Supplementary Material A:

End-to-End Workflow and Usage Tutorial for OppNDA

Contents

1 Introduction

OppNDA (Opportunistic Network Data Analyzer) is a comprehensive framework that streamlines the workflow for configuring, running, and analyzing simulations in the ONE (Opportunistic Network Environment) simulator. This document provides a step-by-step walkthrough of the framework, supplementing the main manuscript titled **OppNDA: A Modular and Scalable Automation Framework for Streamlining DTN Research with the ONE Simulator**.

1.1 Workflow Overview

The complete OppNDA workflow consists of the following stages:

1. **Setup**: Install dependencies and launch the application
2. **Configuration**: Define simulation scenarios using the web interface
3. **Simulation**: Execute the ONE simulator with configured parameters
4. **Averaging**: Aggregate results across multiple simulation seeds
5. **Analysis**: Generate publication-ready visualizations
6. **Regression**: Train machine learning models on simulation data

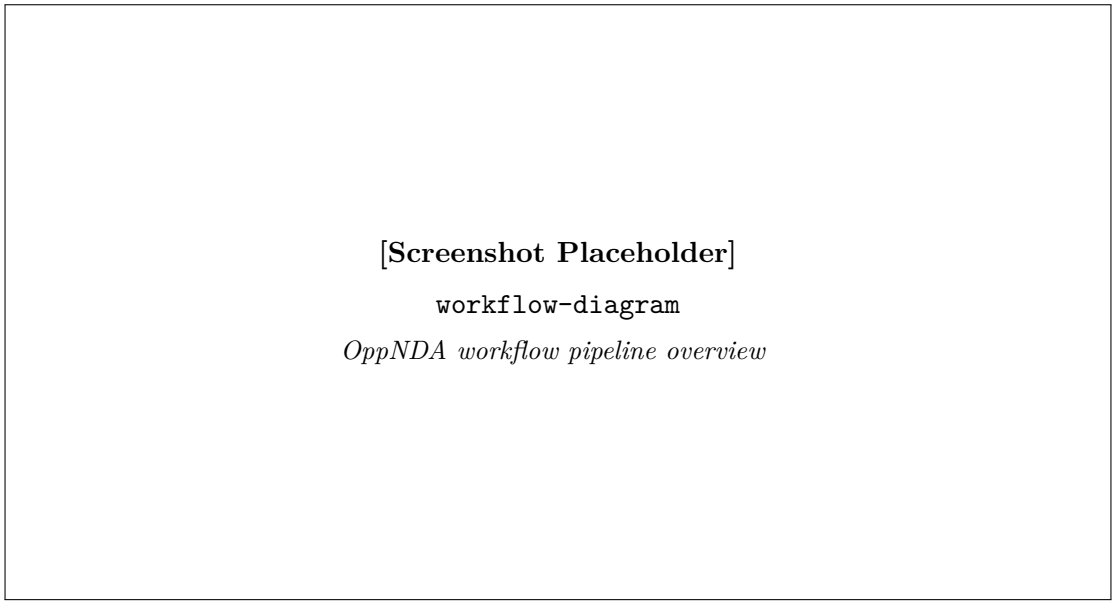

Figure 1: OppNDA workflow pipeline overview

1.2 Prerequisites

Before proceeding, ensure you have the following:

- Python 3.9 or higher

- ONE Simulator (for generating simulation reports)
- A modern web browser (Chrome, Firefox, Edge)
- Git (for cloning the repository)
- Docker (optional, for containerized deployment)

2 Installation and Setup

OppNDA provides multiple installation methods to accommodate different environments and preferences.

2.1 Method 1: Automated Setup Scripts

The simplest installation method uses the provided setup scripts.

