Overview:

Random network

sequence = nx.random_powerlaw_tree_sequence(100, tries=5000)

G = nx.configuration_model(sequence)

Assignment objective:

- show empirically how close to expectation the real network generated from the White Helmets dataset is compared to a baseline model
- compare the characteristics of the network extracted from the White Helmets dataset described in the first lecture with the expected characteristics of a “regular” network generated using some classical network generation models.
- Choose and justify your choice for at least 5 network metrics
- Choose at least two generative network models to compare against. Justify your choice.
- Via empirical evaluation, make the case that the network structures from the White Helmets dataset are expected/unexpected (with an attempt to distinguish between organic and coordinated behaviour)

Dataset

- The dataset has the following format: , <userID_1>, <userID_2>, <timestamp_1>, <timestamp_2>
- where the posts by userID_1 and userID_2 are within 52 seconds of each others and both contain a link to the same videoID.
- timestamp_1 is the time when userID_1 posted the url to videoID, timestamp_2 is the time when userID_2 posted the same url.

Metrics:

Chosen metrics and reasons for choosing theses metrics:

- Clustering Coefficient → Measures how likely it is that two neighbors of a node are also connected. Rationale: Coordinated campaigns often form tightly connected groups that amplify similar narratives. Random networks usually have low clustering, so higher values indicate non-random coordination.

- Modularity → Quantifies the presence of distinct communities within the network. Rationale: Influence operations typically involve groups of accounts promoting similar messages. High modularity compared to a random baseline suggests organized, topic-based coordination.

- Degree Assortativity → Captures whether nodes of similar degree tend to connect with each other. Rationale: Coordinated networks often display unusual degree correlations (e.g., central "hub" accounts linked to many smaller ones). Deviations from randomness reveal hierarchical or controlled connectivity.

- Betweenness Centrality → Measures how frequently a node appears on the shortest paths between others. Rationale: Cross-platform actors who bridge YouTube, Twitter, and Facebook tend to have high betweenness. This highlights key agents coordinating dissemination across platforms.

- Degree Distribution → Describes how many links each node has. Rationale: Random networks tend to have homogeneous degree patterns. Real disinformation networks often exhibit heavy-tailed distributions with dominant hubs — evidence of intentional amplification.

Networks:

Reason for comparing real networks with random networks:

Comparing two complete different types of random/baseline networks with the real network to see which structures and properties in the real network are not random.

- Which properties of the real network are statistically significant and which properties are occurring also in random networks
- If a real network, for example, has a much higher clustering than a random network of the same density → indication of community structures or local organization.
- Evaluating the benefits of a real network in comparison to a random network

Chosen random networks:

- Watts–Strogatz Graph (Small-World Network) This model is useful because it combines local clustering with short average path lengths, resembling many real-world social systems. Comparing the real network to a WS-graph helps determine whether the observed small-world effects are naturally emergent or the result of coordinated behavior.

