NTNU
Kunnskap for en bedre verden

Department of Information Security and Communication Technology

TTM4158 - Dependable Performance Design

Basic Concepts

Author:
Vetle Pettersen
Magnus Brox-Einarsen
Even Johan Pereira Haslerud

29.09.2025

Table of Contents

List of Figures **i**

List of Tables **i**

1 Task 1.1: Basic Calculation and Plotting **1**

2 Task 1.2: Probability Theory — Football Scenario **3**

3 Task 1.3: Network Dependability **3**

4 Task 1.4: Poisson Process – Bus Stop Scenario **7**

4.0.1 Code Implementation 8

5 Task 1.5: Stochastic Process – Multiple Access **9**

6 Task 1.6: Little's Law in the Department **9**

List of Figures

1 (i) Poisson PMF and CDF for $\lambda = 1$. 1

2 (ii) Poisson PMF and CDF for $\lambda = 0.1$. 1

3 (i) Exponential PDF and CDF for $\mu = 1$. 2

4 (ii) Exponential PDF and CDF for $\mu = 10$. 2

5 $W(\lambda, \mu)$ as a function of μ (left) and λ (right). 2

6 Illustrate G_c 4

7 Illustrate: G_r 5

8 Illustrating G_d 6

9 Illustrating G_t 6

10 Poisson distribution of the number of passengers in 15 minutes. 7

11 Exponential distribution of inter-arrival times between passengers. 8

List of Tables

1 Average node degree for the four networks. 4

1 Task 1.1: Basic Calculation and Plotting

(a) Poisson Distribution

For $X \sim \text{Poisson}(\lambda)$ we plotted the PMF $P(X = x)$ and CDF $F_X(x)$.

- (i) $\lambda = 1$: Mass is concentrated at small x, peaks at $x = 0, 1$, and decays quickly. The CDF exceeds 0.98 by $x = 4$.
- (ii) $\lambda = 0.1$: Probability ≈ 0.90 at $x = 0$, very small beyond $x = 1$. The CDF jumps to nearly 1 by $x = 2$.

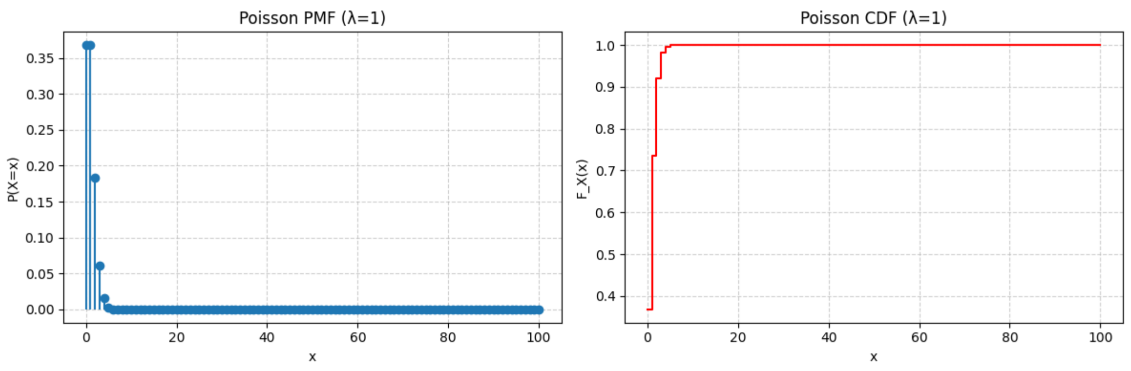

Figure 1: (i) Poisson PMF and CDF for $\lambda = 1$.

