PENGEMBANGAN PROGRAM KULIAH MODEL LAPISAN FISIK DAN DATALINK JARINGAN KOMPUTER BERBASIS PUSTAKA GRAFIS SEBAGAI MEDIA PERKULIAHAN PEMODELAN DAN SIMULASI JARINGAN PADA PROGRAM STUDI INFORMATIKA

Berkah I. Santoso, ST, MTI (0309068003)

Program Studi Informatika
Fakultas Teknik dan Ilmu Komputer
Universitas Bakrie
Jakarta

2025

LEMBAR PENGESAHAN

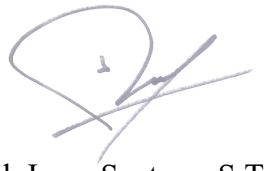
1.	Judul Program	: Pengembangan Program Kuliah Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis Sebagai Media Perkuliahan Pemodelan dan Simulasi Jaringan Kelas Reguler dan Karyawan Pada Program Studi Informatika
2.	Dosen Penyusun	
	a. Nama Lengkap	: Berkah Iman Santoso, S.T, MTI
	b. Jenis Kelamin	: Laki Laki
	c. NIDN	: 0309068003
	d. Pangkat/Golongan	: Asisten Ahli / III b
	e. Jabatan	: Dosen
	f. No. telepon seluler/Alamat Surel	: 08128114857 / berkah.santoso@bakrie.ac.id
3.	Anggota Tim Pengusul Kegiatan	
	a. Dosen	: -
	b. Praktisi	: -
4.	Peserta	
	a. Mahasiswa	: -
	b. Alumni	: -
5.	Biaya Kegiatan	
	a. Universitas Bakrie	: -
	b. Sumber lain	: -
6.	Tahun Pelaksanaan	: 2025

Jakarta, 26 November 2025

Mengetahui,
Ketua Program Studi Informatika

Iwan Adhicandra, MIEEE, MIET, MBCS.
NIP: 0018025806

Dosen Pengusul



Berkah Iman Santoso, S.T, MTI
NIDN: 0309068003

Daftar Isi

Bagian 1

Kata Pengantar

Atas berkat rahmat Allah Subhanahu Wa Ta'ala, selesailah Pengembangan Program Kuliah Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis Sebagai Media Perkuliahan Pemodelan dan Simulasi Jaringan pada Kelas Reguler dan Karyawan, Program Studi Informatika, Fakultas Teknik dan Ilmu Komputer. Oleh karena itu, penulis sebagai pengembang program kuliah model lapisan fisik dan datalink jaringan komputer berbasis pustaka grafis untuk media perkuliahan Pemodelan dan Simulasi Jaringan Kelas Reguler dan Karyawan, memberikan penghargaan yang setinggi-tingginya dan menyampaikan ucapan terima kasih yang sebesar-besarnya atas segala bantuannya terutama kepada yang terhormat :

1. Rektor Universitas Bakrie.
2. Ketua Program Studi Informatika, FTIK, Universitas Bakrie.
3. Kepala Perpustakaan Universitas Bakrie.
4. Kolega Dosen pada Program Studi Informatika, FTIK.

Semoga kegiatan ini dapat berkembang terus sejalan dengan perkembangan Program Studi Informatika Universitas Bakrie.

Jakarta, November 2025
Pelaksana Pengembangan Program Kuliah Prodi Informatika

Bagian 2

Latar Belakang

Maksud dan tujuan diselenggarakannya Pengembangan Program Kuliah Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis sebagai Media Perkuliahan Pemodelan dan Simulasi Jaringan pada Kelas Reguler dan Karyawan, Program Studi Informatika adalah dalam rangka memberikan alternatif perangkat bantu bagi mahasiswa reguler dan mahasiswa kelas karyawan yang bekerja pada jam kerja, untuk tetap mempelajari pemodelan dan simulasi jaringan komputer tanpa harus tergantung pada *software* simulator seperti OpNet®yang saat ini telah berlisensi *proprietary* dan berbayar setelah diakuisisi oleh Riverbed Technology®menjadi Riverbed Modeler®. Selain itu simulator untuk model lapisan fisik dan datalink ini memberikan terobosan penggunaan teknologi terkini untuk meningkatkan kemampuan dosen selaku salah satu mediator pembelajaran sebagai perluasan jangkauan kanal akses perkuliahan kepada mahasiswa kelas reguler dan kelas karyawan. Kegiatan Pengembangan Program Kuliah ini sejalan dengan upaya Universitas Bakrie dalam menerapkan *experience-based learning* untuk dapat terus memberikan layanan pembelajaran dalam rangka memberikan *update* teknologi terkini yang dapat dijangkau oleh mahasiswa kelas reguler dan karyawan. Penggunaan aplikasi Model Lapisan Fisik dan Datalink Jaringan Komputer berbasis Pustaka Grafis sebagai Media Perkuliahan Pemodelan dan Simulasi Jaringan dirasakan tepat guna untuk Kelas Reguler dan Karyawan. Hal tersebut merupakan upaya penulis agar mahasiswa peserta kelas Pemodelan dan Simulasi Jaringan dapat mempelajari simulasi perilaku *traffic* jaringan komputer pada setiap lapisannya tanpa harus memiliki ketergantungan pada *software prioprietary* simulator berbayar. Berdasarkan kondisi tersebut maka gagasan untuk melaksanakan Pengembangan Program Kuliah melalui pengembangan aplikasi Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis sebagai Media Perkuliahan Pemodelan dan Simulasi Jaringan pada Kelas Reguler dan Karyawan, Program Studi Informatika ini kami susun. Aplikasi simulasi jaringan komputer yang mendeskripsikan perilaku *traffic* pada masing- masing lapisan beserta dengan protokol yang digunakan sangat diperlukan untuk pengukuran akurasi dan efektifitas algoritma pada protokol yang akan diobservasi. Permasalahan beranjak dari lapisan fisik hingga lapisan *datalink* yang dapat diobservasi, yaitu kualitas transmisi dari media yang digunakan, dimana sinyal yang dikirimkan dapat terpengaruh oleh degradasi fisik media [1]. Pada lapisan fisik dengan media *wireless*, kinerja jaringan yang dinyatakan dengan perbandingan pengiriman paket dan *throughput* dapat terpengaruh oleh keberagaman link antara wireless client dengan wireless access point [2].

Kebutuhan simulator jaringan komputer yang bersifat modular, deskriptif, *multi-platform* berbasis pustaka grafis untuk dapat menjelaskan perilaku traffic pada masing-masing lapisan jaringan merupakan keniscayaan pada usulan algoritma dan protokol yang digunakan dalam rangka efisiensi waktu desain topologi. Keberagaman link antara *transmitter* dengan *receiver* menentukan pemenuhan persyaratan kualitas usulan desain jaringan mulai dari penggunaan pada jaringan *Internet of Things*, komputasi

awan hingga *Software Defined Networking* [3].
Berdasarkan beberapa materi yang bersifat *up to date* tersebut, maka penulis memulai proses pengembangan program kuliah dengan metode pendekatan keterbaruan teknologi, dalam hal ini penggunaan aplikasi Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis sebagai Media Perkuliahan Pemodelan dan Simulasi Jaringan pada Kelas Reguler dan Karyawan, Program Studi Informatika yang dilaksanakan pada perkuliahan pada Semester Ganjil 2025/2026 dengan penerapan metode *hybrid learning* kelas secara *offline* pada kelas dan *online*.

