

Lab WiFi Report

Andrea Botticella*

andrea.botticella@studenti.polito.it

Renato Mignone*

renato.mignone@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, offering valuable insights for optimizing mixed network configurations.

1 BACKGROUND AND OBJECTIVES

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

5 The main objectives of the lab are to:

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

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

2 METHODOLOGY AND CONCEPTS

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

19 2.1 Selected Tools

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

- 22 • **iperf3**: Used to generate traffic and measure goodput in both
- 23 TCP and UDP modes. By executing repeated tests, iperf3 provides
- 24 key metrics such as minimum, maximum, average, and standard
- 25 deviation of the throughput.
- 26 • **Wireshark**: Employed to capture and analyze network traffic,
- 27 Wireshark enabled us to inspect data flows, identify control and
- 28 data frames, and validate experimental results.

*The authors collaborated closely in developing this project.

All the authors are affiliated with Politecnico di Torino, Turin, Italy.

Elia Innocenti*

elia.innocenti@studenti.polito.it

Simone Romano*

simone.romano2@studenti.polito.it

- **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:
• **-server**: Launches the iperf3 server in JSON output mode.
• **<SERVER_IP>**: Specifies the server's IP address when running in client mode.
• **-udp**: Switches the test from the default TCP mode to UDP.
• **-bitrate**: Sets the target bitrate for UDP tests (e.g., 10M for 10 Mbps).
• **-iter**: Determines the number of test iterations to perform. The output files (logs, JSON, CSV reports, and raw output) generated by this script are used to document the experimental results and facilitate further analysis.

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.

For Ethernet connections, the efficiency for TCP is computed as:

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

with the Maximum Segment Size (MSS) defined as the MTU minus the headers. For a standard MTU of 1500 bytes, we obtain:

- $MSS \approx 1460$ 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:

- $MSS \approx 1472$ 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:

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

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 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 <i>Operating System:</i> Ubuntu 24.04.2 LTS <i>Ethernet Interface:</i> Realtek RTL8111/8168/8211/8411 <i>Wireless Interface:</i> Realtek RTL8852BE (802.11ax) 2x2
PC2	Microsoft Surface Laptop Go 3 <i>Operating System:</i> Ubuntu 24.10 <i>Ethernet Interface:</i> via Anker PowerExpand+ USB-C Hub <i>Wireless Interface:</i> Intel Alder Lake-P CNVi (802.11ax) 2x2
Router	Vodafone Power Station Wi-Fi 6 <i>Ethernet Ports:</i> 4 × 1 GbE ports <i>Wi-Fi:</i> 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	<i>Cabling:</i> CAT.5E	<i>Nominal Speed:</i> 1 Gbps	<i>Protocol:</i> 802.3??
Wi-Fi	<i>Standard:</i> 802.11ax	<i>Security Protocol:</i> ?	<i>Frequency:</i> 5 GHz
	<i>Nominal Speed:</i> 1200 Mbps	<i>Bandwidth:</i> 80 MHz	<i>Channel:</i> ?

Table 2: Ethernet and Wi-Fi Connection Specifications

This hardware setup allows us to compare Ethernet versus WiFi 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 WiFi link speed is assumed to be approximately 867 Mbps, the half-duplex nature of WiFi effectively halves the throughput available for data transfer. Assuming a WiFi 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 WiFi interface. Here, the bottleneck is the WiFi link; however, since only one device is utilizing WiFi, 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:

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.

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)

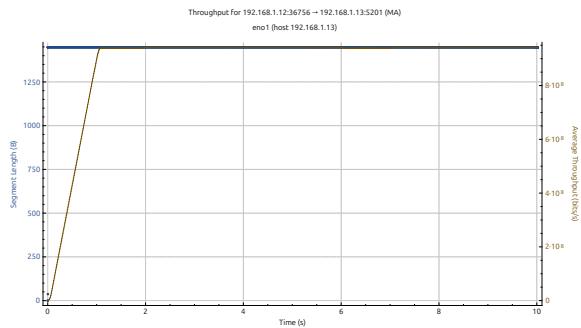


Figure 1: TCP Throughput in the Ethernet Scenario.

Figure 2 illustrates the round-trip time (RTT), which remains very low (typically within a few milliseconds), highlighting the minimal latency in wired connections.

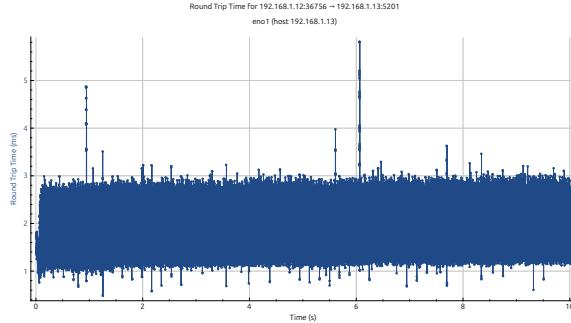


Figure 2: TCP Round Trip Time in the Ethernet Scenario.

