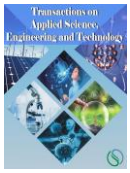

Appendix 1

Code for latency calculation

```
import matplotlib.pyplot as plt
import pandas as pd
# Parameters
packet_sizes = [10, 100, 1000, 10000, 100000] # Packet size in bits
data_rate = 1.54e6 # Data rate in bits/sec (1.54 Mbps)
num_links = 3 # Number of links in the channel (BS -> Relay -> UAV -> User)
# Calculate the delay for each packet size
delays = [(packet_size / data_rate) * num_links * 1000 for packet_size in packet_sizes] # Delay in ms
# Create data table
data = {
"Packet Size (bits)": packet_sizes,
"Total Delay (ms)": delays
}
df = pd.DataFrame(data)
print("Data table for plotting the graph:")
print(df)
# Plotting the graph
plt.figure(figsize=(10, 6))
plt.plot(packet_sizes, delays, marker='o', color='b', linestyle='-')
plt.xlabel("Packet Size (bits)")
plt.ylabel("Total Delay (ms)")
plt.xscale("log") # Logarithmic scale for packet size
plt.title("Packet transmission delay as a function of packet size")
plt.grid(True)
plt.show()
```

Appendix 2

Code for utilization calculation

```
import matplotlib.pyplot as plt
import pandas as pd
# Given parameters
packet_sizes = [10, 100, 1000, 10000, 100000] # Packet sizes in bits
users = [1, 3, 5]
# Number of users
channel_bandwidth = 1.54e6
# Channel bandwidth in bits/s (1.54 Mbps)
# Table to store data
data = {'Packet Size (bits)': packet_sizes}
for user_count in users:
load_percent = [(packet_size * user_count / channel_bandwidth) * 100 for packet_size in packet_sizes]
data[f'Load (%) for {user_count} user(s)'] = load_percent
# Create a DataFrame for the table
df = pd.DataFrame(data)
print("Data table for plotting graphs:")
print(df)
# Plotting graphs
plt.figure(figsize=(10, 6))
for user_count in users:
load_percent = [(packet_size * user_count / channel_bandwidth) * 100 for packet_size in packet_sizes]
plt.plot(packet_sizes, load_percent, label=f"{user_count} user(s)")
# Plot settings
plt.xlabel("Packet size (bits)")
plt.ylabel("Channel load (%)")
plt.title("Channel load dependence on packet size and number of users")
plt.legend()
plt.xscale('log') # Logarithmic scale for clarity
plt.grid(True)
plt.show()
```

Appendix 3

Code for calculation of utilization on data transfer rate

```
import matplotlib.pyplot as plt
import pandas as pd
# Given parameters
packet_size = 100000 # Packet size in bits (100 Kbits)
num_users = 5 # Number of users
data_rates = [1.544e6, 2.048e6, 4e6, 6e6, 10e6, 34e6, 45e6] # Data rates in bits/sec
# Calculate channel load for each rate
load_percentages = [(packet_size * num_users / rate) * 100 for rate in data_rates]
# Create data frame
data = {
"Data Rate (Mbps)": [rate / 1e6 for rate in data_rates],
"Channel Load (%)": load_percentages
}
df = pd.DataFrame(data)
print("Data frame for plotting a graph:")
print(df)
# Plotting a graph
plt.figure(figsize=(10, 6))
plt.plot(data["Data Rate (Mbps)"], data["Channel Load (%)"], marker='o', color='b', linestyle='-')
plt.xlabel("Data Transfer Rate (Mbps)")
plt.ylabel("Channel Load (%)")
plt.title("Dependence of Channel Load on Data Transfer Rate")
plt.grid(True)
plt.show()
```

Appendix 4

Code for calculation of utilization on BER

```
import matplotlib.pyplot as plt
import pandas as pd
# Parameters
packet_size = 10000 # Packet size in bits (10K)
num_users = 5 # Number of users
data_rate = 45e6 # Data rate in bits/sec (45M)
ber_values = [0, 0.01, 0.02, 0.03, 0.04, 0.05] # Bit error rates in %
# Calculate channel load for each BER value
channel_loads = [(packet_size * num_users / (data_rate * (1 - ber))) * 100 for ber in ber_values]
# Create data table
data = {
"BER (%)": ber_values,
"Channel Load (%)": channel_loads
}
df = pd.DataFrame(data)
print("Data table for plotting the graph:")
print(df)
# Plotting the graph
plt.figure(figsize=(10, 6))
plt.plot(data["BER (%)"], data["Channel Load (%)"], marker='o', color='b', linestyle='-')
plt.xlabel("BER (%)")
plt.ylabel("Channel Load (%)")
plt.title("Dependence of channel load on the bit error rate (BER)")
plt.grid(True)
plt.show()
```

Appendix 5

Code for adaptive data transfer modelling

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import time
# Dependency parameters
```

```
data_rates = [6, 10, 45] # Data Rates in Mbps
slopes = [0.0005, 0.0003, 0.00007] # Slope coefficients for lag y vs batch size d
# Create training data
data = []
for rate, slope in zip(data_rates, slopes):
for d in range(100, 10000, 100): # Batch size with 100-bit increments
y = slope * d # Calculate lag
data.append([y, rate, d])
# Convert data to DataFrame for model training
df = pd.DataFrame(data, columns=["y", "DataRate", "d"])
# Split into features and target variable
X = df[["y", "DataRate"]]
y = df["d"]
# Train linear regression model
model = LinearRegression()
model.fit(X, y)
# Predict packet size d for y = 2 ms and Data Rate = 6 Mbps
y_test = 2
data_rate_test = 6
predicted_d = model.predict([[y_test, data_rate_test]])
predicted_d = int(round(predicted_d[0])) # Rounding
# Adapt transmission parameters
d = predicted_d - 1000 # Decrease by 1000 bits
y_values = []
d_values = []
time_values = []
adaptation_count = 0
print(f"Initial packet size: {d} bits")
while adaptation_count < 3:
# Calculate latency for current packet size
y = 0.0005 * d if data_rate_test == 6 else (0.0003 * d if data_rate_test == 10 else 0.00007 * d)
# Write data for graphs
time_values.append(len(time_values))
y_values.append(y)
d_values.append(d)
# Print information when adaptation threshold is reached
if y >= 2:
print(f"Adaptation threshold reached: latency {y:.2f} ms with packet size {d} bits")
adaptation_count += 1
d -= 400 # Reduce packet size after threshold is reached
else:
d += 100 # Increasing the packet size
time.sleep(1) # Delay for data transfer simulation (1 minute for each step)
# Plotting y(t) and d(t) dependencies
fig, axs = plt.subplots(2, 1, figsize=(10, 8))
# Plotting y(t) dependency
axs[0].plot(time_values, y_values, marker="o", color="b")
axs[0].set_title("Dependence of y(t)")
axs[0].set_xlabel("Time (minutes)")
axs[0].set_ylabel("Delay y (ms)")
# Plotting d(t) dependency
axs[1].plot(time_values, d_values, marker="o", color="g")
axs[1].set_title("Dependence of packet size d(t)")
axs[1].set_xlabel("Time (minutes)")
axs[1].set_ylabel("Packet size d (bits)")
plt.tight_layout()
plt.show()
# Output data for building a table
table_data = pd.DataFrame({"Time (minutes)": time_values, "Delay y (ms)": y_values, "Packet   size d (bits)": d_values})
print("\nData table for graphs:")
print(table_data)
```