Bagian 3

Permasalahan

Ketersediaan aplikasi simulator dalam bentuk model lapisan jaringan komputer, dimulai dari lapisan fisik hingga lapisan aplikasi yang bersifat deskriptif, *free* dan *open source* untuk pembelajaran pemodelan dan simulasi jaringan komputer masih dirasakan kurang. Beberapa aplikasi simulator seperti NS2, OMNET++, Opnet IT Guru Academic Edition [4] [5], dirasakan masih belum dapat memenuhi kebutuhan akademik untuk pemodelan dan simulasi jaringan. Pengembangan program kuliah ini mencoba membahas penggunaan model jaringan komputer dimulai dari lapisan fisik hingga lapisan *datalink* berbasis pustaka grafis Tkinter menggunakan bahasa pemrograman Python yang bersifat *reusable* atau *use-the-battery* dari pustaka eksisting, memiliki variasi obyek pada setiap lapisan. Hal tersebut diperlukan dalam rangka studi lebih lanjut terkait karakteristik dan perilaku *traffic* jaringan komputer pada kedua lapisan dasar untuk memenuhi kebutuhan penyediaan model agar dapat meningkatkan efisiensi waktu desain jaringan komputer.

Berdasarkan uraian beberapa aplikasi simulator pemodelan jaringan komputer tersebut, maka penulis memandang perlu untuk memberikan usulan pengembangan aplikasi pemodelan dan simulasi jaringan komputer yang tepat dan sesuai dengan kondisi pengguna untuk mendukung proses pembelajaran secara *hybrid*, dengan mempertimbangkan lisensi *software* simulator, aspek teknis, pengembangan lebih lanjut dan aspek kemudahan penggunaan berikut portabilitas dengan sistem operasi oleh mahasiswa reguler dan karyawan peserta kelas pembelajaran *online*.

Bagian 4

Tujuan Pengembangan Program Kuliah

Pengembangan Program Kuliah Model Lapisan Fisik dan Datalink Jaringan Komputer ini ditujukan untuk memberikan fokus studi agar dapat tercapai tujuan pembelajaran bagi mahasiswa Kelas Reguler dan Karyawan yang mengikuti perkuliahan *hybrid* dan mengembangkan diri mereka pada masa depan *open source software*. Pemanfaatan aplikasi Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis dirasakan cukup tepat untuk menjawab kebutuhan tersebut agar mahasiswa mendapatkan alternatif solusi untuk menyerap penyampaian materi pembelajaran dengan keterbatasan lisensi aplikasi simulator sehingga tepat guna bagi peserta kelas reguler dan kelas karyawan yang mengikuti perkuliahan hybrid mata kuliah Pemodelan dan Simulasi Jaringan.

Bagian 5

Metode Pengembangan Program Kuliah

Pengembangan program kuliah menggunakan aplikasi Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis untuk Media Perkuliahan Pemodelan dan Simulasi Jaringan pada Kelas Reguler dan Karyawan, maka Program Studi Informatika menggunakan metode *rapid development and hands-on experience*, yaitu penulis menggunakan bahasa pemrogrman Python, pustaka grafis Tkinter, numpy, pandas, scipy, matplotlib dengan memanfaatkan pustaka Python yang telah tersedia. Berdasarkan paparan pada permasalahan, para pengguna aplikasi *network modeling and simulation* hanya memanfaatkan aplikasi yang telah tersedia baik yang bersifat *proprietary* maupun *open source* untuk melakukan *modeling and simulation*, untuk penggunaan dasar, tanpa melakukan eksplorasi lebih lanjut mengenai pengembangan aplikasi pemodelan dan simulasi jaringan dengan bahasa pemrograman yang tersedia. Penulis melakukan eksplorasi dalam rangka membangun simulator pemodelan jaringan komputer agar penulis dapat memanfaatkan pustaka bahasa pemrograman Python dan mengajarkan mahasiswa kelas Reguler dan Karyawan. Para mahasiswa dapat melihat sekaligus mempelajari kode sumber simulator pemodelan jaringan komputer dengan menggunakan aplikasi *text editor* yang telah terinstalasi pada notebook atau PC. Mahasiswa kelas Reguler dan Karyawan yang mengikuti perkuliahan hybrid (*offline*: reguler, *online*:karyawan) dapat melihat kode sumber aplikasi simulator untuk mata kuliah Pemodelan dan Simulasi Jaringan pada portal akademik. Sehingga diharapkan semua kanal yang mereka gunakan dapat membantu dalam mengakses kode sumber dan penjelasan materi perkuliahan Pemodelan dan Simulasi Jaringan.

Bagian 6

Sasaran dan Luaran

Sasaran dan Luaran dari pengembangan program kuliah penggunaan Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis Sebagai Media Perkuliahan Pemodelan dan Simulasi Jaringan pada Kelas Reguler dan Karyawan, Program Studi Informatika, adalah sebagai berikut :

1. Sasaran dari pengembangan program kuliah ini adalah bertambahnya pengetahuan kolega dosen pada Program Studi Informatika atas alternatif usulan dalam penggunaan bahasa pemrograman Python untuk membangun aplikasi Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis yang disampaikan secara *hybrid*, yaitu *offline* bagi mahasiswa kelas Reguler dan *online* bagi mahasiswa kelas Karyawan dalam rangka mengikuti mata kuliah Pemodelan dan Simulasi Jaringan pada Semester Ganjil 2025/2026.
2. Luaran dari pengembangan program kuliah ini adalah tersedianya aplikasi simulator Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis dengan memanfaatkan penggunaan bahasa pemrograman Python.
3. Kemudahan dalam akses melihat kode sumber aplikasi simulator pemodelan lapisan jaringan komputer sebagai materi pembelajaran dirasakan memiliki manfaat yang tinggi bagi mahasiswa, dalam rangka melakukan review materi perkuliahan terutama menjelang dan pada saat Ujian Tengah Semester dan Ujian Akhir Semester.