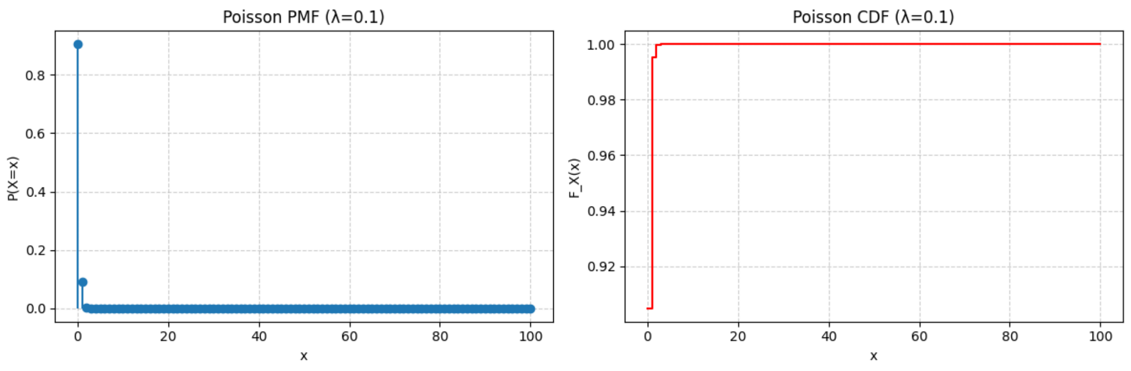

Figure 2: (ii) Poisson PMF and CDF for $\lambda = 0.1$.

(b) Exponential Distribution

For $Y \sim \text{Exp}(\mu)$ with PDF $f_Y(y) = \mu e^{-\mu y}$ and CDF $F_Y(y) = 1 - e^{-\mu y}$:

- (i) $\mu = 1$: Decay is moderate, CDF rises gradually.
- (ii) $\mu = 10$: Much faster decay, CDF jumps quickly to 1 within $y < 0.5$.

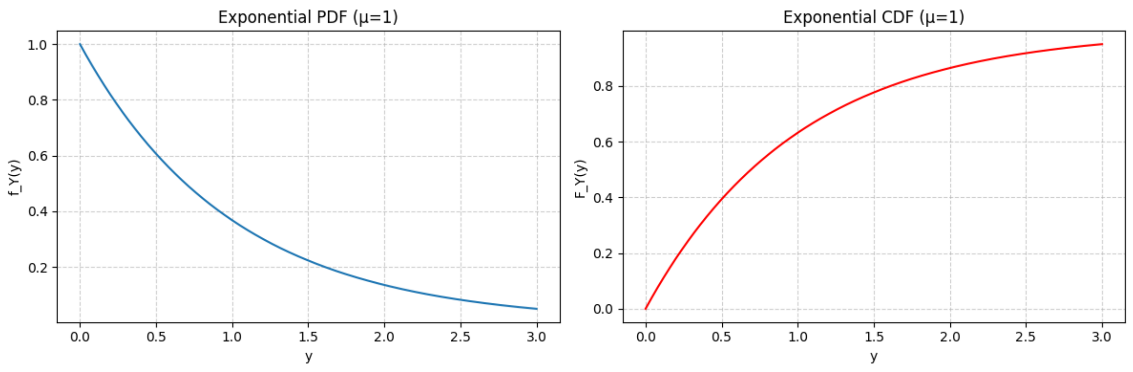

Figure 3: (i) Exponential PDF and CDF for $\mu = 1$.

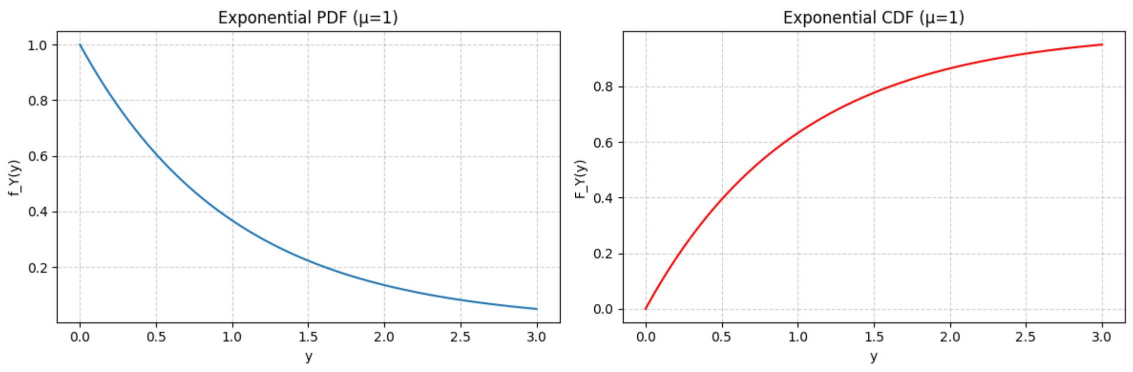

Figure 4: (ii) Exponential PDF and CDF for $\mu = 10$.

