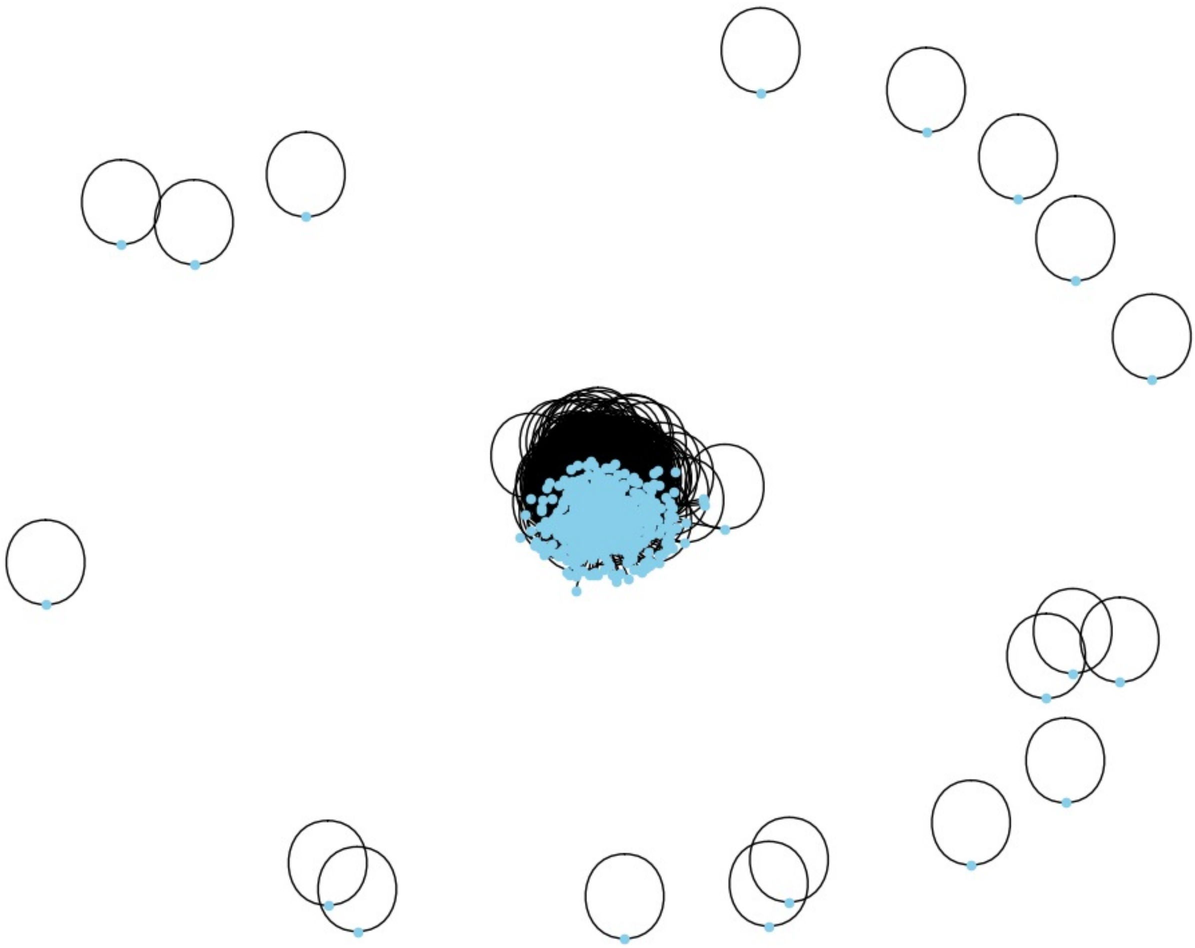
# 11508 Muzamil Khan SNA Phase 4

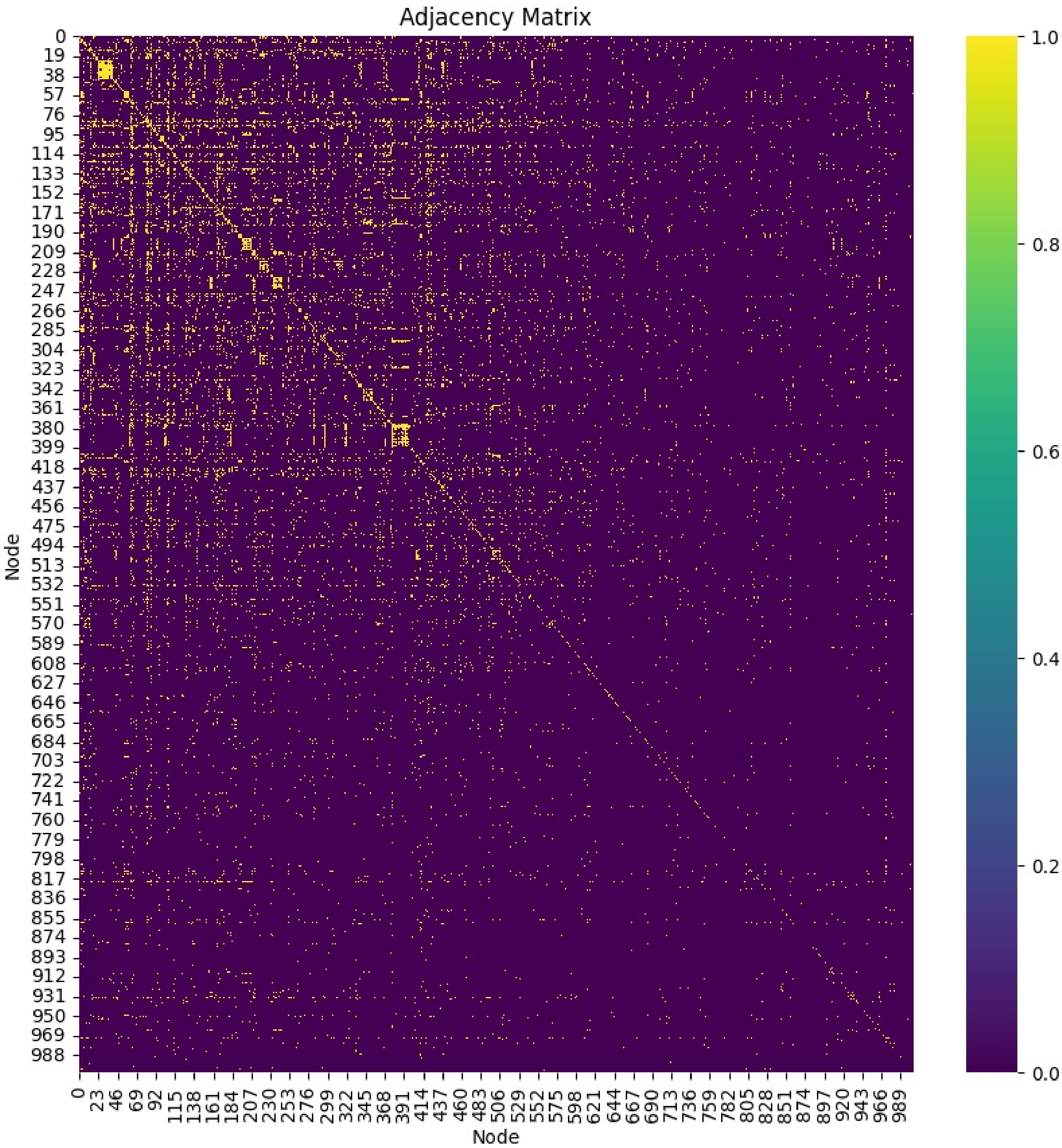
# email-Eu-core Dataset

In [6]:

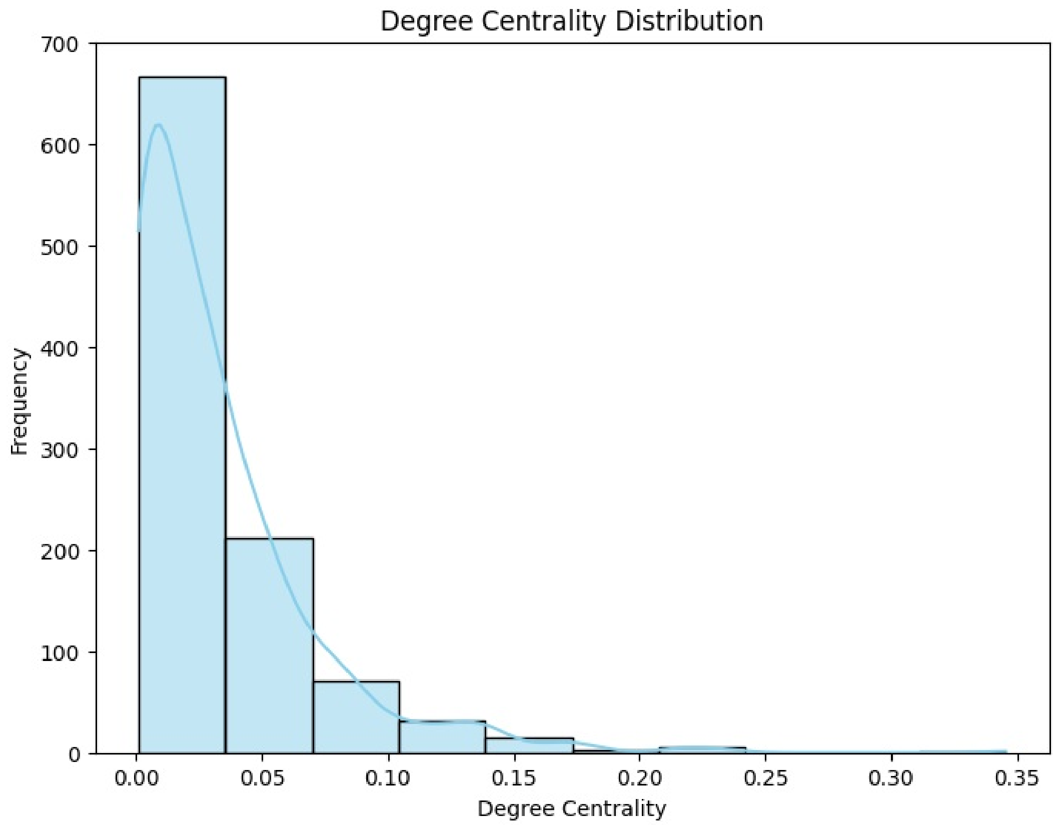
```
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Data Collection
# Load the edge list from the "email-Eu-core.txt" file
data = pd.read_csv("email-Eu-core.txt", sep=' ', header=None, names=['Source', 'Target'])

# Step 2: Data Preparation
# In this case, the data is already in the format of an edge list

# Step 3: Network Construction
# Create a network object and add edges
G = nx.Graph()
G.add_edges_from(data.values)

# Step 4: Visualization
# Visualize the network
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(G)  # Using spring layout for visualization
nx.draw(G, pos, with_labels=False, node_size=20, node_color='skyblue', font_size=8)
plt.title('Network Visualization')
plt.show()

# Sociogram (Adjacency Matrix)
adj_matrix = nx.adjacency_matrix(G).todense()
plt.figure(figsize=(10, 10))
sns.heatmap(adj_matrix, cmap='viridis')
plt.title('Adjacency Matrix')
plt.xlabel('Node')
plt.ylabel('Node')
plt.show()

# Mathematical Formulation
# Compute degree centrality
degree_centrality = nx.degree_centrality(G)
# Compute other centrality measures as needed
closeness_centrality = nx.closeness_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)

