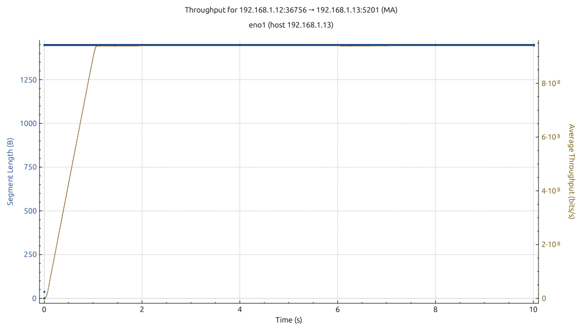

Figure 1: TCP Throughput in the Ethernet Scenario.

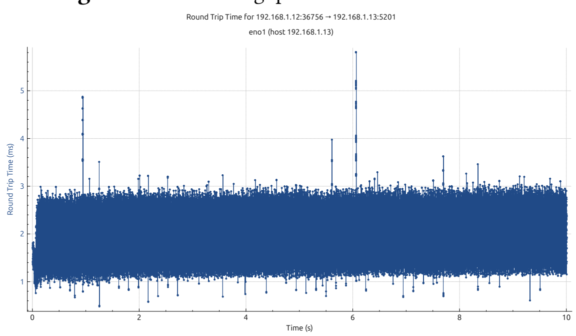

Figure 2: TCP Round Trip Time in the Ethernet Scenario.

Furthermore, the I-O graph (Fig. ??) confirms a consistent packet flow with little variation, indicating that the Ethernet setup effectively utilizes the available capacity.
Overall, the Ethernet scenario demonstrates a near-ideal performance with high throughput and minimal latency, closely matching the theoretical predictions.

2. **Both WiFi:**
In the WiFi scenario, the throughput graph (Fig. 3) shows an initial ramp-up phase during the first 2 seconds, after which the throughput fluctuates around an average value that is significantly lower than the theoretical maximum of approximately 347 Mbps. These fluctuations suggest that protocol overhead, wireless interference, and the half-duplex nature of WiFi adversely affect performance.

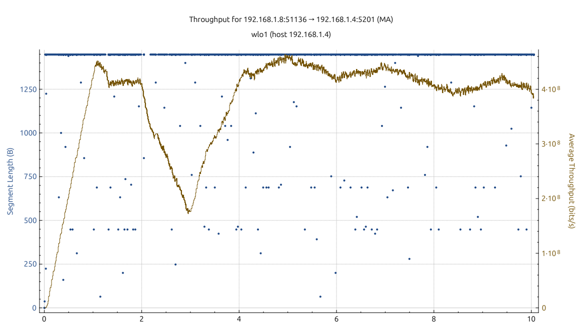

Figure 3: TCP Throughput in the WiFi Scenario.

The round-trip time (RTT) measurements (Fig. 4) reveal RTT values ranging from about 50 to 200 ms, indicating intermittent delays likely due to congestion and contention in the wireless medium.

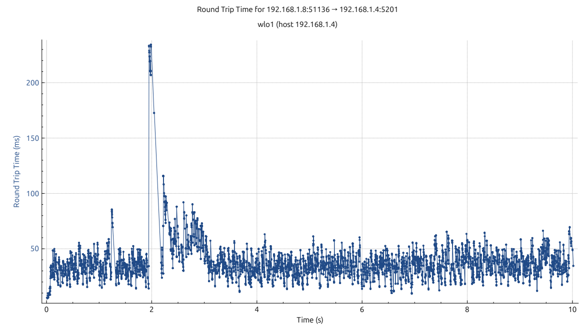

Figure 4: TCP Round Trip Time in the WiFi Scenario.

Furthermore, the I-O graph (Fig. ??) illustrates a variable number of transmitted packets per interval, reflecting the dynamic nature of WiFi communication where channel conditions and collision avoidance mechanisms influence performance.
Overall, while the theoretical capacity for TCP over WiFi is estimated to be around 347 Mbps, the experimental data indicate that real-world factors substantially reduce the effective throughput.

3. **Mixed:**
Figure 5 displays the TCP throughput for the mixed configuration. The graph shows that the throughput reaches a stable level after an initial ramp-up phase, although it remains below the Ethernet scenario and is consistent with the expected reduction due to the reliance on the wireless link.

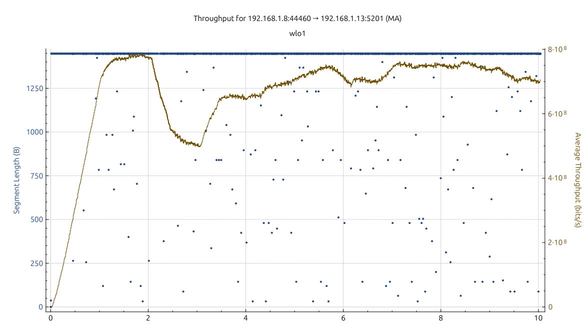

Figure 5: TCP Throughput in the Mixed Ethernet/WiFi Scenario.