Bagian 7

Uraian Pengembangan Program Kuliah

Uraian Pengembangan Program Kuliah Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis Sebagai Media Perkuliahan Pemodelan dan Simulasi Jaringan terbagi menjadi :

- Lapisan Fisik
- Lapisan Datalink

7.1 Lapisan Fisik

Pada aplikasi model Lapisan Fisik merupakan aplikasi yang dikembangkan dengan bahasa pemrograman Python untuk mendeskripsikan simulasi pada *Physical Layer* dengan antar muka grafis dengan memanfaatkan pustaka Tkinter, numpy, pandas, scipy dan matplotlib. Berikut ini merupakan listing kode sumber pada file final_physical_layer_snr_research_simulation.py:

Listing 7.1: Physical Layer Simulation

```
#!/usr/bin/python3
import tkinter as tk
from tkinter import ttk, filedialog
import numpy as np
import pandas as pd
from scipy.stats import bernoulli
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# =====================================================
# Physical Layer Model (Fiber / Gigabit / Ethernet / BOTH)
# =====================================================
class PhysicalLayerModel:
def __init__(self, medium, rtt_ms):
self.medium = medium

self.media = {
        "Fiber Optic": {"rate": 10e9, "error": 0.001},
        "Gigabit Ethernet": {"rate": 1e9, "error": 0.01},
        "Ethernet": {"rate": 100e6, "error": 0.03},
```

```
}

self.rtt = rtt_ms / 1000
self.time = 0
self.history = []

def step(self):
records = []

for m in self.media:
if self.medium not in [m, "BOTH"]:
continue

rate = self.media[m]["rate"]
error_prob = self.media[m]["error"]

bits_sent = int(rate * self.rtt)
error = bernoulli.rvs(error_prob)

if error:
bits_ok = int(bits_sent * 0.9)
crc_status = "CRC Error"
else:
bits_ok = bits_sent
crc_status = "CRC OK"

throughput = bits_ok / self.rtt / 1e6  # Mbps

records.append({
        "RTT": self.time,
        "Medium": m,
        "Bits Sent": bits_sent,
        "Bits Received": bits_ok,
        "Throughput (Mbps)": throughput,
        "CRC Status": crc_status,
        "Error": bool(error)
})

for r in records:
self.history.append(r)

self.time += 1

def dataframe(self):
return pd.DataFrame(self.history)

```

```
# ======================================================
# GUI Application
# ======================================================
class PhysicalLayerSimulator(tk.Tk):
def __init__(self):
super().__init__()
self.title("Physical Layer Simulator (Fiber / Gigabit / Ethernet / BOTH)")
self.geometry("1600x980")

self.model = None
self.running = False

self._build_controls()
self._build_plots()
self._build_table()

# ------------------------------------------------
# Controls
# ------------------------------------------------
def _build_controls(self):
frame = ttk.LabelFrame(self, text="Simulation Controls")
frame.pack(side=tk.LEFT, fill=tk.Y, padx=10, pady=10)

self.medium = tk.StringVar(value="BOTH")
self.rtt = tk.IntVar(value=10)

ttk.Label(frame, text="Physical Medium").pack(anchor=tk.W)
ttk.Combobox(
frame,
textvariable=self.medium,
values=["Fiber Optic", "Gigabit Ethernet", "Ethernet", "BOTH"],
state="readonly"
).pack(fill=tk.X)

ttk.Label(frame, text="RTT (ms)").pack(anchor=tk.W, pady=(8, 0))
ttk.Entry(frame, textvariable=self.rtt).pack(fill=tk.X)

ttk.Button(frame, text="Start Simulation", command=self.start).pack(
pady=(15, 5), fill=tk.X
)
ttk.Button(frame, text="Stop Simulation", command=self.stop).pack(
pady=5, fill=tk.X
)
ttk.Button(frame, text="Export CSV", command=self.export_csv).pack(
pady=5, fill=tk.X
)
```

```
113
114 ttk.Separator(frame).pack(fill=tk.X, pady=10)
115
116 ttk.Button(frame, text="Exit Application", command=self.exit_app).pack(
117 pady=5, fill=tk.X
118 )
119
120 self.status = ttk.Label(frame, text="Status: Idle")
121 self.status.pack(pady=10)
122
123 # -----------------------------------------------
124 # Plots
125 # -----------------------------------------------
126 def _build_plots(self):
127 frame = ttk.Frame(self)
128 frame.pack(side=tk.TOP, fill=tk.BOTH, expand=True)
129
130 self.fig, (self.ax1, self.ax2, self.ax3) = plt.subplots(3, 1, figsize=(7, 8))
131
132 self.ax1.set_title("Throughput Evolution")
133 self.ax1.set_ylabel("Throughput (Mbps)")
134 self.ax1.grid(True)
135
136 self.ax2.set_title("Physical Traffic Load (Bits Sent)")
137 self.ax2.set_ylabel("Bits")
138 self.ax2.grid(True)
139
140 self.ax3.set_title("CRC Bit-Level Visualization")
141 self.ax3.set_ylabel("Bit Status")
142 self.ax3.set_xlabel("Bit Index")
143
144 self.canvas = FigureCanvasTkAgg(self.fig, master=frame)
145 self.canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
146
147 # -----------------------------------------------
148 # Per-RTT Table
149 # -----------------------------------------------
150 def _build_table(self):
151 frame = ttk.LabelFrame(self, text="Per-RTT Value Table")
152 frame.pack(side=tk.BOTTOM, fill=tk.X, padx=10, pady=10)
153
154 cols = ("RTT", "Medium", "Bits Sent", "Bits Received",
155 "Throughput (Mbps)", "CRC Status", "Error")
156 self.tree = ttk.Treeview(frame, columns=cols, show="headings", height=8)
157
158 for col in cols:
```

```
self.tree.heading(col, text=col)
self.tree.column(col, width=160)

self.tree.pack(fill=tk.X)

# ------------------------------------------------
# Simulation Logic
# ------------------------------------------------
def start(self):
self.model = PhysicalLayerModel(
medium=self.medium.get(),
rtt_ms=self.rtt.get()
)

self.tree.delete(*self.tree.get_children())
self.running = True
self.status.config(text="Simulation Running (Physical Layer)")
self.run_step()

def stop(self):
self.running = False
self.status.config(text="Simulation Stopped")

def run_step(self):
if not self.running:
return

self.model.step()
df = self.model.dataframe()

# Update plots
self.ax1.clear()
self.ax2.clear()
self.ax3.clear()

for m in df["Medium"].unique():
sub = df[df["Medium"] == m]
self.ax1.plot(sub["RTT"], sub["Throughput (Mbps)"], marker="o", label=m)
self.ax2.plot(sub["RTT"], sub["Bits Sent"], marker="o", label=m)

# CRC visualization (last RTT)
last = df.tail(1).iloc[0]
bits = np.random.choice(
[0, 1], size=50,
p=[0.9 if last["CRC Status"] == "CRC OK" else 0.7,
0.1 if last["CRC Status"] == "CRC OK" else 0.3]
```

```
)
self.ax3.bar(range(len(bits)), bits)

self.ax1.legend()
self.ax2.legend()

self.ax1.set_title("Throughput Evolution")
self.ax2.set_title("Physical Traffic Load Evolution")
self.ax3.set_title("CRC Visualization ({last['CRC Status']})")

self.canvas.draw()

# Insert latest rows
for _, r in df.tail(len(df["Medium"].unique())).iterrows():
self.tree.insert(
"", tk.END,
values=(
r["RTT"], r["Medium"], r["Bits Sent"],
r["Bits Received"],
f"{r['Throughput (Mbps)']:.2f}",
r["CRC Status"], r["Error"]
)
)

self.after(800, self.run_step)

# ------------------------------------------------
# CSV Export
# ------------------------------------------------
def export_csv(self):
if not self.model:
return

df = self.model.dataframe()
if df.empty:
return

path = filedialog.asksaveasfilename(
defaultextension=".csv",
filetypes=[("CSV Files", "*.csv")]
)
if path:
df.to_csv(path, index=False)
self.status.config(text="CSV Exported Successfully")

# ------------------------------------------------
```

```
# Exit
# ------------------------------------------------
def exit_app(self):
self.running = False
self.destroy()

# ======================================================
# Run Application
# ======================================================
if __name__ == "__main__":
        app = PhysicalLayerSimulator()
        app.mainloop()
```