# Convert centrality measures to DataFrame
centrality_measures_df = pd.DataFrame({
    'Node': list(degree_centrality.keys()),
    'Degree Centrality': list(degree_centrality.values()),
    'Closeness Centrality': list(closeness_centrality.values()),
    'Betweenness Centrality': list(betweenness_centrality.values())
})

# Interpretation: Print centrality measures for all nodes
print("Centrality Measures for All Nodes:")
print(centrality_measures_df)

# Additional Analysis and Visualization
# Plot degree centrality distribution for all nodes
plt.figure(figsize=(8, 6))
sns.histplot(centrality_measures_df['Degree Centrality'], kde=True, bins=10, color='skyblue')
plt.title('Degree Centrality Distribution')
plt.xlabel('Degree Centrality')
plt.ylabel('Frequency')
plt.show()

# Design Network (display various centralities)
# Melt dataframe for seaborn
centrality_measures_melted = centrality_measures_df.melt(id_vars=['Node'], var_name='Centrality Measure', value_

plt.figure(figsize=(14, 8))
sns.barplot(x='Value', y='Node', hue='Centrality Measure', data=centrality_measures_melted, palette='viridis')
plt.title('Centrality Measures for All Nodes')
plt.xlabel('Centrality Value')
plt.ylabel('Node')
plt.legend(title='Centrality Measure')
plt.show()
```