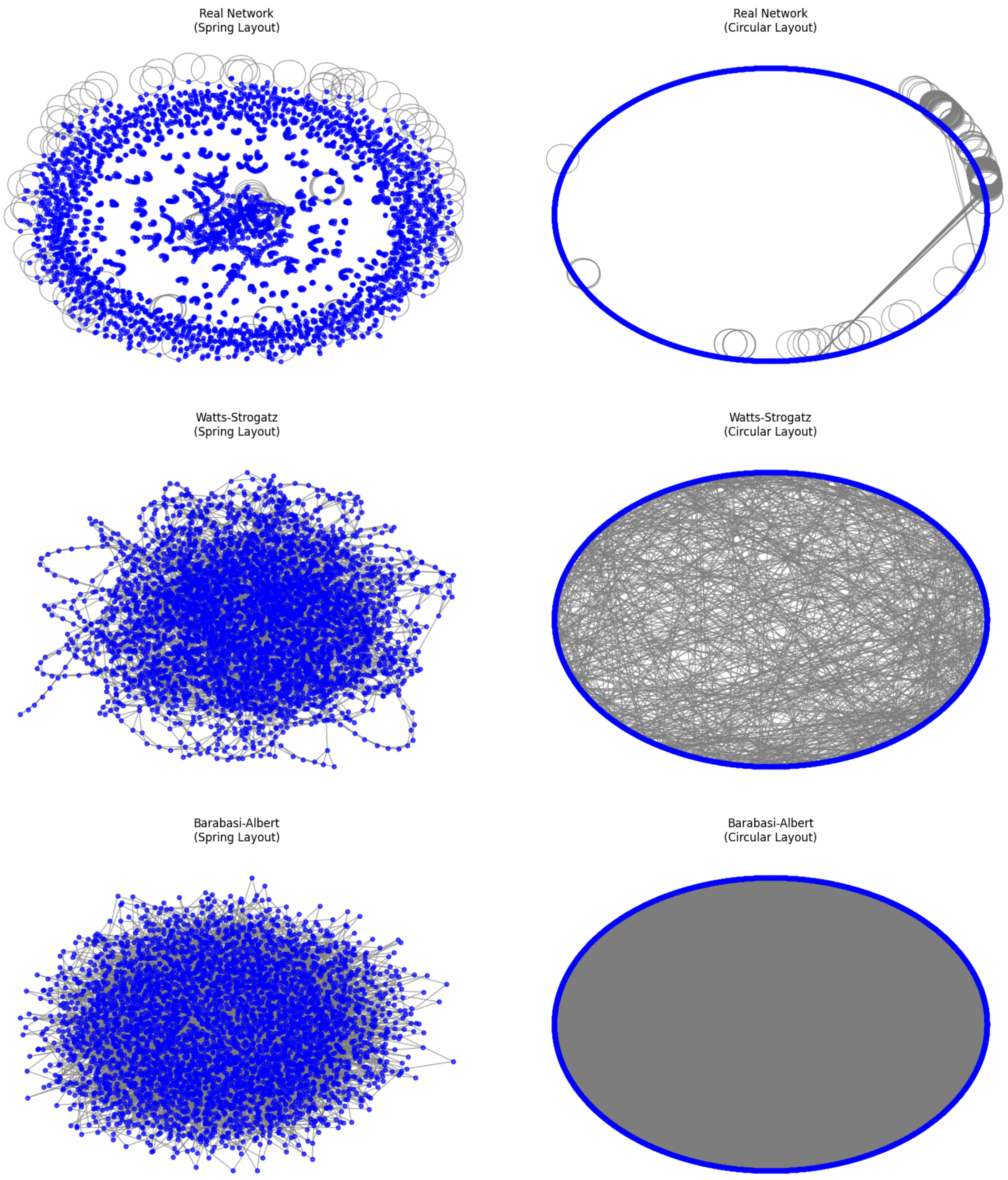
- Barabási–Albert Model (Scale-Free Network) This model generates networks with hub nodes that follow a power-law degree distribution. Comparing the real network with a BA-model helps to identify whether the presence of highly connected hubs is a natural outcome of preferential attachment or evidence of intentional coordination.