(c) Function W

We define

$$W(\lambda, \mu) = \frac{1}{\mu - \lambda}, \quad \mu > \lambda.$$

- (i) $W(1, 10) = 0.1111$, $W(0.1, 1) = 1.1111$. Case 2 is larger since $\mu - \lambda$ is smaller.
- (ii) With $\lambda = 1$, W decreases as μ increases (from 10 at $\mu = 1.1$ towards 0).
- (iii) With $\mu = 1$, W increases sharply as $\lambda \to 1$ (e.g. $W(0.9) = 10$).

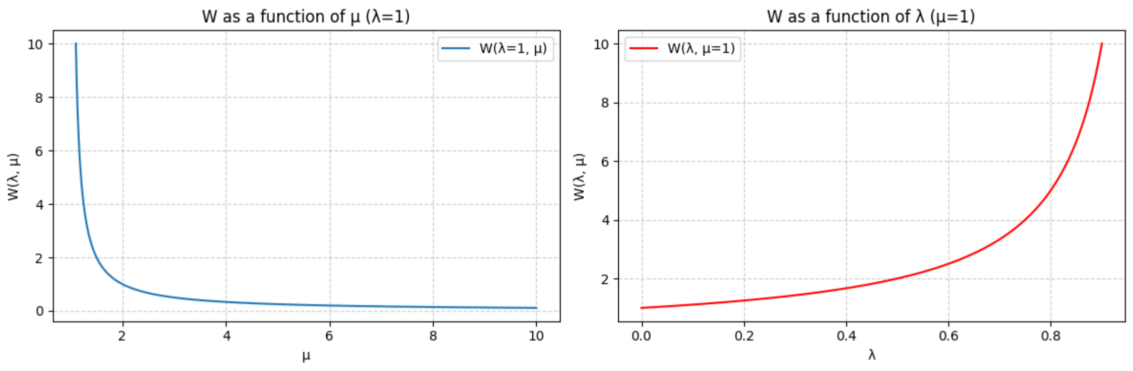

Figure 5: $W(\lambda, \mu)$ as a function of μ (left) and λ (right).

2 Task 1.2: Probability Theory — Football Scenario

Each team scores as $X_{\text{team}} \sim \text{Poisson}(\lambda \cdot 1.5) = \text{Poisson}(0.75)$, independently. Thus, total goals $N = X_{\text{Nor}} + X_{\text{Swe}} \sim \text{Poisson}(1.5)$.

(a) Probability of exactly 2 total goals:

$$P(N = 2) = e^{-1.5}\frac{1.5^2}{2!} \approx 0.251.$$

(b) Probability the result is 1:1:

$$P(X_{\text{Nor}} = 1)\,P(X_{\text{Swe}} = 1) = \left(e^{-0.75}\frac{0.75^1}{1!}\right)^2 \approx 0.126.$$

Notes: $\mathbb{E}[X_{\text{team}}] = 0.75$ per team, so $\mathbb{E}[N] = 1.5$ total.

Code Implementation

Python code used to implement the computations can be observed below:

```python

mu = 0.75 # The expected number of goals per match
print(f"Expected number of goals in a match is {mu}")
lam = 0.5 # Poisson process intensity
print(f"Poisson process intensity is {lam}")
mu2 = mu*2 #Expected numbers of goals in two poisson processes
print(f"Expected number of goals in two matches is {mu2}")
k1 = 2 #number of goals

P_2goals = (np.exp(-mu2)*mu2**k1)/(m.factorial(k1)) #probability of two goals
    during the match
print(f"The probability of scoring two goals during the match is: {P_2goals:.3f}")

# Task b
# We will calculate the probability of both Norway and Sweeden scores 1 goal each
k2 = 1 #number of goals for each team

P_1goal = (np.exp(-mu)*mu**k2)/(m.factorial(k2)) #probability of one goal during
    the match


P_even = P_1goal*P_1goal #probability of both teams scoring one goal each
print(f"The probability of both teams scoring one goal and the match is even: {
    P_even:.3f}")
```