Network Visualization

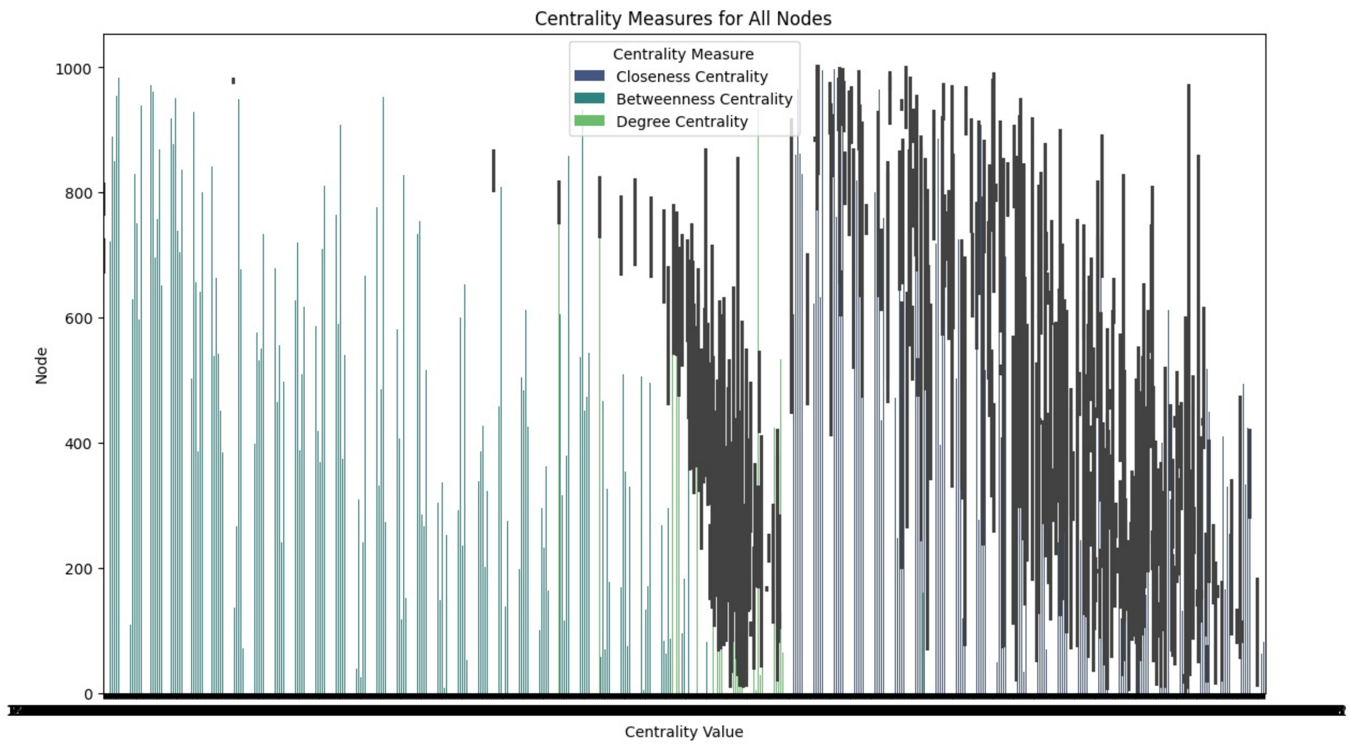





```
Centrality Measures for All Nodes:
      Node  Degree Centrality  Closeness Centrality  Betweenness Centrality
0        0           0.043825              0.421991                0.001124
1        1           0.051793              0.422360                0.001195
2        2           0.094622              0.461490                0.006570
3        3           0.070717              0.441663                0.001654
4        4           0.095618              0.462152                0.005547
...    ...                ...                   ...                     ...
1000  1000           0.005976              0.355934                0.000004
1001  1001           0.009960              0.339789                0.000004
1002  1002           0.000996              0.297983                0.000000
1003  1003           0.000996              0.298167                0.000000
1004  1004           0.000996              0.295975                0.000000

[1005 rows x 4 columns]
```