Furthermore, the I-O graph (Fig. 3) confirms a consistent packet flow with little variation, indicating that the Ethernet setup effectively utilizes the available capacity.

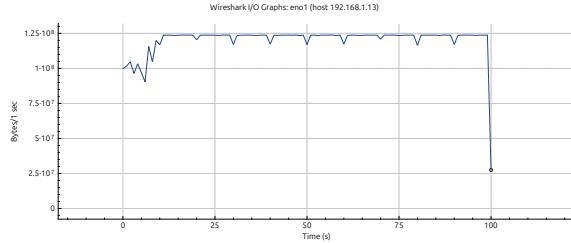


Figure 3: Wireshark I-O Graph for TCP in the Ethernet Scenario.

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. 4) 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.

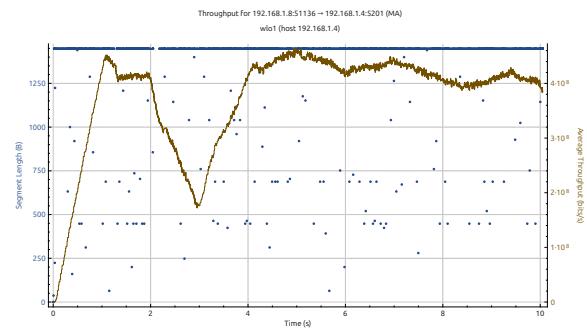


Figure 4: TCP Throughput in the WiFi Scenario.

The round-trip time (RTT) measurements (Fig. 5) reveal RTT values ranging from about 50 to 200 ms, indicating intermittent delays likely due to congestion and contention in the wireless medium.

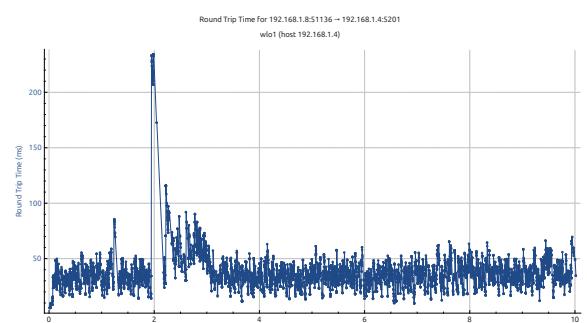


Figure 5: TCP Round Trip Time in the WiFi Scenario.

Furthermore, the I-O graph (Fig. 6) 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.

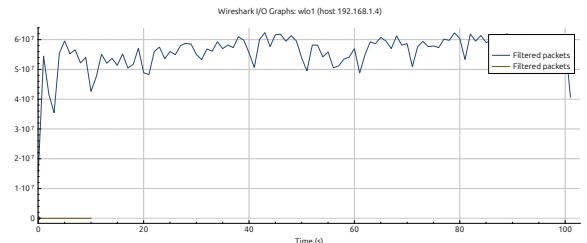


Figure 6: Wireshark I-O Graph for TCP in the WiFi Scenario.

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.

156 **3. Mixed:**
 157 Figure 7 displays the TCP throughput for the mixed configuration.
 158 The graph shows that the throughput reaches a stable level
 159 after an initial ramp-up phase, although it remains below the
 160 Ethernet scenario and is consistent with the expected reduction
 161 due to the reliance on the wireless link.
 162

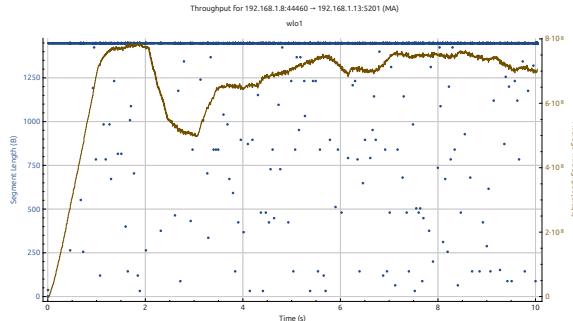


Figure 7: TCP Throughput in the Mixed Ethernet/WiFi Scenario.