Aplikasi simulasi *Physical Layer* dengan *Graphical User Interface*(GUI) menggunakan pustaka grafis Tkinter. Program ini menyimulasikan performa beberapa media jaringan (Fiber Optic, Gigabit Ethernet, Ethernet) dengan parameter RTT (Round Trip Time), lalu menampilkan hasilnya dalam grafik, tabel, dan dapat diekspor ke CSV. Penjelasan kode sumber aplikasi simulator adalah sebagai berikut:

1. Baris ke-1 hingga baris ke-8 merupakan instruksi import terhadap fungsi-fungsi pada pustaka Tkinter (membuat GUI aplikasi), ttk (widget Tkinter yang lebih modern), Numpy (operasi numerik dan array), Pandas (pengelolaan data dalam bentuk tabel), Scipy (simulasi probabilitas dan error), Matplotlib (pembuatan grafik), FigureCanvasTkAgg (penampilan grafik matplotlib di Tkinter) yang diperlukan aplikasi.

2. Baris ke-13 Class PhysicalLayerModel merupakan class untuk menghitung jumlah bit yang dikirim dan diterima, throughput, status Cyclic Redundancy Check (CRC), kesalahan pada pengiriman.

3. Baris ke-14 hingga baris ke-15 merupakan *constructor* fungsi yang dipanggil pada saat pembuatan obyek. Medium merupakan media yang dipilih, sedangkan *rtt_ms* merupakan *round trip time* dalam *milisecond*.

4. Baris ke-17 merupakan definisi media jaringan, dimana media *Fiber Optic* memiliki *data rate* 10 Gbps dan *error probability* 0.1 persen, *Gigabit Ethernet* memiliki *data rate* 1 Gbps dan *error probability* 1 persen, *Ethernet* memiliki *data rate* 100 Mbps dan *error probability* 3 persen.

5. Baris ke-23 hingga baris ke-25 merupakan pendefinisian variabel waktu simulasi.

6. Baris ke-27 merupakan fungsi yang menjalankan satu langkah simulasi jaringan.

7. Baris ke-30 merupakan simulasi untuk *Fiber Optic, Gigabit Ethernet* dan *Ethernet*.

8. Baris ke-37 merupakan perintah untuk menghitung jumlah bit yang dikirimkan.

9. Baris ke-38 merupakan simulasi *error* menggunakan Bernoulli *Distribution*.

10. Baris ke-40 dan baris ke-43 merupakan perintah ketika terjadi *error*, maka hanya 90 persen bit yang dapat diterima dan CRC mengalami keadaan gagal, bila tidak terjadi *error* maka semua bit diterima.

11. Baris ke-47 merupakan perintah untuk menghitung *throughput* dan memiliki satuan Mbps.

12. Baris ke-49 hingga baris ke-57 merupakan perintah untuk menyimpan hasil simulasi.

13. Baris ke-64 merupakan perintah untuk mengubah history pada Pandas DataFrame.

14. Baris ke-70 merupakan perintah class untuk membuat Graphical User Interface (GUI) aplikasi, dimana aplikasi memiliki *control panel*, grafik, tabel dan eksport hasil simulasi menjadi format *Comma Separated Value* (CSV).

15. Baris ke-79 hingga baris ke-145 merupakan perintah untuk kontrol simulasi, berupa pemilihan media, input RTT, tombol kontrol hingga plotting grafik yaitu throughput, traffic load, CRC visualization.

16. Baris ke-150 hingga baris ke-162 merupakan perintah untuk menampilkan hasil simulasi Round Trip Time (RTT).

17. Baris ke-167 hingga baris ke-180 merupakan perintah untuk memulai simulasi, terdiri dari membuat model simulasi, reset tabel, set *status running* dan menjalankan simulasi tahapan pertama.

18. Baris ke-182 hingga baris ke-192 merupakan perintah untuk menjalankan *loop* simulasi utama dengan tahapan menjalankan model, mengambil *dataframe*, update grafik dan *update* tabel.

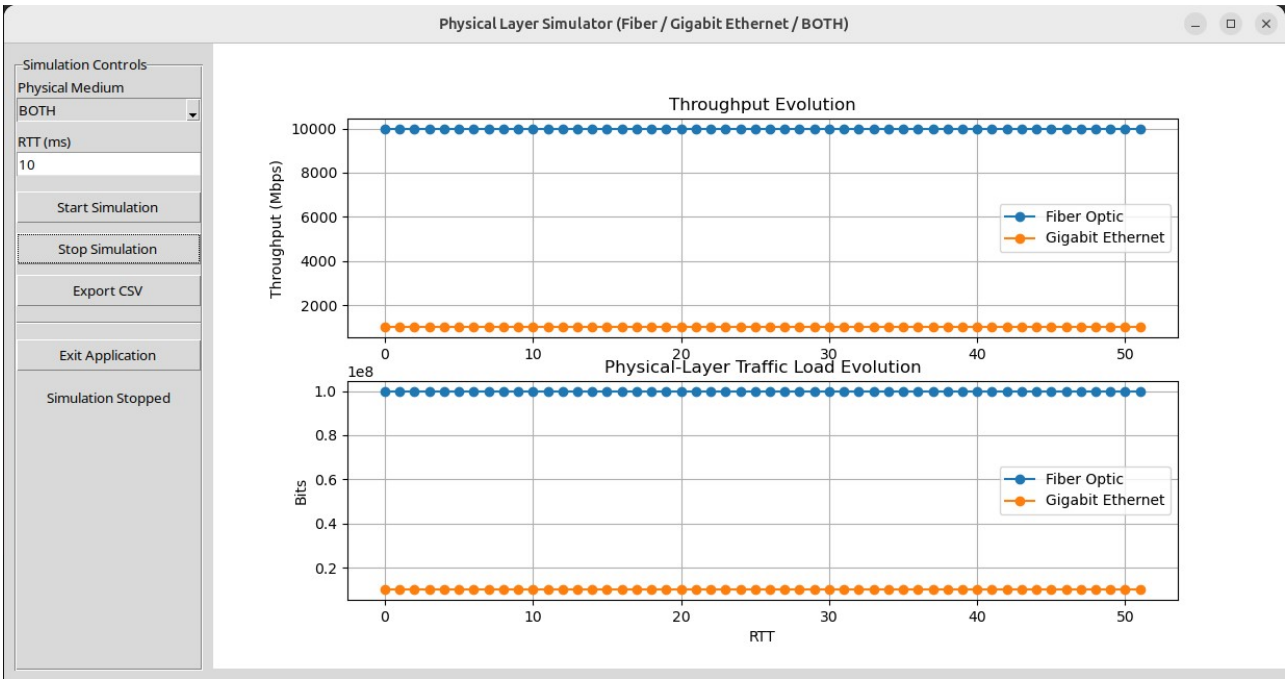
19. Baris ke-196 hingga baris ke-229 merupakan perintah untuk menjalankan update grafik *throughput*, update grafik *traffic*, visualisasi CRC dengan interval simulasi.