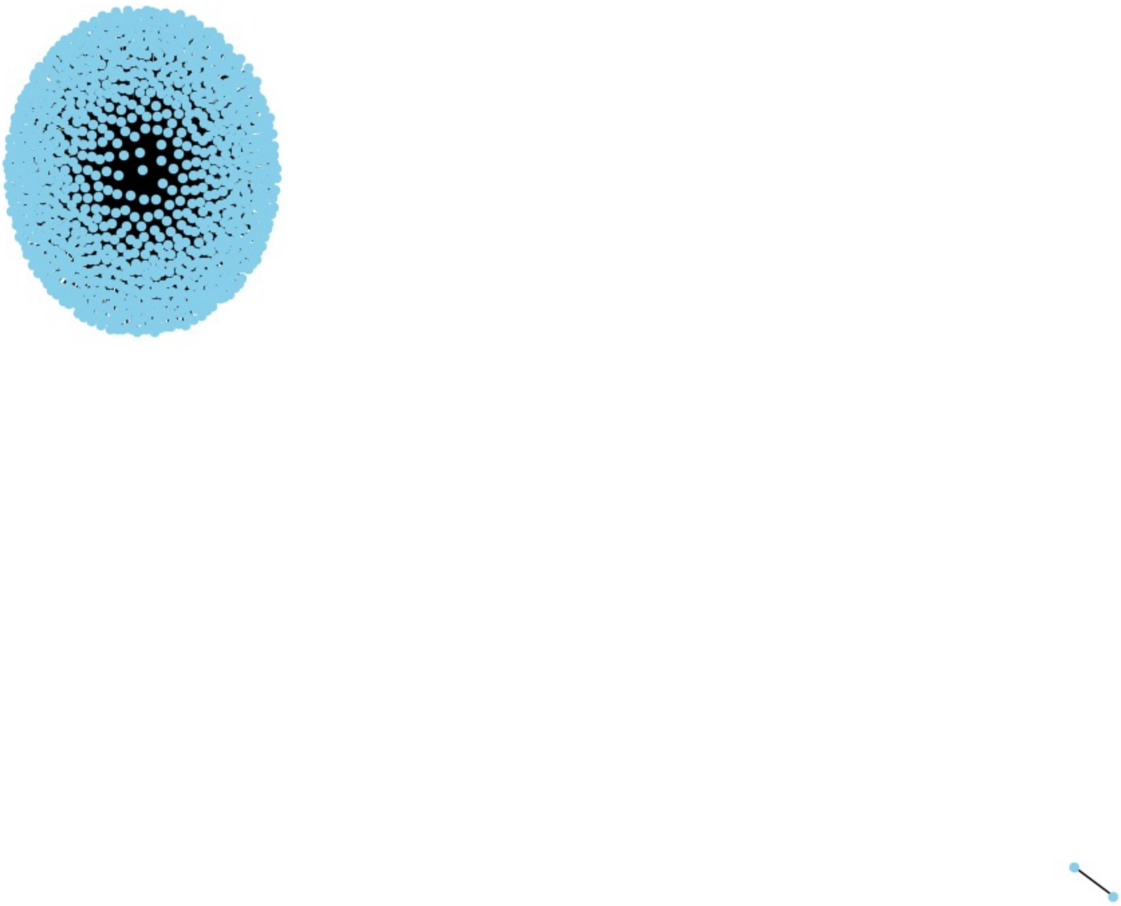

```
C:\Python312\Lib\site-packages\IPython\core\pylabtools.py:170: UserWarning: Creating legend with loc="best" can
be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```

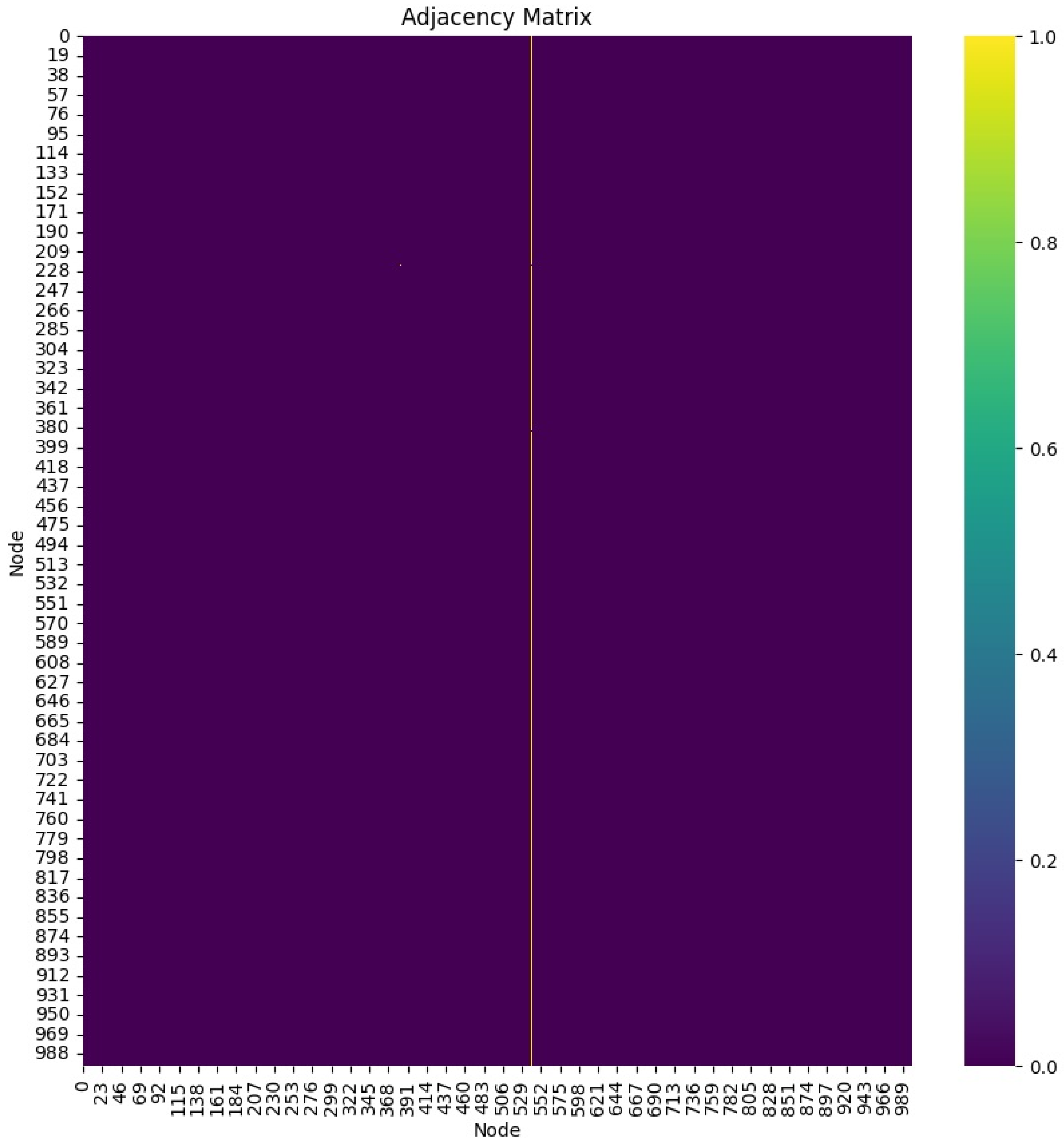


# p2p-Gnutella08 Dataset

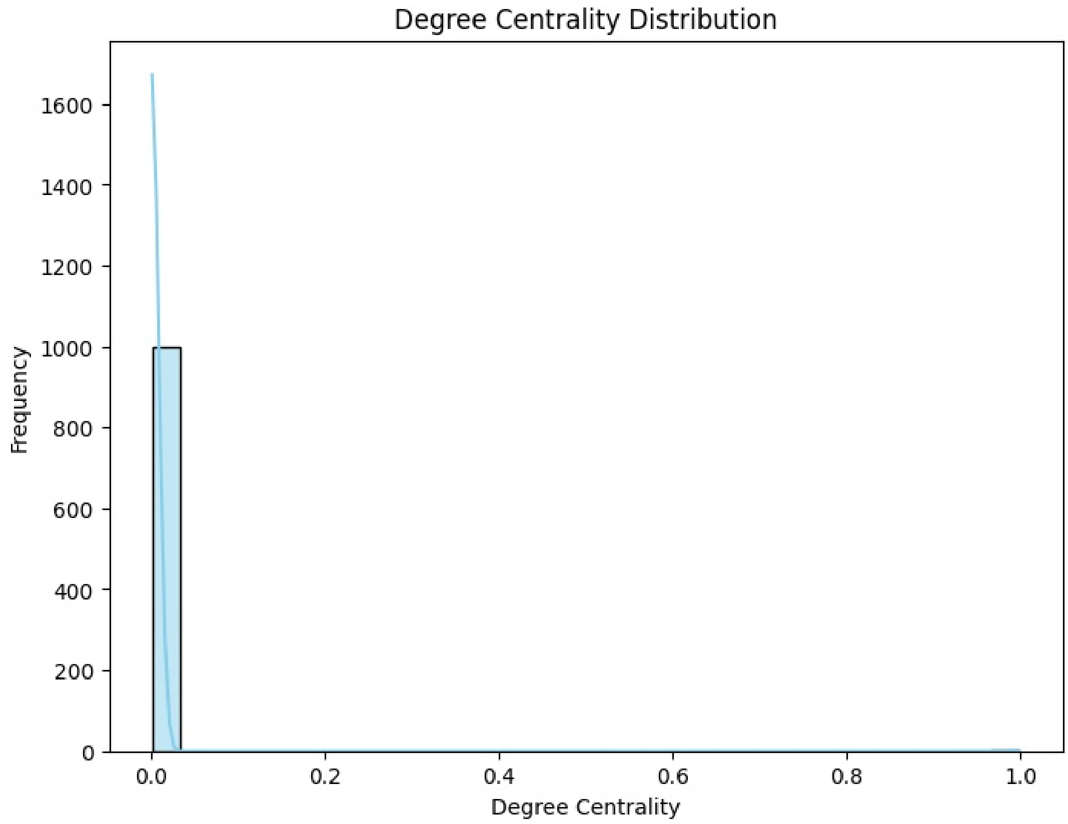
In [19]:
```
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Data Collection
# Load the edge list from the "p2p-Gnutella08.txt" file
data = pd.read_csv("p2p-Gnutella08.txt", sep=' ', header=None, names=['Source', 'Target'])