163 The round-trip time (RTT) measurements, presented in Figure 8,
 164 indicate moderate latency, with RTT values generally remain-
 165 ing within a lower range compared to the pure WiFi scenario.
 166 This suggests that the wired segment helps in reducing overall
 167 latency.
 168

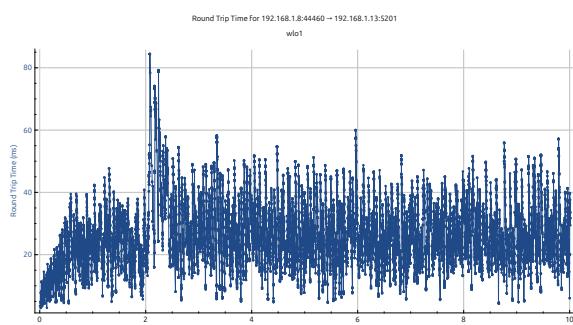


Figure 8: TCP Round Trip Time in the Mixed Ethernet/WiFi Scenario.

169 The I-O graph for TCP (Fig. 9) shows a relatively steady packet
 170 flow over the test intervals, confirming that the mixed config-
 171 uration maintains a stable performance despite the inherent
 172 variability of the wireless link.

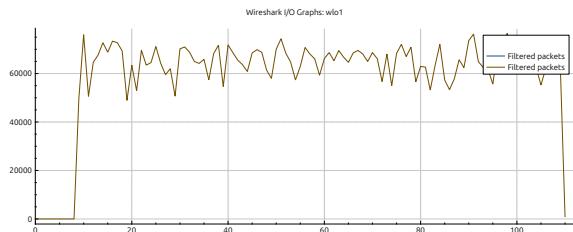


Figure 9: Wireshark I-O Graph for TCP in the Mixed Scenario.

3a. **Shared Capacity:**
 173 In this scenario, a third host connected to the same access point
 174 was concurrently downloading the film "Natale a Rio" (directed
 175 by Neri Parenti), which introduced significant interference dur-
 176 ing the tests. This additional traffic compromised the available
 177 network capacity, leading to degraded performance. Figure 10
 178 shows the TCP throughput under this shared capacity condi-
 179 tion. Compared to the mixed scenario without interference, the
 180 throughput exhibits a notable decrease. The average throughput
 181 is lower, reflecting the reduced available bandwidth caused by
 182 the competing download traffic.
 183

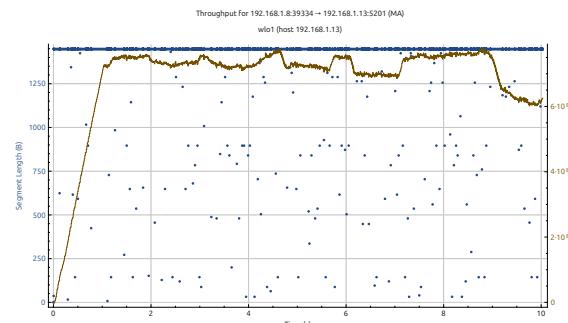


Figure 10: TCP Throughput in the Shared Capacity Scenario (with third-host interference).

The round-trip time measurements (Fig. ??) indicate increased
 184 variability and slightly elevated latency. Although the RTT
 185 values remain relatively moderate, the fluctuations suggest that
 186 the network experiences occasional congestion and delays as a
 187 result of the third host's activity.
 188 The I-O graph for TCP (Fig. 11) further confirms the impact of
 189 the interference. The graph displays irregular intervals and a
 190 lower packet transmission rate compared to the mixed scenario
 191 without the additional load, demonstrating how the extra traffic
 192 disrupts the steady flow of data.
 193

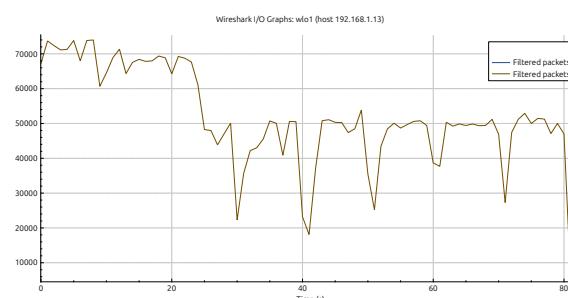


Figure 11: Wireshark I-O Graph for TCP in the Shared Capacity Scenario.

4.2 UDP Performance

The UDP tests offer an insightful comparison to the TCP results
 195 by eliminating congestion control and acknowledgment overhead.
 196 The analysis for UDP performance across different scenarios is
 197 structured as follows:
 198

1. Both Ethernet:

In the Ethernet configuration, the I-O graph for UDP indicates

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)

248 UDP is less affected by protocol overhead, the increased contention for the wireless medium leads to degraded performance.
249 The UDP performance in this scenario is indirectly reflected
250 in the I-O graph. The steady flow of packets observed in a
251 non-interfered environment is disrupted, resulting in a lower
252 effective throughput.
253

254 Despite the inherent resilience of UDP to retransmission de-
lays, the interference from the third host causes a noticeable
255 reduction in performance. The graph shows that the packet flow
256 is not as consistent, further underlining the effects of shared
257 capacity when additional traffic is present.
258

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

5 CONCLUSION

266 In this project, we evaluated the performance of network commu-
267 nication under various scenarios using both wired (Ethernet) and
268 wireless (WiFi) connections. Our experimental results, obtained
269 through automated measurements with iperf3 and detailed packet
270 analysis with Wireshark, were compared against theoretical pre-
271 dictions of goodput for both TCP and UDP protocols.