Listing 1: Probability Theory

3 Task 1.3: Network Dependability

(a) Average node degree

The average node degree is given by

$$\bar{k} = \frac{2|E|}{|V|}.$$

Network	#Nodes $|V|$	#Edges $|E|$	Average Degree $\bar{k}$
G_c	7	6	$12/7 \approx 1.714$
G_r	7	7	$14/7 = 2.000$
G_d	7	8	$16/7 \approx 2.286$
G_t	7	6	$12/7 \approx 1.714$

Table 1: Average node degree for the four networks.

Observing on the 1 over, the average degree for G_c is 1.714, G_r is 2.0, G_d is 2.286 and G_t is 1.714.

Code Implementation

We used Python with the `networkx` library to construct the graphs G_c, G_r, G_d, G_t and calculate the average node degree for each. The following script was used:

```python
nodes = list("abcdefg")

Gc = nx.Graph()
Gc.add_nodes_from(nodes)
Gc.add_edges_from([("a","b"),("b","c"),("c","d"),("d","e"),("e","f"),("f","g")])

Gr = Gc.copy()
Gr.add_edge("a","g")

Gd = Gc.copy()
Gd.add_edges_from([("a","c"),("e","g")])

Gt = Gd.copy()
Gt.remove_edges_from([("a","b"),("f","g")])

graphs = {"Gc": Gc, "Gr": Gr, "Gd": Gd, "Gt": Gt}

def average_degree(G):
    return 2*G.number_of_edges()/G.number_of_nodes()

# Task (a)
avg_deg = {name: average_degree(G) for name, G in graphs.items()}
print("Average node degree:")
for name, val in avg_deg.items():
    print(f"{name}: {val:.3f}")
```

Listing 2: Average node degree

(b) Rankings by centrality

G_c (chain $a - b - c - d - e - f - g$).

- Degree: $\{b, c, d, e, f\} > \{a, g\}$ $(2 > 1)$.
- Betweenness: $d > \{c, e\} > \{b, f\} > \{a, g\}$.
- Closeness: $d > \{c, e\} > \{b, f\} > \{a, g\}$.

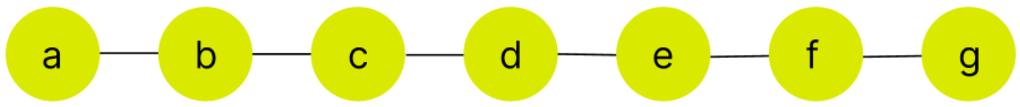

Figure 6: Illustrate G_c

In G_c, node d is the most central according to betweenness and closeness, acting as the main bridge in the chain. End nodes a and g are least central, while nodes c and e are secondary bridges.

G_r (ring, add $a - g$).

- Degree: all equal (2).
- Betweenness: all equal.
- Closeness: all equal.

Symmetric cycle $\Rightarrow$ all nodes have identical centralities.