# Step 2: Data Preparation
# In this case, the data is already in the format of an edge list

# Step 3: Network Construction
# Create a network object and add edges
G = nx.Graph()
G.add_edges_from(data.values)

# Reduce the number of nodes and edges for faster processing
G = G.subgraph(list(G.nodes)[:1000])  # Use the first 1000 nodes for visualization

# Step 4: Visualization
# Visualize the network
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=False, node_size=20, node_color='skyblue', font_size=8)
plt.title('Network Visualization')
plt.show()

# Sociogram (Adjacency Matrix)
adj_matrix = nx.adjacency_matrix(G).todense()
plt.figure(figsize=(10, 10))
sns.heatmap(adj_matrix, cmap='viridis')
plt.title('Adjacency Matrix')
plt.xlabel('Node')
plt.ylabel('Node')
plt.show()

# Compute and visualize centrality measures
centrality_measures = {
    "Node": list(G.nodes())
}

# Add centrality measures for each centrality metric
centrality_functions = {
    "Degree Centrality": nx.degree_centrality,
    "Closeness Centrality": nx.closeness_centrality,
    "Betweenness Centrality": nx.betweenness_centrality
}

for centrality_name, centrality_func in centrality_functions.items():
    centrality_values = centrality_func(G)
    centrality_measures[centrality_name] = list(centrality_values.values())

# Convert centrality measures to DataFrame
centrality_df = pd.DataFrame(centrality_measures)

# Print centrality measures for the first 1000 nodes
print("Centrality Measures for the First 1000 Nodes:")
print(centrality_df.head(1000))

# Plot degree centrality distribution
plt.figure(figsize=(8, 6))
sns.histplot(centrality_df["Degree Centrality"], kde=True, bins=30, color='skyblue')
plt.title('Degree Centrality Distribution')
plt.xlabel('Degree Centrality')
plt.ylabel('Frequency')
plt.show()

# Plot centrality measures for the first 1000 nodes
plt.figure(figsize=(12, 8))
sns.barplot(data=centrality_df.head(1000), palette='viridis')
plt.title('Centrality Measures for the First 1000 Nodes')
plt.xlabel('Centrality Measure')
plt.ylabel('Centrality Value')
plt.show()
```

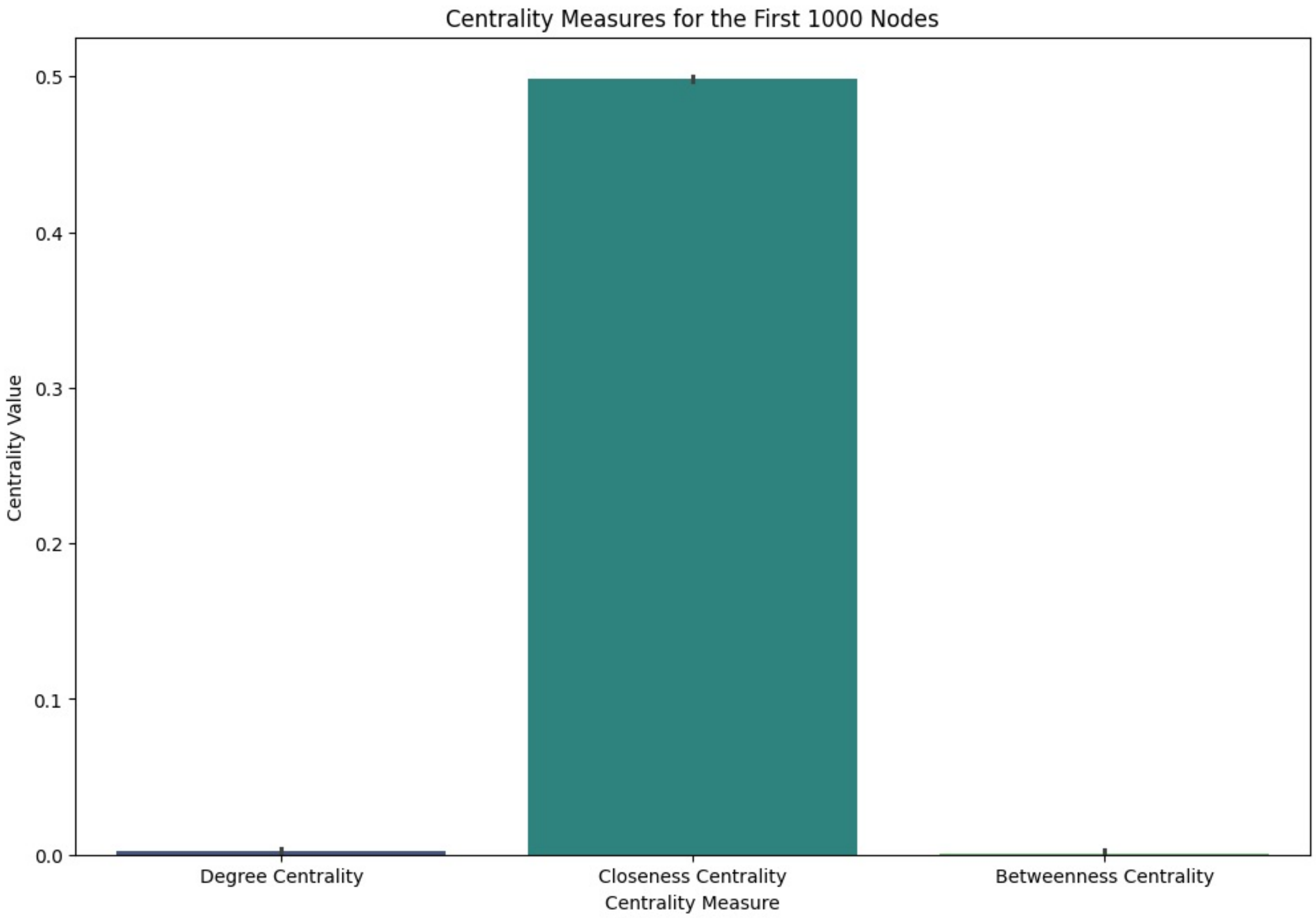
Network Visualization