20. Baris ke-234 hingga baris ke-248 merupakan perintah untuk menjalankan ekspor hasil simulasi ke format CSV, dengan tahapan mengambil dataframe, membuka dialog *save file*, menyimpan data dalam format CSV.

21. Baris ke-253 hingga baris ke-255 merupakan baris perintah untuk menghentikan aplikasi dan menutup aplikasi.

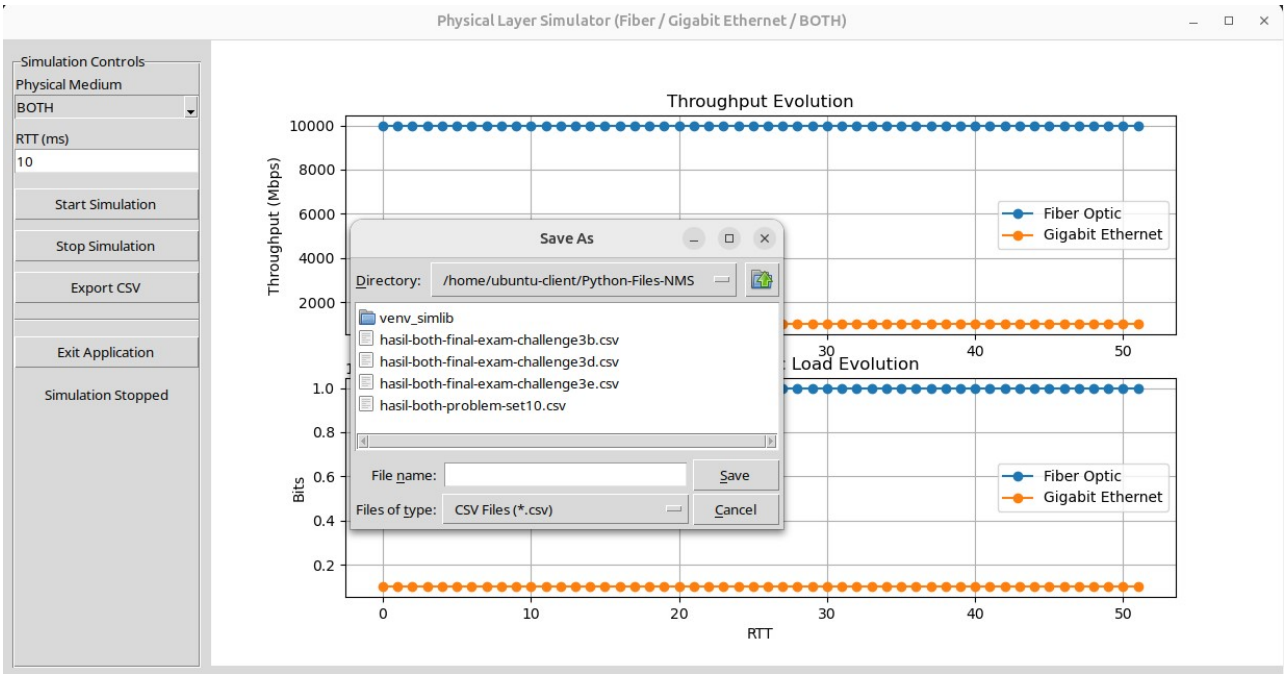
22. Baris ke-260 hingga baris ke-262 merupakan perintah untuk menjalankan program utama dengan menjalankan *event loop* Tkinter.

Setelah aplikasi Physical Layer Simulator dijalankan, maka akan terlihat seperti terlihat pada Gambar 7.1 berikut ini: Untuk proses ekspor ke format CSV pada kedua medium yang divisualisasikan, maka



Gambar 7.1: Tampilan aplikasi Physical Layer Simulator untuk kedua medium

akan terlihat seperti pada Gambar 7.2 berikut ini:



Gambar 7.2: Tampilan ekspor hasil simulasi aplikasi Physical Layer Simulator menjadi format CSV

7.2 Lapisan Datalink

Aplikasi simulator Lapisan Datalink merupakan aplikasi yang dikembangkan dengan bahasa pemrograman Python untuk mendeskripsikan simulasi pada *Datalink Layer* dengan antar muka grafis dengan memanfaatkan pustaka Tkinter, pustaka grafis native untuk pengembangan aplikasi. Berikut ini merupakan listing kode sumber pada file final_datalink_simulation.py:

Listing 7.2: Datalink Layer Simulation

```
#!/usr/bin/python3
import tkinter as tk
from tkinter import ttk, filedialog
import numpy as np
import pandas as pd
from scipy.stats import bernoulli
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


# ====================================================
# Data Link Layer Model (MAC / LLC / BOTH)
# ====================================================
class DataLinkLayerModel:
def __init__(self, protocol, rtt_ms, payload_bytes):
self.protocol = protocol

# Ethernet constants (bytes)
self.preamble = 8
```

```
self.header = 14
self.fcs = 4

# Error detection probability (CRC-style)
self.error_prob = 0.1

self.payload = payload_bytes
self.frame_size = self.preamble + self.header + self.payload + self.fcs
self.frame_bits = self.frame_size * 8

self.rtt = rtt_ms / 1000
self.window_mac = 1
self.window_llc = 1
self.time = 0
self.history = []

def step(self):
error = bernoulli.rvs(self.error_prob)

# ---------------- MAC ----------------
if self.protocol in ["MAC", "BOTH"]:
if error:
self.window_mac = max(1, self.window_mac // 2)
state_mac = "MAC: Error Detected (Retransmission)"
else:
self.window_mac += 1
state_mac = "MAC: Frame Transmitted"

throughput_mac = (self.window_mac * self.frame_bits) / self.rtt / 1e6
self.history.append({
        "RTT": self.time,
        "Layer": "MAC",
        "Frame Size (bytes)": self.frame_size,
        "Window": self.window_mac,
        "Throughput (Mbps)": throughput_mac,
        "Error": error,
        "State": state_mac
})

# ---------------- LLC ----------------
if self.protocol in ["LLC", "BOTH"]:
self.window_llc = 1
throughput_llc = (self.frame_bits / self.rtt) / 1e6

if error:
throughput_llc *= 0.5
```

```
state_llc = "LLC: Error Handling"
else:
state_llc = "LLC: Control Frame"

self.history.append({
        "RTT": self.time,
        "Layer": "LLC",
        "Frame Size (bytes)": self.frame_size,
        "Window": self.window_llc,
        "Throughput (Mbps)": throughput_llc,
        "Error": error,
        "State": state_llc
})

self.time += 1

def dataframe(self):
return pd.DataFrame(self.history)


# ======================================================
# GUI Application
# ======================================================
class DataLinkLayerSimulator(tk.Tk):
def __init__(self):
super().__init__()
self.title("Data Link Layer Simulator (MAC / LLC / BOTH)")
self.geometry("1550x920")

self.model = None
self.running = False

self._build_controls()
self._build_plots()
self._build_table()

# ------------------------------------------------
# Controls
# ------------------------------------------------
def _build_controls(self):
frame = ttk.LabelFrame(self, text="Simulation Controls")
frame.pack(side=tk.LEFT, fill=tk.Y, padx=10, pady=10)

self.protocol = tk.StringVar(value="BOTH")
self.rtt = tk.IntVar(value=100)
self.payload = tk.IntVar(value=1000)
```

```
112
113 ttk.Label(frame, text="Data Link Sublayer").pack(anchor=tk.W)
114 ttk.Combobox(
115 frame,
116 textvariable=self.protocol,
117 values=["MAC", "LLC", "BOTH"],
118 state="readonly"
119 ).pack(fill=tk.X)
120
121 ttk.Label(frame, text="RTT (ms)").pack(anchor=tk.W, pady=(8, 0))
122 ttk.Entry(frame, textvariable=self.rtt).pack(fill=tk.X)
123
124 ttk.Label(frame, text="Payload Size (bytes)").pack(anchor=tk.W, pady=(8, 0))
125 ttk.Entry(frame, textvariable=self.payload).pack(fill=tk.X)
126
127 ttk.Button(frame, text="Start Simulation", command=self.start).pack(
128 pady=(15, 5), fill=tk.X
129 )
130 ttk.Button(frame, text="Stop Simulation", command=self.stop).pack(
131 pady=5, fill=tk.X
132 )
133 ttk.Button(frame, text="Export CSV", command=self.export_csv).pack(
134 pady=5, fill=tk.X
135 )
136
137 ttk.Separator(frame).pack(fill=tk.X, pady=10)
138
139 ttk.Button(frame, text="Exit Application", command=self.exit_app).pack(
140 pady=5, fill=tk.X
141 )
142
143 self.status = ttk.Label(frame, text="Status: Idle")
144 self.status.pack(pady=10)
145
146 # -----------------------------------------------
147 # Plots
148 # -----------------------------------------------
149 def _build_plots(self):
150 frame = ttk.Frame(self)
151 frame.pack(side=tk.TOP, fill=tk.BOTH, expand=True)
152
153 self.fig, (self.ax1, self.ax2) = plt.subplots(2, 1, figsize=(7, 6))
154
155 self.ax1.set_title("Throughput Evolution")
156 self.ax1.set_ylabel("Throughput (Mbps)")
157 self.ax1.grid(True)
```

```
158
159 self.ax2.set_title("Data Link Traffic Load Evolution")
160 self.ax2.set_xlabel("RTT")
161 self.ax2.set_ylabel("Window Size")
162 self.ax2.grid(True)
163
164 self.canvas = FigureCanvasTkAgg(self.fig, master=frame)
165 self.canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
166
167 # ------------------------------------------------
168 # Per-RTT Table
169 # ------------------------------------------------
170 def _build_table(self):
171 frame = ttk.LabelFrame(self, text="Per-RTT Value Table")
172 frame.pack(side=tk.BOTTOM, fill=tk.X, padx=10, pady=10)
173
174 cols = ("RTT", "Layer", "Frame Size (bytes)",
175 "Window", "Throughput (Mbps)", "Error", "State")
176 self.tree = ttk.Treeview(frame, columns=cols, show="headings", height=8)
177
178 for col in cols:
179 self.tree.heading(col, text=col)
180 self.tree.column(col, width=150)
181
182 self.tree.pack(fill=tk.X)
183
184 # ------------------------------------------------
185 # Simulation Logic
186 # ------------------------------------------------
187 def start(self):
188 self.model = DataLinkLayerModel(
189 protocol=self.protocol.get(),
190 rtt_ms=self.rtt.get(),
191 payload_bytes=self.payload.get()
192 )
193
194 self.tree.delete(*self.tree.get_children())
195 self.running = True
196 self.status.config(text="Simulation Running (Data Link Layer)")
197 self.run_step()
198
199 def stop(self):
200 self.running = False
201 self.status.config(text="Simulation Stopped")
202
203 def run_step(self):
```

```
204 if not self.running:
205 return
206
207 self.model.step()
208 df = self.model.dataframe()
209
210 # Update plots
211 self.ax1.clear()
212 self.ax2.clear()
213
214 for layer in df["Layer"].unique():
215 sub = df[df["Layer"] == layer]
216 self.ax1.plot(sub["RTT"], sub["Throughput (Mbps)"], marker="o", label=layer)
217 self.ax2.plot(sub["RTT"], sub["Window"], marker="o", label=layer)
218
219 self.ax1.set_title("Throughput Evolution")
220 self.ax1.set_ylabel("Throughput (Mbps)")
221 self.ax1.legend()
222 self.ax1.grid(True)
223
224 self.ax2.set_title("Data Link Traffic Load Evolution")
225 self.ax2.set_xlabel("RTT")
226 self.ax2.set_ylabel("Window Size")
227 self.ax2.legend()
228 self.ax2.grid(True)
229
230 self.canvas.draw()
231
232 # Insert latest RTT rows
233 for _, r in df.tail(len(df["Layer"].unique())).iterrows():
234 self.tree.insert(
235 "", tk.END,
236 values=(
237 r["RTT"],
238 r["Layer"],
239 r["Frame Size (bytes)"],
240 r["Window"],
241 f"{r['Throughput (Mbps)']:.2f}",
242 r["Error"],
243 r["State"]
244 )
245 )
246
247 self.after(800, self.run_step)
248
249 # ------------------------------------------------
```

```
# CSV Export
# ------------------------------------------------
def export_csv(self):
if not self.model:
return

df = self.model.dataframe()
if df.empty:
return

file_path = filedialog.asksaveasfilename(
defaultextension=".csv",
filetypes=[("CSV Files", "*.csv")]
)
if file_path:
df.to_csv(file_path, index=False)
self.status.config(text="CSV Exported Successfully")

# ------------------------------------------------
# Exit
# ------------------------------------------------
def exit_app(self):
self.running = False
self.destroy()

# ======================================================
# Run Application
# ======================================================
if __name__ == "__main__":
        app = DataLinkLayerSimulator()
        app.mainloop()
```