In [2]:
```python
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from collections import Counter
import community as community_louvain



def calculate_clustering_coefficient(G):
    return nx.average_clustering(G)

def calculate_modularity(G, partition):
    return community_louvain.modularity(partition, G)

def calculate_degree_assortativity(G):
    return nx.degree_assortativity_coefficient(G)

def calculate_betweenness_centrality(G):
    return nx.betweenness_centrality(G)

def calculate_degree_distribution(G):
    degree_sequence = [d for n, d in G.degree()]
    return Counter(degree_sequence)

# Analysis function to compute all metrics
def compute_network_metrics(G, network_name):
    """Compute all metrics for a network without plotting"""
    print(f"Computing metrics for {network_name}...")

    metrics = {}

    # Clustering Coefficient
    metrics['clustering_values'] = list(nx.clustering(G).values())
    metrics['avg_clustering'] = calculate_clustering_coefficient(G)

    # Modularity
    partition = community_louvain.best_partition(G)
    metrics['modularity'] = calculate_modularity(G, partition)

    # Degree Assortativity
    metrics['degree_assortativity'] = calculate_degree_assortativity(G)

    # Betweenness Centrality
    betweenness_dict = calculate_betweenness_centrality(G)
    metrics['betweenness_values'] = list(betweenness_dict.values())
    metrics['avg_betweenness'] = np.mean(metrics['betweenness_values'])

    # Degree Distribution
    metrics['degree_count'] = calculate_degree_distribution(G)

    return metrics

# Plotting functions for side-by-side comparison
def plot_clustering_comparison(metrics_dict):
    """Plot clustering coefficient distributions side by side"""
    fig, axes = plt.subplots(1, 3, figsize=(18, 5))

    for idx, (network_name, metrics) in enumerate(metrics_dict.items()):
        sns.histplot(metrics['clustering_values'], bins=30, kde=True, ax=axes[id
        axes[idx].set_title(f"{network_name}\nAvg: {metrics['avg_clustering']:.4
        axes[idx].set_xlabel('Clustering Coefficient')
        axes[idx].set_ylabel('Frequency')

    plt.tight_layout()
    plt.show()

def plot_modularity_comparison(metrics_dict):
    """Plot modularity comparison as bar chart"""
    fig, ax = plt.subplots(figsize=(10, 6))

    networks = list(metrics_dict.keys())
    modularity_values = [metrics['modularity'] for metrics in metrics_dict.value

    bars = ax.bar(networks, modularity_values, color=['#1f77b4', '#ff7f0e', '#2c
    ax.set_ylabel('Modularity')
    ax.set_title('Modularity Comparison')
    ax.set_ylim(0, max(modularity_values) * 1.2)

    # Add value labels on bars
    for bar, value in zip(bars, modularity_values):
        height = bar.get_height()
        ax.text(bar.get_x() + bar.get_width()/2., height,
                f'{value:.4f}',
                ha='center', va='bottom')

    plt.tight_layout()
    plt.show()

def plot_degree_assortativity_comparison(metrics_dict):
    """Plot degree assortativity comparison as bar chart"""
    fig, ax = plt.subplots(figsize=(10, 6))

    networks = list(metrics_dict.keys())
    assortativity_values = [metrics['degree_assortativity'] for metrics in metri

    bars = ax.bar(networks, assortativity_values, color=['#1f77b4', '#ff7f0e', '
    ax.set_ylabel('Degree Assortativity')
    ax.set_title('Degree Assortativity Comparison')
    ax.axhline(y=0, color='black', linestyle='--', linewidth=0.8, alpha=0.5)

    # Add value labels on bars
    for bar, value in zip(bars, assortativity_values):
        height = bar.get_height()
        y_pos = height if height > 0 else height
        va = 'bottom' if height > 0 else 'top'
        ax.text(bar.get_x() + bar.get_width()/2., y_pos,
                f'{value:.4f}',
                ha='center', va=va)

    plt.tight_layout()
    plt.show()

def plot_betweenness_comparison(metrics_dict):
    """Plot betweenness centrality distributions side by side"""
    fig, axes = plt.subplots(1, 3, figsize=(18, 5))
    for idx, (network_name, metrics) in enumerate(metrics_dict.items()):
        sns.histplot(metrics['betweenness_values'], bins=30, kde=True, ax=axes[i
        axes[idx].set_title(f"{network_name}\nAvg: {metrics['avg_betweenness']:.
        axes[idx].set_xlabel('Betweenness Centrality')
        axes[idx].set_ylabel('Frequency')

    plt.tight_layout()
    plt.show()

def plot_degree_distribution_comparison(metrics_dict):
    """Plot degree distributions side by side on log-log scale"""
    fig, axes = plt.subplots(1, 3, figsize=(18, 5))

    for idx, (network_name, metrics) in enumerate(metrics_dict.items()):
        degree_count = metrics['degree_count']
        degrees, counts = zip(*sorted(degree_count.items()))
        axes[idx].loglog(degrees, counts, marker='o', linestyle='None')
        axes[idx].set_title(f"{network_name}")
        axes[idx].set_xlabel('Degree (log scale)')
        axes[idx].set_ylabel('Count (log scale)')
        axes[idx].grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

def compare_networks(networks_dict):
    """
    Compare multiple networks with side-by-side plots
    networks_dict: dictionary with network names as keys and NetworkX graphs as
    """
    # Compute metrics for all networks
    metrics_dict = {}
    for network_name, G in networks_dict.items():
        metrics_dict[network_name] = compute_network_metrics(G, network_name)
        print(f"{network_name}:")
        print(f"  Average Clustering Coefficient: {metrics_dict[network_name]['a
        print(f"  Modularity: {metrics_dict[network_name]['modularity']:.4f}")
        print(f"  Degree Assortativity: {metrics_dict[network_name]['degree_asso
        print(f"  Average Betweenness Centrality: {metrics_dict[network_name]['a
        print(f"  Degree Distribution: {metrics_dict[network_name]['degree_count
        print()

    # Plot comparisons
    print("\n--- Clustering Coefficient Comparison ---")
    plot_clustering_comparison(metrics_dict)

    print("\n--- Modularity Comparison ---")
    plot_modularity_comparison(metrics_dict)

    print("\n--- Degree Assortativity Comparison ---")
    plot_degree_assortativity_comparison(metrics_dict)

    print("\n--- Betweenness Centrality Comparison ---")
    plot_betweenness_comparison(metrics_dict)

    print("\n--- Degree Distribution Comparison ---")
    plot_degree_distribution_comparison(metrics_dict)

####################
```