```
Centrality Measures for the First 1000 Nodes:
          Node  Degree Centrality  Closeness Centrality  \
0    160\t2110           0.001001              0.499249
1       22\t28           0.001001              0.499249
2       44\t53           0.001001              0.499249
3     39\t1922           0.001001              0.499249
4      38\t725           0.001001              0.499249
..         ...                ...                   ...
995      30\t3           0.001001              0.499249
996  191\t2371           0.001001              0.499249
997   147\t177           0.001001              0.499249
998  138\t1752           0.001001              0.499249
999    4\t1903           0.001001              0.499249

     Betweenness Centrality
0                       0.0
1                       0.0
2                       0.0
3                       0.0
4                       0.0
..                      ...
995                     0.0
996                     0.0
997                     0.0
998                     0.0
999                     0.0

[1000 rows x 4 columns]
```



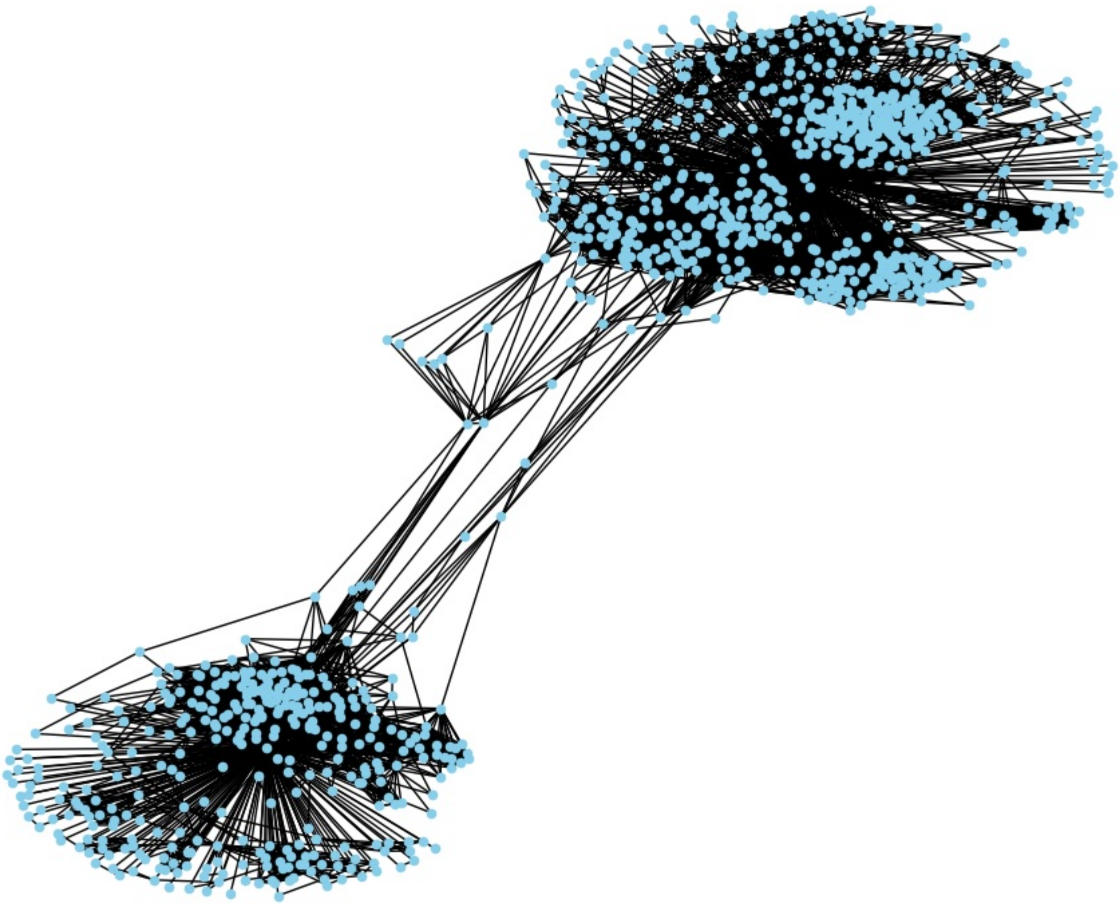



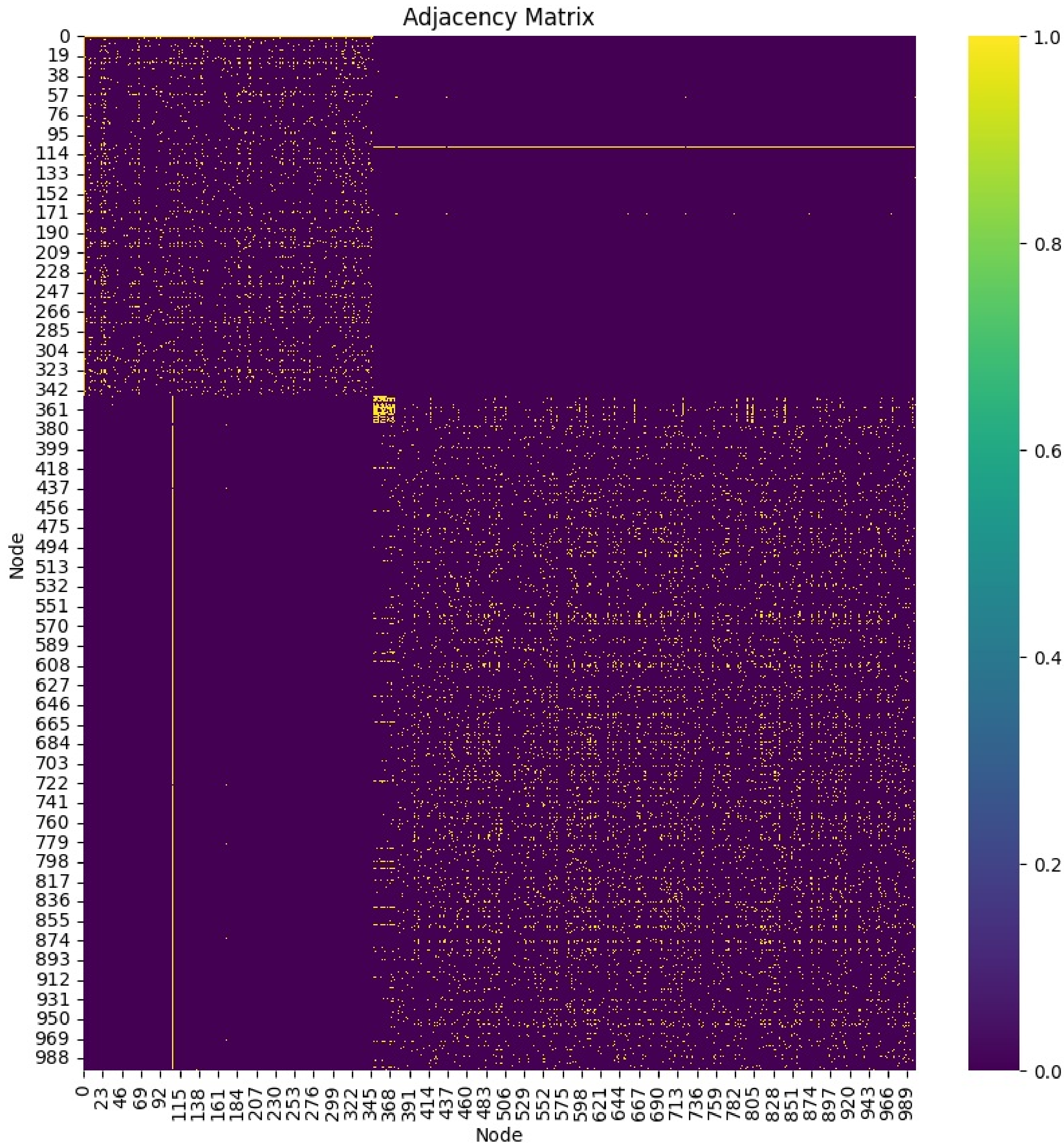

# facebook_combined

In [26]:
```
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Data Collection
# Load the edge list from the "facebook_combined.txt" file
data = pd.read_csv("facebook_combined.txt", sep=' ', header=None, names=['Source', 'Target'])
```