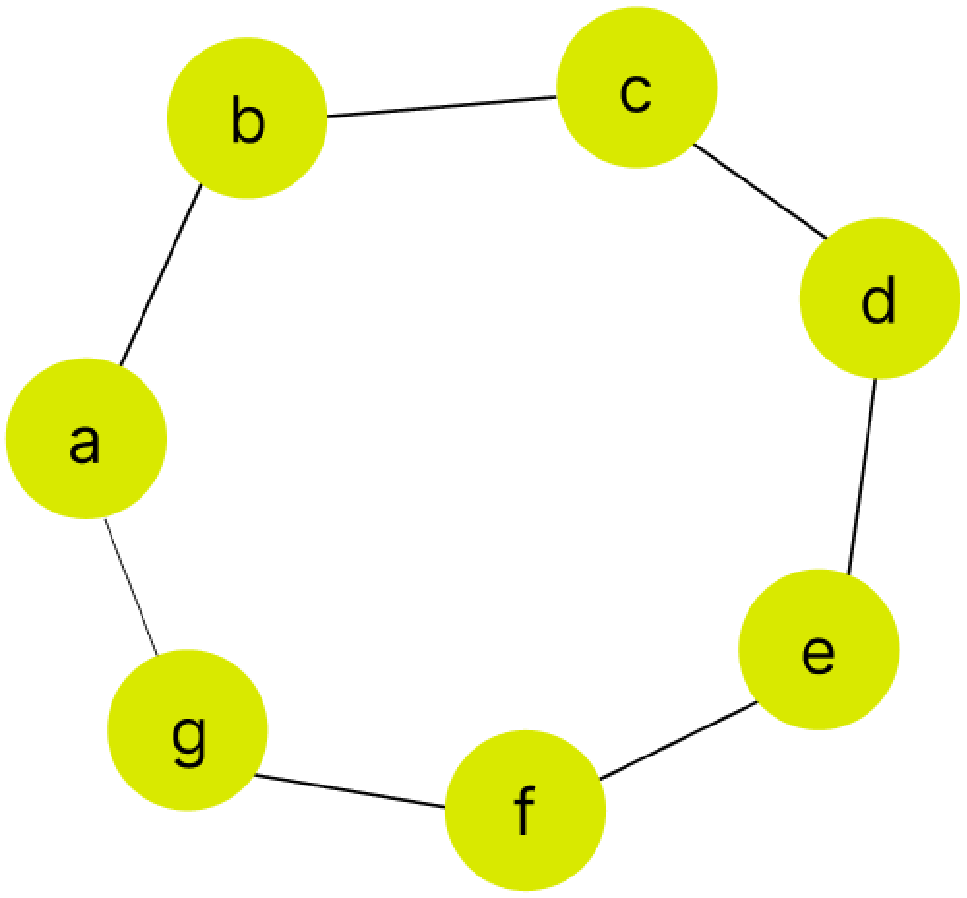

Figure 7: Illustrate: G_r

Since G_r is a symmetric cycle, all nodes are identical in terms of centrality. No single node is more important than another.

G_d (add (a, c) and (e, g)).

- Degree: $\{c, e\} > \{a, b, d, f, g\}$ $(3 > 2)$.

- Betweenness: $d > \{c, e\} > \{a, b, f, g\}$.
- Closeness: $d > \{c, e\} > \{b, f\} > \{a, g\}$.

Nodes c and e are gateways; node d acts as the main bridge.

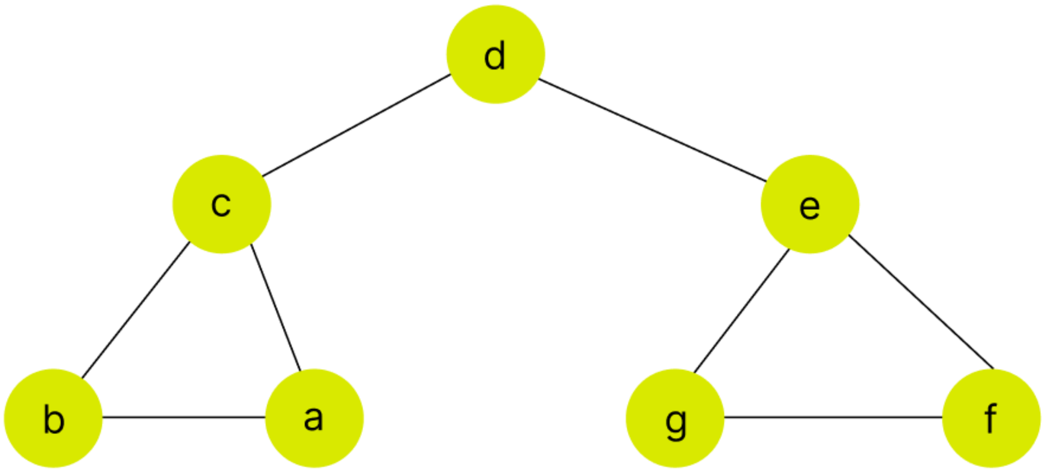

Figure 8: Illustrating G_d

Nodes c and e gain higher degree centrality as gateways, while d remains the main bridge with the highest betweenness and closeness.