2.1.1 Windows

1. Download or clone the OppNDA repository:

```
git clone https://github.com/nafisshahriar/oppnda.git
cd oppnda
```

2. Run the setup script:

```
scripts\setup.bat
```

3. Launch the application:

```
scripts\start.bat
```

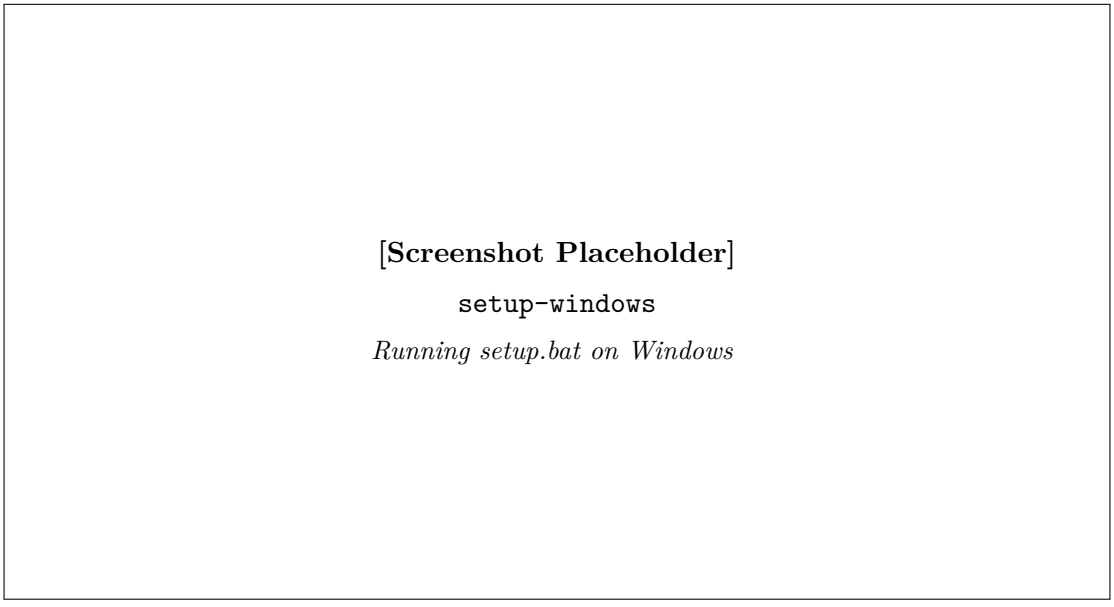

Figure 2: Running setup.bat on Windows

2.1.2 Linux/macOS

1. Clone and navigate to the repository:

```
git clone https://github.com/nafisshahriar/oppnda.git
cd oppnda
```

2. Run the setup script:

```
bash scripts/setup.sh
```

3. Launch the application:

```
bash scripts/start.sh
```

2.2 Method 2: Manual Installation

For more control over the installation process:

```
# Create and activate virtual environment
python -m venv venv
source venv/bin/activate    # Linux/macOS
venv\Scripts\activate       # Windows

# Install dependencies
pip install -r requirements.txt

# Run the application
python OppNDA.py
```

2.3 Method 3: Docker Installation

For containerized deployment, OppNDA includes Docker support:

2.3.1 Using Docker Compose (Recommended)

```
# Build and run with Docker Compose
docker-compose up --build
```

2.3.2 Manual Docker Build

```
# Build the Docker image
docker build -t oppnda .

# Run the container
docker run -p 5001:5001 --name OppNDA oppnda
```

[Screenshot Placeholder]

`docker-running`

OppNDA running in a Docker container

Figure 3: OppNDA running in a Docker container

Tip

Docker is particularly useful for consistent deployment across different machines and for isolated testing environments. The containerized version includes all dependencies pre-configured.

2.4 Accessing the Web Interface

After launching OppNDA, open your web browser and navigate to:

`http://localhost:5001/settings`

[Screenshot Placeholder]

`homepage`

OppNDA web interface home page

Figure 4: OppNDA web interface home page

3 Scenario Configuration

The Settings page provides a comprehensive interface for configuring ONE simulator scenarios. This section covers both creating new configurations and importing existing ones.

3.1 Importing Existing Configuration Files

OppNDA supports importing existing ONE simulator configuration files (`.txt` format) directly into the GUI.

1. Click the **Import Config** button in the toolbar
2. Select your existing ONE configuration file
3. The GUI will parse and populate all settings automatically
4. Review and modify imported settings as needed