```python
# Step 2: Data Preparation
# In this case, the data is already in the format of an edge list

# Step 3: Network Construction
# Create a network object and add edges
G = nx.Graph()
G.add_edges_from(data.values)

# Reduce the number of nodes and edges for faster processing
G = G.subgraph(list(G.nodes)[:1000])  # Use the first 1000 nodes for visualization

# Step 4: Visualization
# Visualize the network
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(G)  # Using spring layout for visualization
nx.draw(G, pos, with_labels=False, node_size=20, node_color='skyblue', font_size=8)
plt.title('Network Visualization')
plt.show()

# Sociogram (Adjacency Matrix)
adj_matrix = nx.adjacency_matrix(G).todense()
plt.figure(figsize=(10, 10))
sns.heatmap(adj_matrix, cmap='viridis')
plt.title('Adjacency Matrix')
plt.xlabel('Node')
plt.ylabel('Node')
plt.show()

# Compute and visualize centrality measures
centrality_measures = {
    "Degree Centrality": nx.degree_centrality(G),
    "Closeness Centrality": nx.closeness_centrality(G),
    "Betweenness Centrality": nx.betweenness_centrality(G)
}

# Convert centrality measures to DataFrame
centrality_df = pd.DataFrame(centrality_measures)

# Print centrality measures for the first 10 nodes
print("Centrality Measures for the First 10 Nodes:")
print(centrality_df.head(10))

# Plot degree centrality distribution
plt.figure(figsize=(8, 6))
sns.histplot(centrality_df["Degree Centrality"], kde=True, bins=10, color='skyblue')
plt.title('Degree Centrality Distribution')
plt.xlabel('Degree Centrality')
plt.ylabel('Frequency')
plt.show()