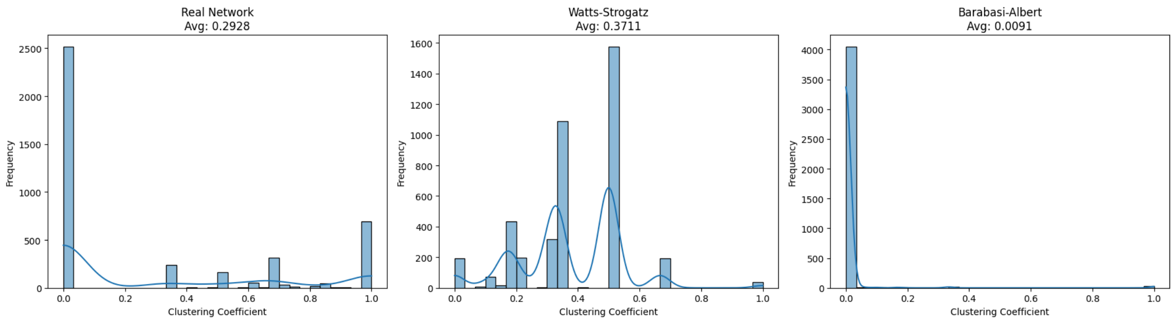
```python
# Load real network from White Helmets dataset
def load_real_network(file_path):
    G = nx.Graph()
    with open(file_path, 'r') as f:
        for line in f:
            parts = line.strip().split(',')
            if len(parts) >= 7:
                user1, user2 = parts[2], parts[3]
                G.add_edge(user1, user2)
    return G

# Creating random networks

# Creating WS model
def create_ws_model(n, k, p):
    return nx.watts_strogatz_graph(n, k, p)

# Creating BA model
def create_ba_model(n, m):
    return nx.barabasi_albert_graph(n, m)

# Print network overviews with two different layouts
def plot_network_overview(G, title):
    """Plot network with both spring and circular layout side by side"""
    fig, axes = plt.subplots(1, 2, figsize=(16, 6))

    # Spring Layout
    pos_spring = nx.spring_layout(G, seed=42)
    nx.draw(G, pos_spring, ax=axes[0], node_size=20, node_color='blue',
            edge_color='gray', alpha=0.7)
    axes[0].set_title(f"{title}\n(Spring Layout)")
    axes[0].axis('off')

    # Circular Layout
    pos_circular = nx.circular_layout(G)
    nx.draw(G, pos_circular, ax=axes[1], node_size=20, node_color='blue',
            edge_color='gray', alpha=0.7)
    axes[1].set_title(f"{title}\n(Circular Layout)")
    axes[1].axis('off')

    plt.tight_layout()
    plt.show()

# Main analysis
print("Loading networks...")
real_network = load_real_network('data/pairwise_52seconds_share.csv')
ws_network = create_ws_model(real_network.number_of_nodes(), 4, 0.1) # parameter
ba_network = create_ba_model(real_network.number_of_nodes(), 2) # parameter: n =

# Create dictionary of networks
networks = {
    "Real Network": real_network,
    "Watts-Strogatz": ws_network,
    "Barabasi-Albert": ba_network
}

# Print network overviews (takes some time)
for network_name, G in networks.items():
    plot_network_overview(G, network_name)
```

```
# Compare all networks
compare_networks(networks)
```