The round-trip time (RTT) measurements, presented in Figure 6, indicate moderate latency, with RTT values generally remaining within a lower range compared to the pure WiFi scenario. This suggests that the wired segment helps in reducing overall latency.

The I-O graph for TCP (Fig. ??) shows a relatively steady packet flow over the test intervals, confirming that the mixed configuration maintains a stable performance despite the inherent variability of the wireless link.

3a. **Shared Capacity:**
In this scenario, a third host connected to the same access point

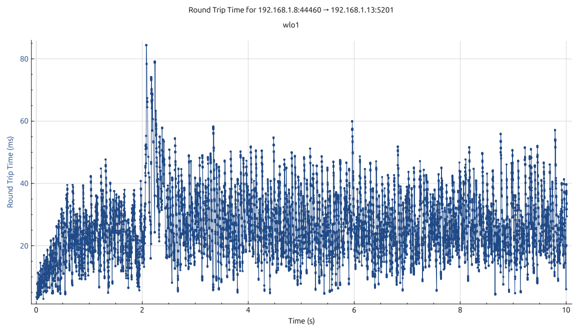

Figure 6: TCP Round Trip Time in the Mixed Ethernet/WiFi Scenario.

was concurrently downloading the film "Natale a Rio" (directed by Neri Parenti), which introduced significant interference during the tests. This additional traffic compromised the available network capacity, leading to degraded performance. Figure **??** shows the TCP throughput under this shared capacity condition. Compared to the mixed scenario without interference, the throughput exhibits a notable decrease. The average throughput is lower, reflecting the reduced available bandwidth caused by the competing download traffic.
The round-trip time measurements (Fig. **??**) indicate increased variability and slightly elevated latency. Although the RTT values remain relatively moderate, the fluctuations suggest that the network experiences occasional congestion and delays as a result of the third host's activity.
The I-O graph for TCP (Fig. 7) further confirms the impact of the interference. The graph displays irregular intervals and a lower packet transmission rate compared to the mixed scenario without the additional load, demonstrating how the extra traffic disrupts the steady flow of data.

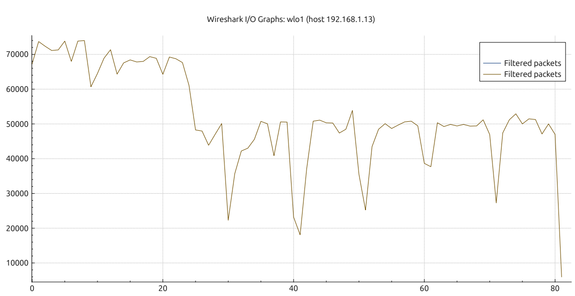

Figure 7: Wireshark I-O Graph for TCP in the Shared Capacity Scenario.

4.2 UDP Performance

The UDP tests offer an insightful comparison to the TCP results by eliminating congestion control and acknowledgment overhead. The analysis for UDP performance across different scenarios is structured as follows:

Test	**UDP: Goodput per flow (Mbps)**				
	Prediction	Average	Min	Max	Std
Both WiFi	?	487.8	453.1	499.9	15.8
Both Ethernet	?	952.8	948.3	954.6	1.73
Mixed	?	674.9	636.6	717.8	28
Shared Capacity	?	472.1	355.9	699.1	121.1

Table 4: UDP Results (Client → Server)

1. **Both Ethernet:**
   In the Ethernet configuration, the I-O graph for UDP indicates a steady packet flow, with minimal fluctuations compared to the WiFi scenario.
   The throughput achieved is very close to the theoretical prediction, confirming that the wired setup reliably supports high-speed data transfer. The absence of retransmission or congestion control overhead in UDP further contributes to this consistent performance.
   In summary, the wired (Ethernet) tests demonstrate that both TCP and UDP protocols achieve performance levels very close to their theoretical capacities, with TCP showing stable throughput and low latency, and UDP exhibiting a consistent packet flow and high throughput. This confirms that, in a controlled wired environment, network performance is minimally impacted by protocol overhead or environmental factors.
2. **Both WiFi:**
   In the WiFi scenario, the I-O graph for UDP demonstrates a more consistent packet flow compared to TCP. However, despite the smoother transmission, the overall throughput remains below the theoretical upper bound. The lack of retransmission mechanisms in UDP allows for slightly higher instantaneous throughput; nonetheless, factors like interference and channel contention continue to impact performance.
   A direct comparison between TCP and UDP in the WiFi scenario shows that UDP can achieve marginally higher throughput due to its reduced overhead. Nonetheless, both protocols suffer from real-world limitations that prevent them from reaching their theoretical capacities. This discrepancy between capacity and theoretical goodput emphasizes the impact of wireless interference, channel contention, and protocol-specific overhead on performance.
3. **Mixed:**
   In the mixed scenario, the UDP I-O graph indicates a consistent flow of packets, similar to the TCP case but with slightly less variability due to the absence of congestion control.
   The throughput observed is in line with expectations given that only the wireless link acts as the bottleneck.