G_t (from G_d, remove (a, b) and (f, g)).

- Degree: $\{c, e\} > d > \{a, b, f, g\}$ $(3 > 2 > 1)$.
- Betweenness: $d > \{c, e\} > \{a, b, f, g\}$.
- Closeness: $d > \{c, e\} > \{b, f\} > \{a, g\}$.

Removing the end links makes c and e more central; d remains the key bridge.

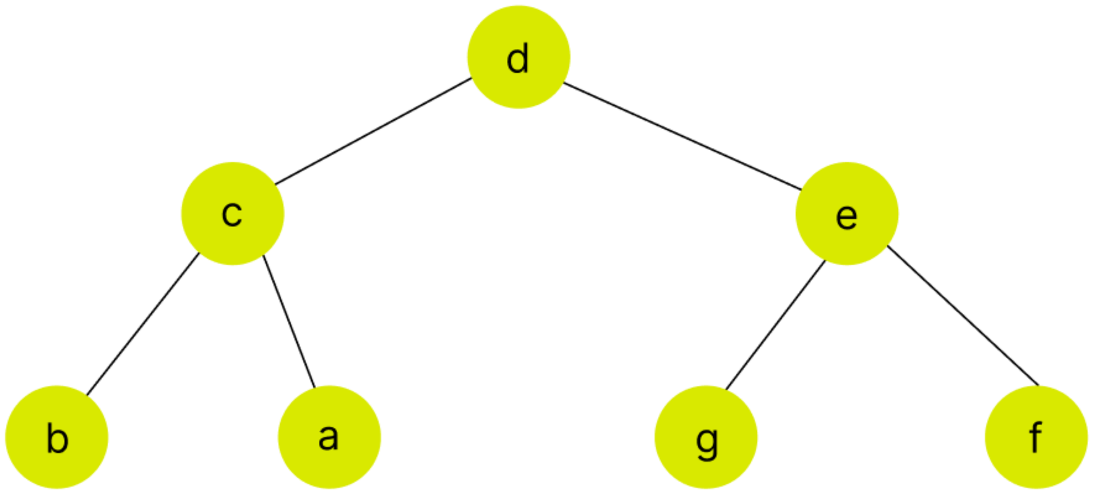

Figure 9: Illustrating G_t

Removing (a, b) and (f, g) increases the importance of c and e, but d continues to dominate as the critical connector.

Result and Comparison

Across networks, node d consistently achieves the highest centrality betweenness and closeness because it lies in the middle of the structure and connects the two halves of the graph. Many shortest paths must pass through d, making it the dominant bridge.

Comparison of measures:

- In G_c, degree centrality cannot fully distinguish nodes, but betweenness and closeness clearly identify d as most central.
- In G_r, the cycle structure makes all nodes equivalent in all measures.
- In G_d, degree highlights c and e as gateways, but betweenness and closeness still favor d.
- In G_t, removing links increases the importance of c and e, yet d remains the critical connector by global measures.

Thus, degree centrality reflects local connectivity, while betweenness and closeness emphasize global position in the network, consistently showing d as the key node for overall communication.

4 Task 1.4: Poisson Process – Bus Stop Scenario

(a) The number of passengers arriving in an interval of 15-minutes follows a Poisson distribution with mean

$$\mu = \lambda t = 5.$$

The resulting distribution is shown in Figure 10. From the plot, we see that the expected number of passengers is 5, while the most likely outcome (mode) is 4.