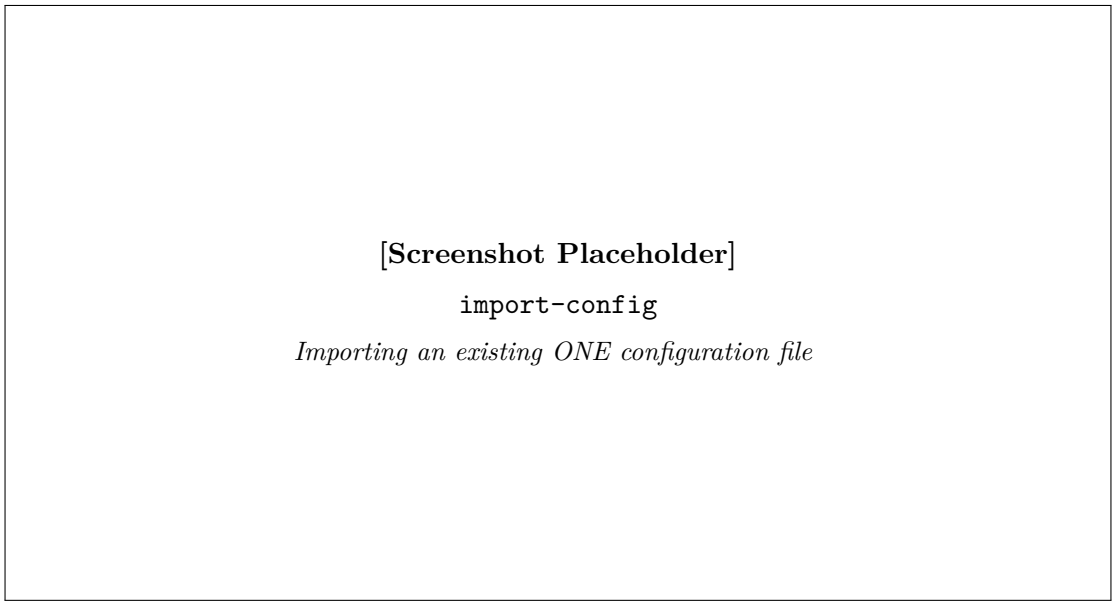

Figure 5: Importing an existing ONE configuration file

Note

When importing, OppNDA automatically recognizes standard ONE parameters and maps them to the corresponding GUI fields. Unknown parameters are preserved and can be viewed in the advanced settings section.

3.2 Scenario Settings

The basic scenario settings define the simulation environment:

1. **Scenario Name**: Enter a descriptive name for your simulation
2. **Simulation Time**: Set the duration in seconds
3. **Update Interval**: Configure the simulation tick rate
4. **World Size**: Define the simulation area dimensions (X, Y)

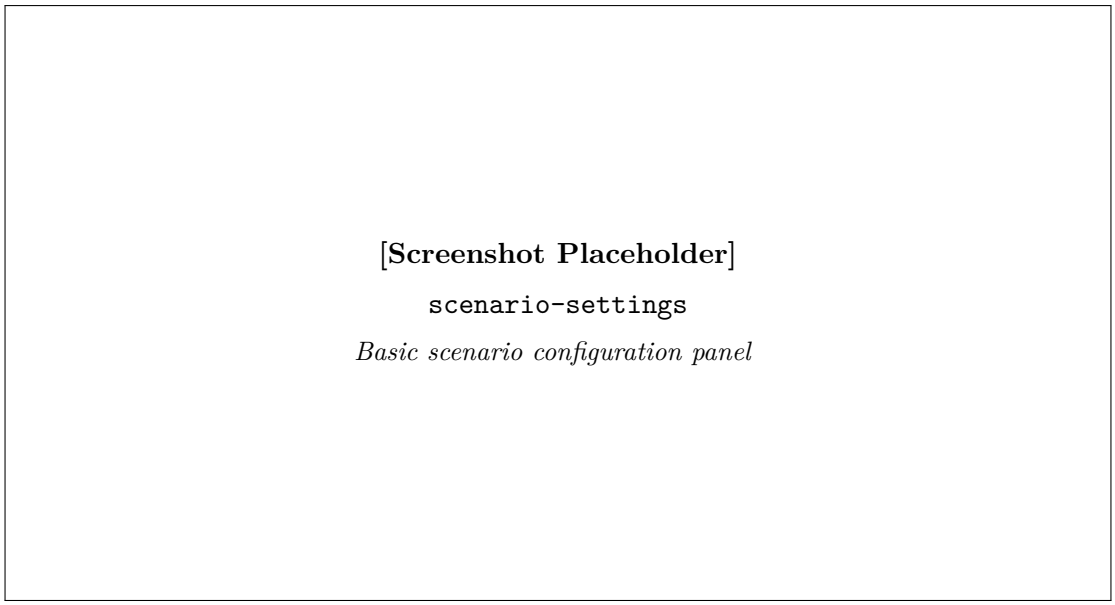

Figure 6: Basic scenario configuration panel

3.3 Network Interfaces

Configure the communication interfaces available to nodes:

1. Click **Add Interface** to create a new interface
2. Select the interface type (e.g., SimpleBroadcastInterface)
3. Configure parameters:
    - Transmission speed
    - Transmission range