   In summary, the mixed scenario demonstrates that while the presence of a wired connection on one end improves overall latency and stability compared to a full WiFi configuration, the performance remains primarily constrained by the wireless link. Both TCP and UDP protocols achieve throughput values that are consistent with theoretical predictions for a mixed Ethernet/WiFi environment.

3a. **Shared Capacity:**
   The UDP tests under the shared capacity scenario also reveal the

negative impact of the additional download traffic. Although UDP is less affected by protocol overhead, the increased contention for the wireless medium leads to degraded performance. The UDP performance in this scenario is indirectly reflected in the I-O graph. The steady flow of packets observed in a non-interfered environment is disrupted, resulting in a lower effective throughput.

Despite the inherent resilience of UDP to retransmission delays, the interference from the third host causes a noticeable reduction in performance. The graph shows that the packet flow is not as consistent, further underlining the effects of shared capacity when additional traffic is present.

Overall, the shared capacity scenario clearly demonstrates that when a third host generates significant traffic (as in the case of streaming a movie), the available network resources are further divided, leading to performance degradation for both TCP and UDP protocols. This scenario highlights the importance of considering real-world usage patterns and interference when designing and evaluating network performance.

5 CONCLUSION

In this project, we evaluated the performance of network communication under various scenarios using both wired (Ethernet) and wireless (WiFi) connections. Our experimental results, obtained through automated measurements with iperf3 and detailed packet analysis with Wireshark, were compared against theoretical predictions of goodput for both TCP and UDP protocols.

Overall, the Ethernet scenario demonstrated near-ideal performance, with throughput and latency closely matching the theoretical values. This confirms that a controlled wired environment can efficiently utilize available bandwidth with minimal interference. In contrast, the WiFi scenario showed a significant performance drop, with fluctuations in throughput and increased latency due to the inherent limitations of wireless communication such as interference, contention, and the half-duplex nature of WiFi.

The mixed scenario, where one device is connected via Ethernet and the other via WiFi, presented an intermediate case. Here, while the wired segment helped in reducing latency and stabilizing performance, the overall throughput remained limited by the wireless link. Finally, the shared capacity scenario—where an additional host engaged in heavy traffic (streaming a movie)—further degraded performance for both TCP and UDP tests. This clearly highlights the impact of network congestion and shared medium contention on real-world performance.

These findings emphasize the importance of considering environmental and traffic-related factors when designing and optimizing network infrastructures. While theoretical models provide useful upper bounds, actual network performance is influenced by a range of practical factors that must be taken into account for effective network planning and troubleshooting.

A APPENDIX

Server Mode Initialization

```python
def run_server():
    """Run iperf3 server with clean output handling.
        """
    server_logger.info("Starting iperf3 server...")

    proc = subprocess.Popen(
        ["iperf3", "-s", "-J"],
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
        text=True,
        bufsize=1,
    )
    # Handle server output and errors in a separate
        thread...
```

Listing 1: Excerpt for server mode initialization.

Client Mode Execution and Reporting

```python
def run_client(server_ip, udp=False, bitrate="1M",
     iterations=10):
    """Run iperf3 client tests and generate reports.
        """
    for i in range(iterations):
        cmd = ["iperf3", "-c", server_ip, "-J", "-t",
            "10", "-i", "1"]
        if udp:
            cmd.extend(["-u", "-b", bitrate])

        result = subprocess.run(
            cmd,
            capture_output=True,
            text=True,
            check=True
        )

        data = json.loads(result.stdout)
        # Extract test data, compute statistics, and
            log results...
```

Listing 2: Excerpt for client mode execution.

Logging and Output Management

```python
def setup_logger(log_file, name):
    logger = logging.getLogger(name)
    logger.setLevel(logging.DEBUG)
    handler = logging.FileHandler(log_file)
    formatter = logging.Formatter(
        '%(asctime)s - %(levelname)s - %(message)s'
    )
    handler.setFormatter(formatter)
    logger.addHandler(handler)
    return logger
```

Listing 3: Excerpt for logging setup.

These snippets illustrate how the script handles server mode initialization, client tests (both TCP and UDP), and logging. Error handling, multithreaded `stderr` management, and CSV report generation are also included in the complete script.