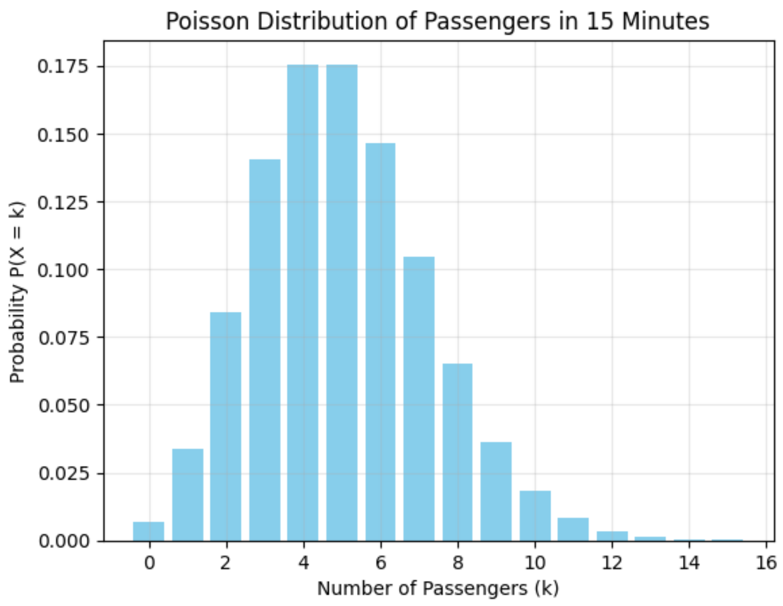

Figure 10: Poisson distribution of the number of passengers in 15 minutes.

(b) The inter-arrival times of passengers follow an exponential distribution with rate

$$\lambda = \frac{1}{3}.$$

The average inter-arrival time is therefore 3 minutes. The distribution is shown in Figure 11, which illustrates that short inter-arrival times are more probable, while longer waiting times become increasingly unlikely.

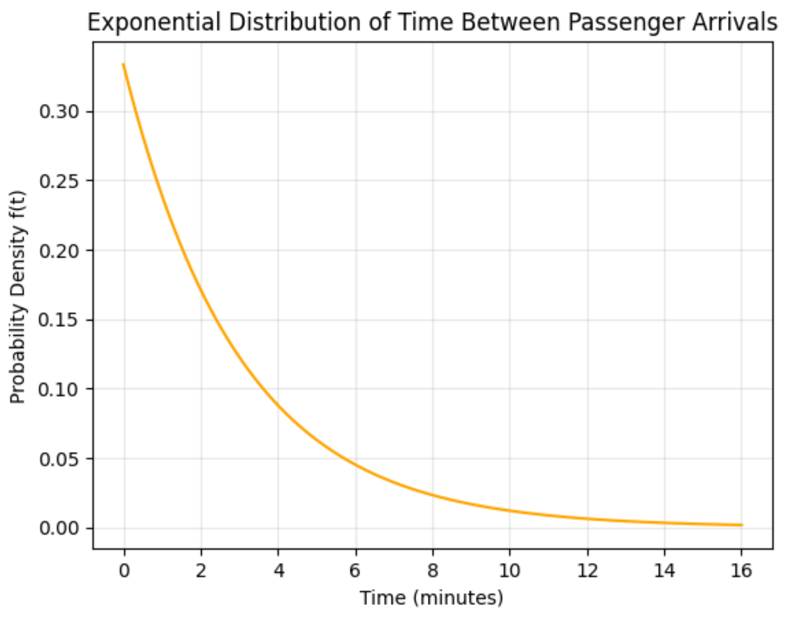

Figure 11: Exponential distribution of inter-arrival times between passengers.

4.0.1 Code Implementation

Under is the implementet code for plotting and calculating the computation needed to solve the task:

```python
# Task a
t = 15
lamb = 1/3 # Poisson process intensity (1 passenger arrives every 3 minutes)
mu = lamb*t # The expected number of passengers in 15 minutes
print(f"The expected number of passengers in {t} minutes is {int(mu)}")
k_values = np.arange(0, 16) # k values from 0 to 15
probabilities = []


def poisson_prob(mu, k): #Function to calculate poisson probability
    return (np.exp(-mu)*mu**k)/(m.factorial(k))

for k in k_values:
    prob = poisson_prob(mu, k) # Calculate the probability for each k nhbhh
    probabilities.append(prob) # Store the probability

plt.bar(k_values, probabilities, color='skyblue')
plt.xlabel('Number of Passengers (k)')
plt.ylabel('Probability P(X = k)')
plt.title('Poisson Distribution of Passengers in 15 Minutes')
plt.grid(True, alpha=0.3)
plt.show()

print(f"Most likely number of passengers: {k_values[np.argmax(probabilities)]}")

# Task b
def time_distribution(lamb, t): #Function to calculate the PDF of exponential
    distribution
    return lamb * np.exp(-lamb * t) # PDF formula

t_values = np.linspace(0, 16, 100) # Time values from 0 to 16 minutes
pdf_values = time_distribution(lamb, t_values) # Calculate the PDF values

plt.plot(t_values, pdf_values, color='orange') # Plot the PDF (Probability Density
    Function)
plt.xlabel('Time (minutes)')
plt.ylabel('Probability Density f(t)')
plt.title('Exponential Distribution of Time Between Passenger Arrivals')
plt.grid(True, alpha=0.3)
plt.show()

mean_time = 1/lamb # Mean time between arrivals
print(f"The average inter-arrival time is {mean_time} minutes")
```