272 Overall, the Ethernet scenario demonstrated near-ideal perfor-
273 mance, with throughput and latency closely matching the theoreti-
274 cal values. This confirms that a controlled wired environment can
275 efficiently utilize available bandwidth with minimal interference. In
276 contrast, the WiFi scenario showed a significant performance drop,
277 with fluctuations in throughput and increased latency due to the in-
278 herent limitations of wireless communication such as interference,
279 contention, and the half-duplex nature of WiFi.

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

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

201 a steady packet flow, with minimal fluctuations compared to
202 the WiFi scenario.

203 The throughput achieved is very close to the theoretical pre-
204 diction, confirming that the wired setup reliably supports high-
205 speed data transfer. The absence of retransmission or congestion
206 control overhead in UDP further contributes to this consistent
207 performance.

208 In summary, the wired (Ethernet) tests demonstrate that both
209 TCP and UDP protocols achieve performance levels very close
210 to their theoretical capacities, with TCP showing stable through-
211 put and low latency, and UDP exhibiting a consistent packet flow
212 and high throughput. This confirms that, in a controlled wired
213 environment, network performance is minimally impacted by
214 protocol overhead or environmental factors.

215 2. Both WiFi:

216 In the WiFi scenario, the I-O graph for UDP demonstrates a
217 more consistent packet flow compared to TCP. However, despite
218 the smoother transmission, the overall throughput remains
219 below the theoretical upper bound. The lack of retransmission
220 mechanisms in UDP allows for slightly higher instantaneous
221 throughput; nonetheless, factors like interference and channel
222 contention continue to impact performance.

223 A direct comparison between TCP and UDP in the WiFi scenario
224 shows that UDP can achieve marginally higher throughput due
225 to its reduced overhead. Nonetheless, both protocols suffer from
226 real-world limitations that prevent them from reaching their
227 theoretical capacities. This discrepancy between capacity and
228 theoretical goodput emphasizes the impact of wireless inter-
229 ference, channel contention, and protocol-specific overhead on
230 performance.

231 3. Mixed:

232 In the mixed scenario, the UDP I-O graph indicates a consistent
233 flow of packets, similar to the TCP case but with slightly less
234 variability due to the absence of congestion control.

235 The throughput observed is in line with expectations given that
236 only the wireless link acts as the bottleneck.

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

244 3a. Shared Capacity:

245 The UDP tests under the shared capacity scenario also reveal the
246 negative impact of the additional download traffic. Although

A APPENDIX

295 Server Mode Initialization

```
296 def run_server():
297     """Run iperf3 server with clean output handling.
298     """
299     server_logger.info("Starting iperf3 server...")
300
301     proc = subprocess.Popen(
302         ["iperf3", "-s", "-J"],
303         stdout=subprocess.PIPE,
304         stderr=subprocess.PIPE,
305         text=True,
306         bufsize=1,
307     )
308     # Handle server output and errors in a separate
309     # thread...
```

Listing 1: Excerpt for server mode initialization.

310 Client Mode Execution and Reporting

```
311 def run_client(server_ip, udp=False, bitrate="1M",
312                 iterations=10):
313     """Run iperf3 client tests and generate reports.
314     """
315     for i in range(iterations):
316         cmd = ["iperf3", "-c", server_ip, "-J", "-t",
317                "10", "-i", "1"]
318         if udp:
319             cmd.extend(["-u", "-b", bitrate])
320
321         result = subprocess.run(
322             cmd,
323             capture_output=True,
324             text=True,
325             check=True
326         )
327
328         data = json.loads(result.stdout)
329         # Extract test data, compute statistics, and
330         log results...
```

Listing 2: Excerpt for client mode execution.

331 Logging and Output Management

```
332 def setup_logger(log_file, name):
333     logger = logging.getLogger(name)
334     logger.setLevel(logging.DEBUG)
335     handler = logging.FileHandler(log_file)
336     formatter = logging.Formatter(
337         '%(asctime)s - %(levelname)s - %(message)s'
338     )
339     handler.setFormatter(formatter)
340     logger.addHandler(handler)
341
342     return logger
```

Listing 3: Excerpt for logging setup.

342 These snippets illustrate how the script handles server mode
343 initialization, client tests (both TCP and UDP), and logging. Er-
344 ror handling, multithreaded stderr management, and CSV report
345 generation are also included in the complete script.