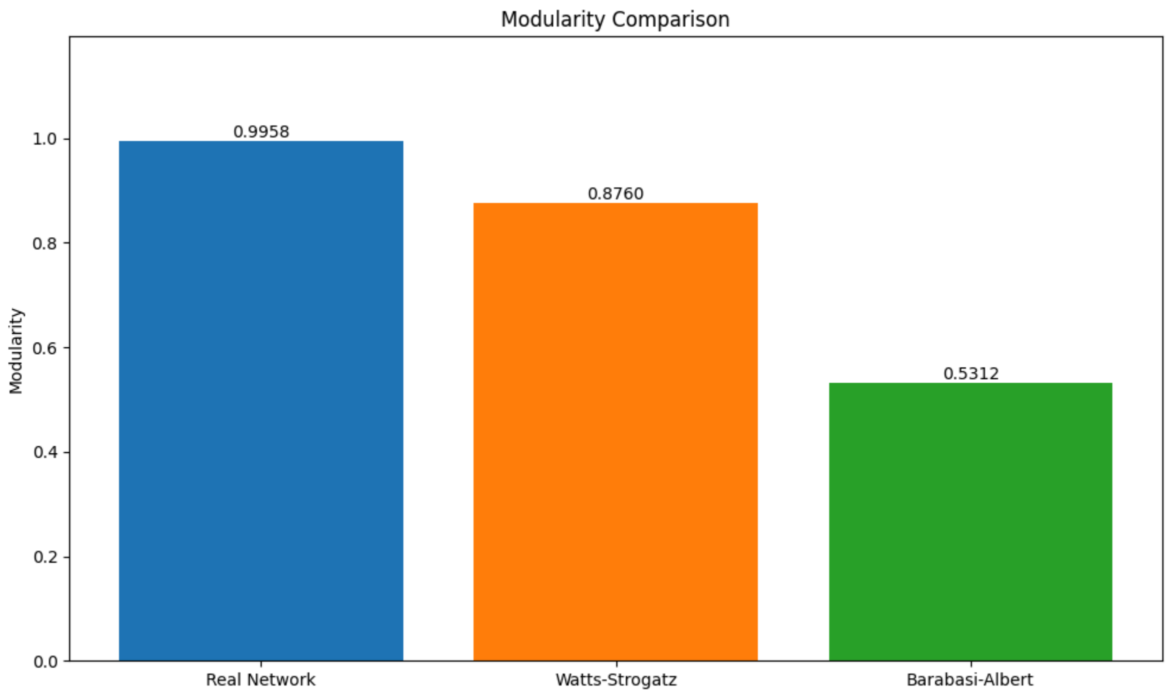
```
Loading networks...
```



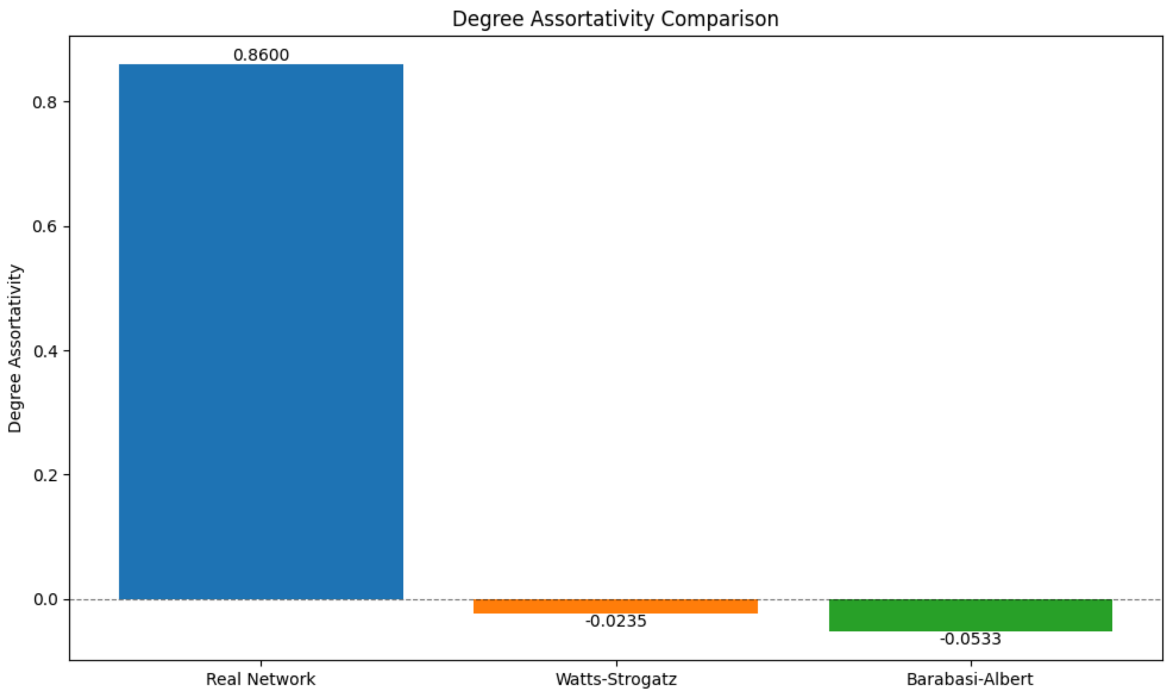
```
Computing metrics for Real Network...
Real Network:
  Average Clustering Coefficient: 0.2928
  Modularity: 0.9958
  Degree Assortativity: 0.8600
  Average Betweenness Centrality: 0.0000
  Degree Distribution: Counter({1: 1835, 2: 1174, 3: 593, 4: 305, 5: 114, 6: 41,
7: 36, 8: 22, 9: 6})

Computing metrics for Watts-Strogatz...
Watts-Strogatz:
  Average Clustering Coefficient: 0.3711
  Modularity: 0.8760
  Degree Assortativity: -0.0235
  Average Betweenness Centrality: 0.0024
  Degree Distribution: Counter({4: 2857, 3: 604, 5: 562, 6: 57, 2: 42, 7: 4})

Computing metrics for Barabasi-Albert...
Barabasi-Albert:
  Average Clustering Coefficient: 0.0091
  Modularity: 0.5312
  Degree Assortativity: -0.0533
  Average Betweenness Centrality: 0.0009
  Degree Distribution: Counter({2: 2089, 3: 822, 4: 408, 5: 211, 6: 137, 7: 100,
8: 68, 9: 55, 10: 37, 11: 31, 14: 21, 13: 18, 12: 18, 16: 15, 17: 10, 15: 10, 21:
8, 20: 6, 18: 6, 30: 5, 19: 5, 25: 4, 38: 3, 35: 3, 23: 3, 27: 3, 33: 2, 37: 2, 3
1: 2, 42: 2, 29: 2, 24: 2, 22: 2, 176: 1, 100: 1, 48: 1, 124: 1, 94: 1, 69: 1, 5
9: 1, 50: 1, 60: 1, 49: 1, 54: 1, 34: 1, 32: 1, 26: 1, 40: 1, 28: 1})


--- Clustering Coefficient Comparison ---
```