# Plot centrality measures for the first 10 nodes
plt.figure(figsize=(12, 8))
sns.barplot(data=centrality_df.head(10), palette='viridis')
plt.title('Centrality Measures for the First 10 Nodes')
plt.xlabel('Centrality Measure')
plt.ylabel('Centrality Value')
plt.show()
```

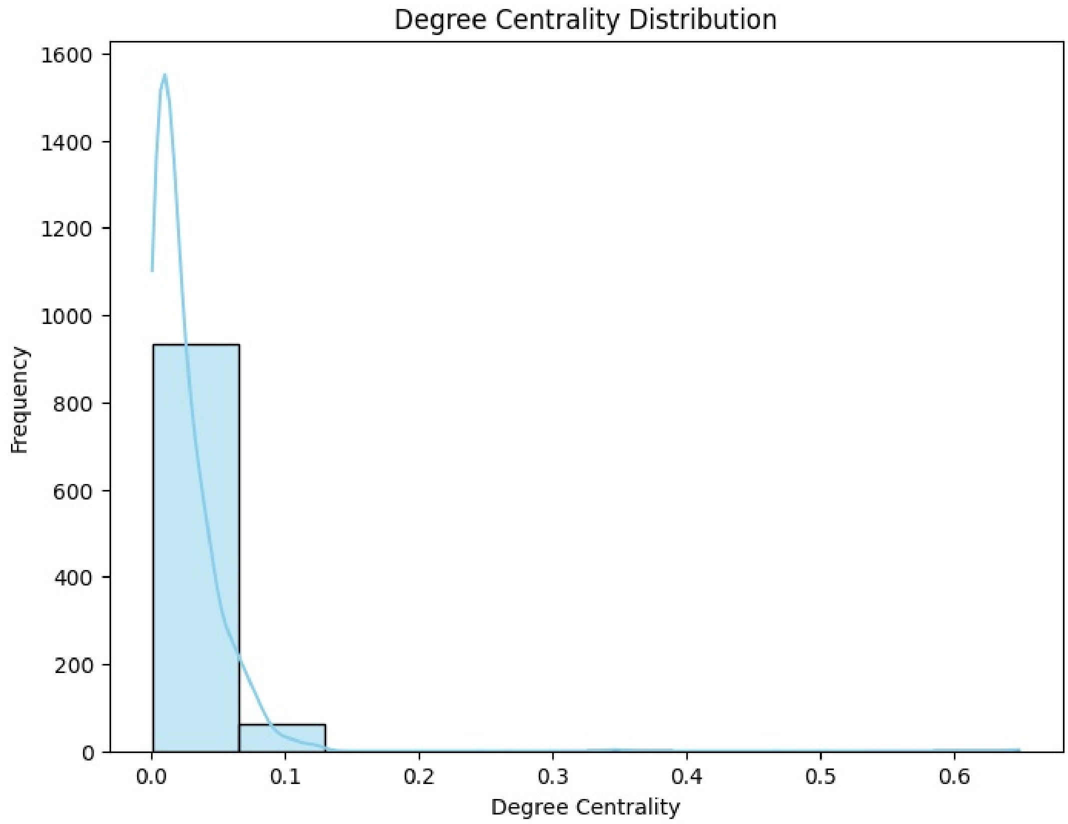
Network Visualization

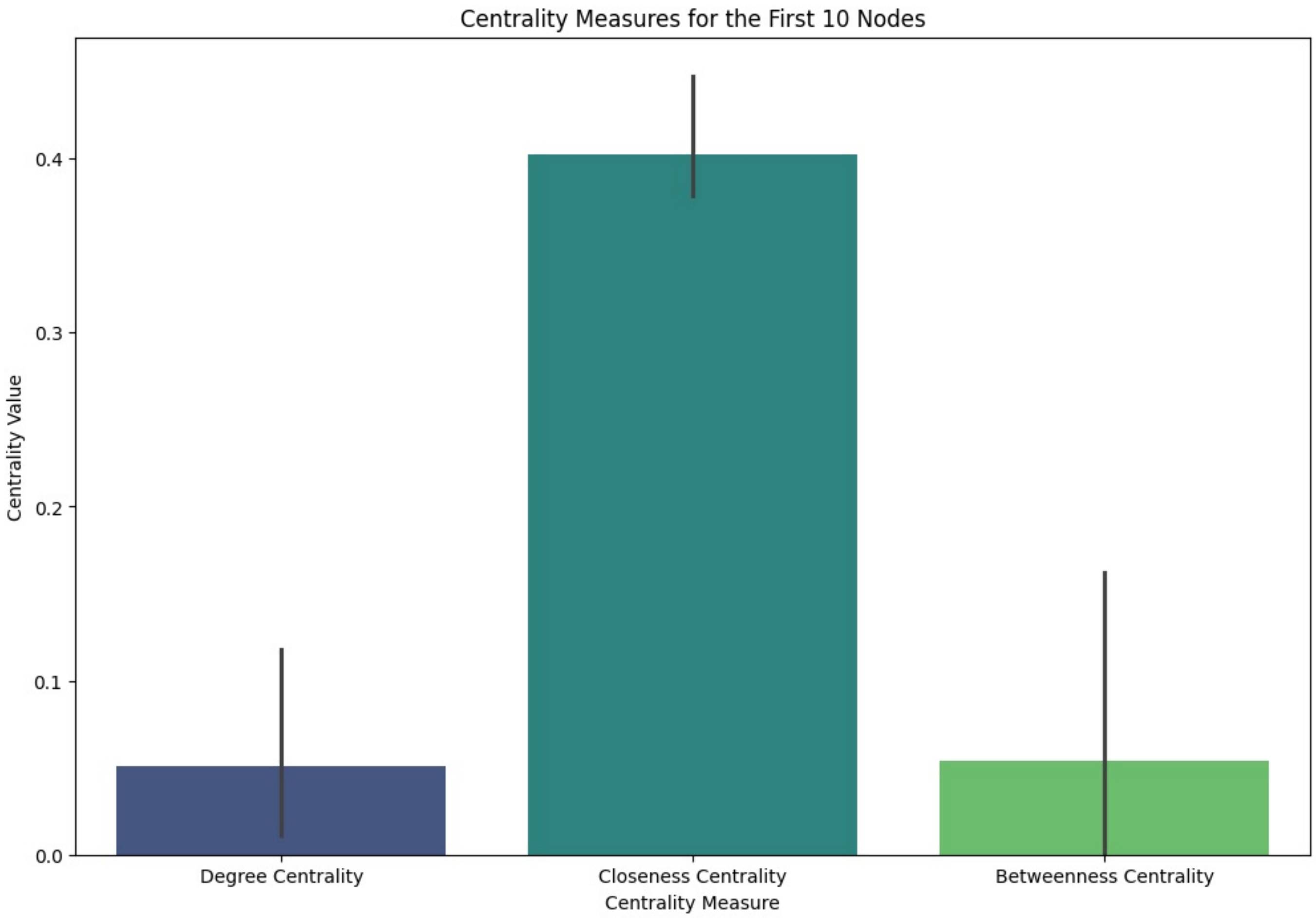





Centrality Measures for the First 10 Nodes:

| | Degree Centrality | Closeness Centrality | Betweenness Centrality |
|---|---|---|---|
| 0 | 0.347347 | 0.605088 | 5.413976e-01 |
| 1 | 0.017017 | 0.379415 | 4.550772e-05 |
| 2 | 0.010010 | 0.378409 | 1.241818e-06 |
| 3 | 0.017017 | 0.379415 | 2.755154e-05 |
| 4 | 0.010010 | 0.378409 | 3.009021e-06 |
| 5 | 0.013013 | 0.378840 | 3.606845e-05 |
| 6 | 0.006006 | 0.377837 | 4.012028e-07 |
| 7 | 0.020020 | 0.379992 | 5.191050e-05 |
| 8 | 0.008008 | 0.378123 | 4.513532e-06 |
| 9 | 0.057057 | 0.385268 | 2.690337e-04 |




The End 11508 Muzamil Khan

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js