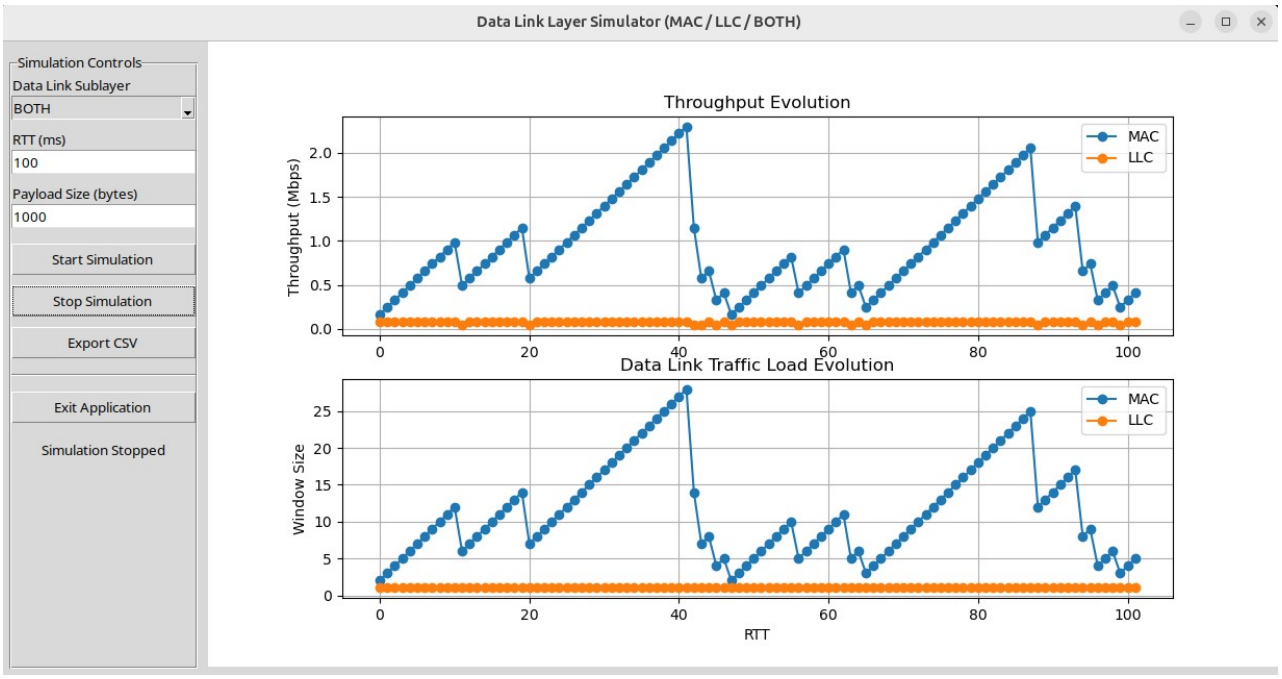
Aplikasi simulator Data Link Layer (Layer 2 OSI) dengan GUI menggunakan pustaka grafis Tkinter. Program ini menyimulasikan perilaku dua sublayer Data Link, yaitu Medium Access Control (MAC) dan Logical Link Control, untuk memperlihatkan bagaimana *frame* dikirim, bagaimana *error* ditangani, bagaimana *throughput* berubah, serta menampilkan hasilnya dalam grafik dan tabel, dan aplikasi simulator ini dapat melakukan ekspor hasil simulasi dalam format CSV. Penjelasan kode sumber aplikasi simulator adalah sebagai berikut:

1. Baris ke-1 hingga baris ke-8 merupakan instruksi import terhadap fungsi-fungsi pada pustaka Tkinter (membuat GUI aplikasi), ttk (widget Tkinter yang lebih modern), Numpy (operasi numerik dan array), Pandas (pengelolaan data dalam bentuk tabel), Scipy (simulasi probabilitas dan error), Matplotlib (pembuatan grafik), FigureCanvasTkAgg (penampilan grafik matplotlib di Tkinter) yang diperlukan aplikasi.

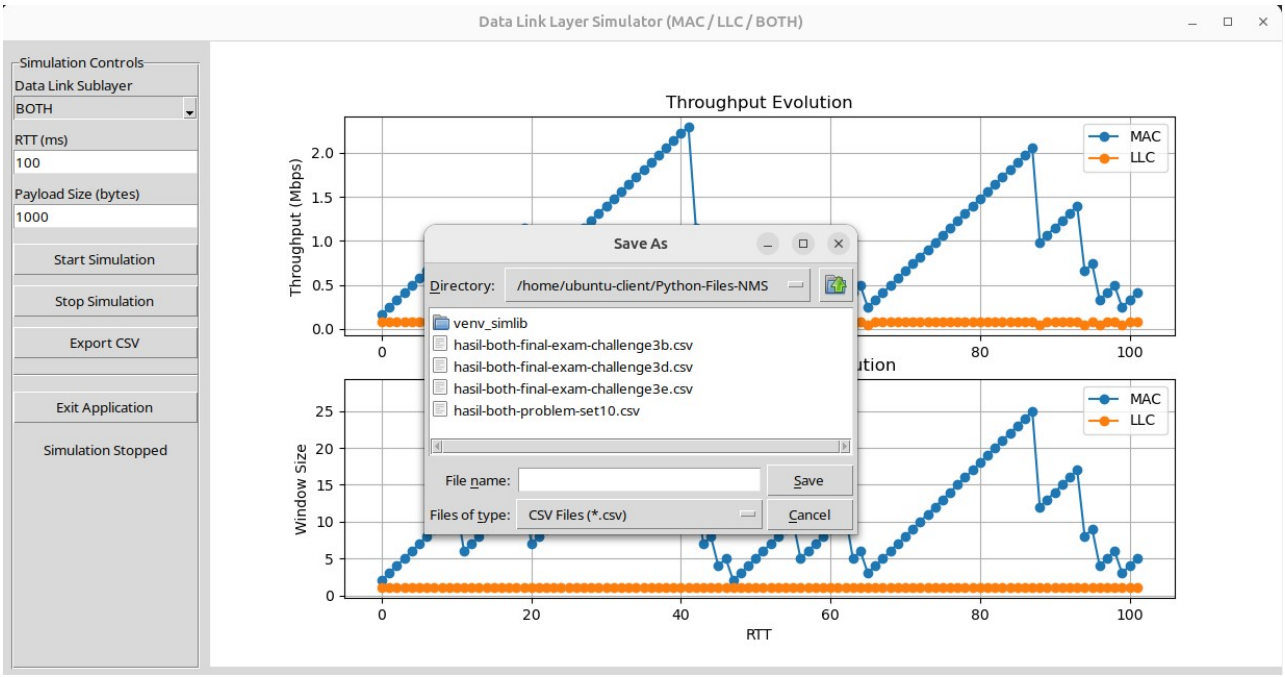
2. Baris ke-14 merupakan perintah untuk menjalankan class DataLinkLayerModel yang melakukan simulasi pengiriman frame, menghitung throughput, mengelola window size, menyimulasikan deteksi kesalahan.