Listing 3: Possion process

5 Task 1.5: Stochastic Process – Multiple Access

(a) If the arrival process is a Poisson process with rate λ, then the number of packets per time slot of duration h follows a Poisson distribution with mean λh. The probability mass function is

$$P(X = k) = e^{-\lambda h}\frac{(\lambda h)^k}{k!}, \quad k = 0, 1, 2, \dots$$

(b) A correct transmission occurs if exactly one packet is transmitted in the slot, i.e. $P(X = 1)$:

$$P(X = 1) = e^{-\lambda h} \cdot (\lambda h).$$

(c) To maximize the probability of correct transmission, we differentiate with respect to $\mu = \lambda h$:

$$\frac{d}{d\mu}P(\mu) = e^{-\mu}(1 - \mu).$$

Setting this equal to zero yields $\mu = 1$, i.e. $\lambda h = 1$. Substituting back gives

$$P(X = 1) = e^{-1} \cdot 1 \approx 0.368.$$

Thus, the channel achieves a maximum utilization of approximately 36.8%.

Code

```
mu = 1
P_1 = (np.exp(-mu) * (mu)**1) / m.factorial(1)
print(f"P(Correct): {P_1:.3f}")
```

Listing 4: Stochastic Process

6 Task 1.6: Little's Law in the Department

(a) Using Little's Law, the average number of students in each study program is given by

$$L = \lambda \cdot W,$$

where λ is the average arrival rate (students per year) and W is the average time spent (years). The results are:

$$
\begin{aligned}
L_{\text{BDIGSEC}} &= 45 \cdot 3.5 = 157.5, \\
L_{\text{MTKOM}} &= 72 \cdot 6 = 432, \\
L_{\text{MDIGSEC}} &= 28 \cdot 2.5 = 70.0, \\
L_{\text{MIS}} &= 23 \cdot 2.5 = 57.5, \\
L_{\text{PhD}} &= 5 \cdot 4 = 20.
\end{aligned}
$$

The total average number of students at the department is therefore

$$L_{\text{total}} = 737.$$

(b) The average number of students arriving each year is the sum of intake rates:

$$\lambda_{\text{total}} = 45 + 72 + 28 + 23 + 5 = 173.$$

(c) The average time a student spends at the department is

$$\hat{W} = \frac{L_{\text{total}}}{\lambda_{\text{total}}} \approx 4.26 \text{ years}.$$

Code Illustration

Used python to compute the calculation.

```

a)
l_bdigsec = 45*3.5
l_mtkom = 72*6
l_mdigsec = 28*2.5
l_mis = 23*2.5
l_phd = 5*4
print(f"a) \nThe average number of students for each study program is: \n BDIGSEC =
     {l_bdigsec} \n MTKOM = {l_mtkom} \n MDIGSEC = {l_mdigsec} \n MIS = {l_mis} \n
    PhD = {l_phd} \n")

l_sum = l_bdigsec + l_mtkom + l_mdigsec + l_mis + l_phd
print(f"The total number of average students at the department is {l_sum}")

b)
lam_sum = 45 + 72 + 28 + 23 + 5
print(f"b) \nAverage number of students arriving each year is {lam_sum}")

c)
w_hat = l_sum / lam_sum
print(f"c) \nAverage time spent by a student is {w_hat}")
```

Listing 5: Stochastic Process