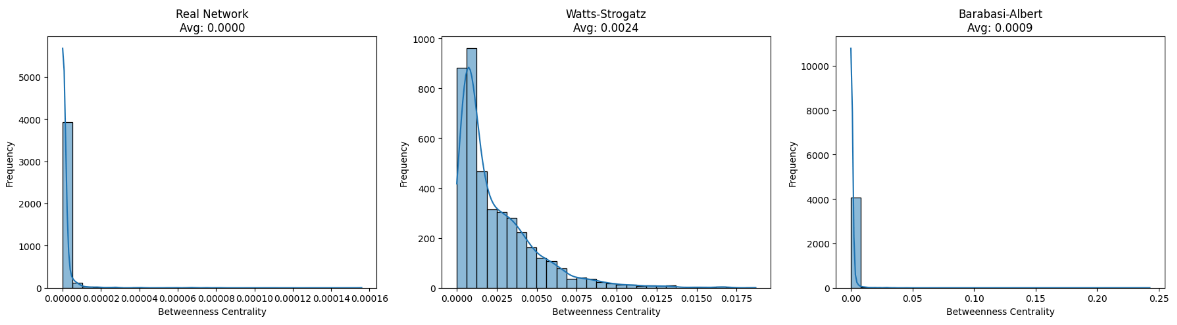
```
--- Modularity Comparison ---
```



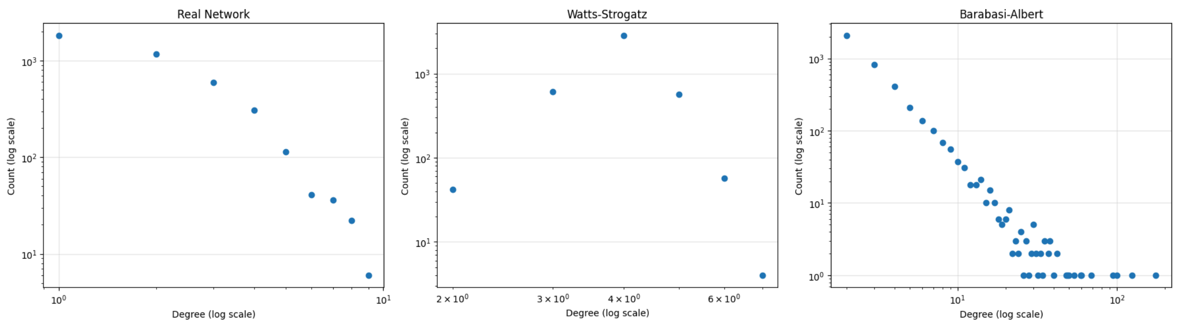
```
--- Degree Assortativity Comparison ---
```



```
--- Betweenness Centrality Comparison ---
```



```
--- Degree Distribution Comparison ---
```



Evaluation of results

Based on these metrics we can compare the 3 graphs and understand the specificities of the White Helmet dataset.

First, by comparing the average clustering coefficients of these graphs, we observe that the White Helmet (WH) network has an average CC of 0.2928, while the BA model is much lower at 0.01 and the WS model is higher at 0.3701. This indicates that the WH network exhibits a community-like organization, though not as pronounced as in a small-world network, as reflected by the higher CC of the WS model.

When we examine the modularity, we observe that the White Helmet dataset (WH) has a very high modularity score of 0.9958. This indicates that the graph consists of highly concentrated clusters with few connections between them. In contrast, the synthetic graphs exhibit much lower modularity, suggesting a less clearly defined community structure.

The degree assortativity is also useful for understanding how the White Helmet organization operates. We observe that the White Helmet (WH) graph is the only one with a positive assortativity coefficient of 0.860, compared to –0.0229 for WS and –0.0627 for BA. This indicates that nodes in the WH network tend to connect primarily with similar nodes, whereas natural networks are typically neutral or disassortative. A high level of assortativity, combined with high modularity, often suggests homophily or the presence of echo chambers within the network.

When examining betweenness centrality, the White Helmet (WH) network stands out once again, as it contains no nodes functioning as bridges between communities. In organic or random networks, a small number of nodes typically connect otherwise separate groups. This observation supports the idea that WH communities are highly self-contained or operate in isolation—consistent with the behavior of coordinated campaigns that do not depend on a few central disseminators.

The degree distribution of the real network shows that many nodes have only a few connections, while there are no highly connected hubs.This indicates that the network does not follow a scale-free structure.

Thanks to these metrics, we can understand how the information campaign unfolded. The White Helmet network is not a naturally random network. It does not exhibit a Small-World structure but is highly clustered into groups that appear to operate independently within the network. This indicates that the entire graph is organized—not through individual actors, but through a decentralized strategy. Each group functions as a proxy for disseminating the information campaign.

Does the analysis make the case of organic/inorganic patterns?

The analysis suggests that the observed patterns are inorganic rather than organic. The structure of the White Helmet network does not emerge naturally from spontaneous individual interactions. Instead, its high concentration in seemingly independent clusters and the absence of a typical Small-World organization indicate a coordinated, decentralized strategy. This implies that the dissemination of information was likely orchestrated rather than organically formed.