3. Baris ke-15 hingga baris ke-21 merupakan perintah konstruktor dan konstanta ethernet frame.
4. Baris ke-24 merupakan simulasi probabilitas *error* menggunakan Bernoulli *distribution*.
5. Baris ke-27 hingga baris ke-28 merupakan perintah untuk menghitung ukuran frame.
6. Baris ke-30 hingga baris ke-34 merupakan perintah untuk menjelaskan parameter simulasi.
7. Baris ke-36 merupakan perintah untuk menjalankan satu langkah simulasi.
8. Baris ke-37 merupakan perintah untuk menjalankan Distribusi Bernoulli yang menghasilkan nilai 1 untuk mengindikasikan terjadinya error dan nilai 0 untuk mengindikasikan tidak terdapat error.
9. Baris ke-40 hingga baris ke-46 merupakan perintah untuk simulasi MAC layer.
10. Baris ke-48 merupakan perintah untuk menghasilkan throughput MAC.
11. Baris ke-49 hingga baris ke-57 merupakan perintah untuk menyimpan data MAC.
12. Baris ke-60 hingga baris ke-62 merupakan perintah untuk menjalankan simulasi LLC layer.
13. Baris ke-64 hingga baris ke-68 merupakan perintah untuk menghasilkan throughput LLC.
14. Baris ke-70 hingga baris ke-78 merupakan perintah untuk menyimpan data LLC.
15. Baris ke-82 hingga baris ke-83 merupakan perintah untuk menjalankan fungsi dataframe yang mengubah history menjad Pandas frame.
16. Baris ke-89 hingga baris ke-165 merupakan perintah untuk menghasilkan GUI aplikasi simulator dengan komponen utama control panel, grafik dan tabel hasil.
17. Baris ke-170 hingga baris ke-182 merupakan perintah untuk menampilkan tabel per Round Trip Time (RTT).
18. Baris ke-187 hingga baris ke-192 merupakan perintah untuk mulai menjalankan simulasi dengan tahapan yaitu membuat model simulasi, membersihkan tabel dan menjalankan loop simulasi.
19. Baris ke-203 hingga baris ke-245 merupakan perintah untuk menjalankan loop utama simulasi, dengan tahapan menjalankan model, mengambil data dataframe, melakukan update grafik dan melakukan update tabel.
20. Baris ke-247 merupakan baris perintah untuk menjalankan interval simulasi.
21. Baris ke-252 hingga baris ke-266 merupakan perintah untuk menjalankan ekspor ke format CSV.
22. Baris ke-271 hingga baris ke-273 merupakan perintah untuk keluar dari aplikasi.
23. Baris ke-278 hingga baris ke-280 merupakan perintah untuk menjalankan program utama.

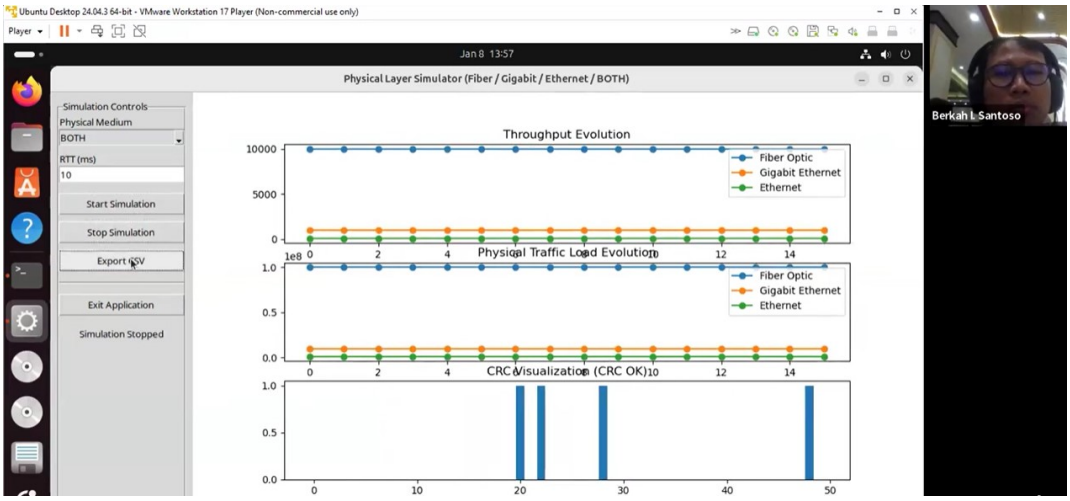
Setelah aplikasi Datalink Layer Simulator dijalankan, maka akan terlihat seperti terlihat pada Gambar 7.3 berikut ini: Untuk proses ekspor ke format CSV pada kedua medium yang divisualisasikan, maka akan terlihat seperti pada Gambar 7.4 berikut ini: Aplikasi simulator Model Lapisan Fisik dan Datalink Jaringan Komputer Berbasis Pustaka Grafis Bahasa Pemrograman Python telah penulis masukkan pada materi final examination pada Mata Kuliah Pemodelan dan Simulasi Jaringan Semester Ganjil 2025/2026, seperti terlihat pada Gambar 7.5 berikut ini:



Gambar 7.3: Tampilan aplikasi Datalink Layer Simulator untuk kedua sub layer



Gambar 7.4: Tampilan ekspor hasil simulasi aplikasi Datalink Layer Simulator menjadi format CSV



Gambar 7.5: Tampilan materi final examination Physical dan Datalink Layer Simulator

Bibliography

[1] V. V. Garbhapu, C. Ware, and M. Lourdiane, "Physical-layer-aware network simulator for future optical functionalities," in *2023 14th International Conference on Network of the Future (NoF)*, 2023, pp. 28–36. 3

[2] M. F. Monir and T. A. Ishmam, "Exploiting link diversity in ieee 802.11 wlan using omnet++," in *2022 IEEE 10th Region 10 Humanitarian Technology Conference (R10-HTC)*, 2022, pp. 355–359. 3

[3] A. O. Nyanteh, M. Li, M. F. Abbod, and H. Al-Raweshidy, "Cloudsimhypervisor: Modeling and simulating network slicing in software-defined cloud networks," *IEEE Access*, vol. 9, pp. 72 484–72 498, 2021. 4

[4] M. Drajic and I. Kokic, "A comparative analysis of simulators ns2 and opnet it guru academic edition," in *2012 20th Telecommunications Forum (TELFOR)*, 2012, pp. 1780–1783. 5

[5] X. Xian, W. Shi, and H. Huang, "Comparison of omnet++ and other simulator for wsn simulation," in *2008 3rd IEEE Conference on Industrial Electronics and Applications*, 2008, pp. 1439–1443. 5