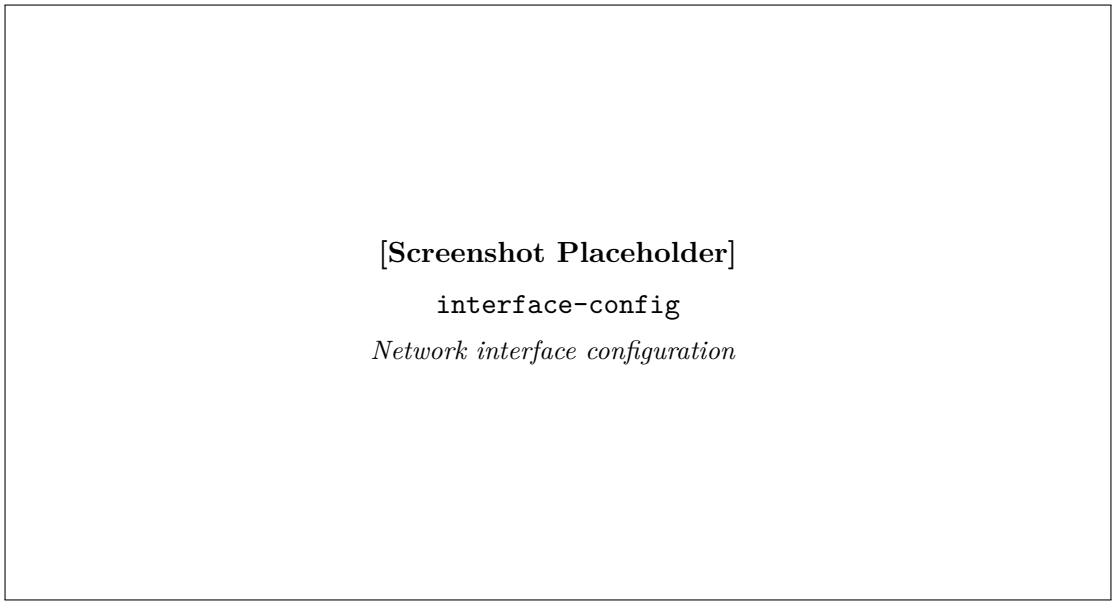

Figure 7: Network interface configuration

3.4 Host Groups

Define different categories of mobile nodes:

1. Click **Add Group** to create a new host group
2. Configure group parameters:
    - Group ID and number of hosts
    - Movement model (RandomWaypoint, ShortestPathMapBasedMovement, etc.)
    - Buffer size
    - Message TTL (Time-To-Live)
    - Router type (Epidemic, SprayAndWait, PRoPHET, etc.)

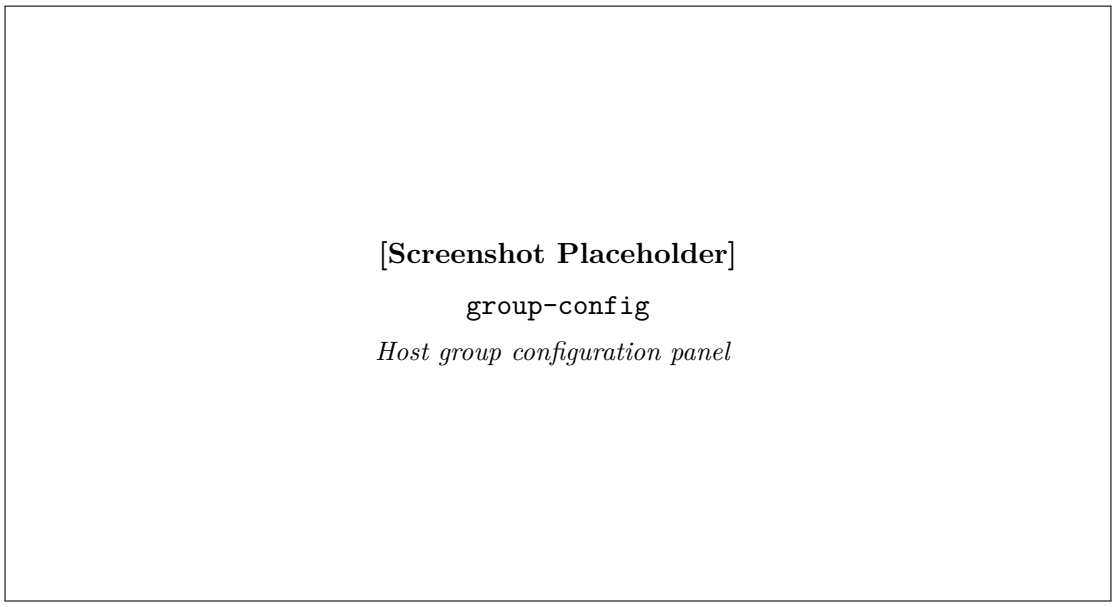

Figure 8: Host group configuration panel

3.5 Message Events

Configure message generation patterns:

1. Click **Add Event** to create a new event generator
2. Set event parameters:
    - Event class (MessageEventGenerator)
    - Message creation interval (min, max)
    - Message size range
    - Source and destination host ranges
    - Event timing (start, end)

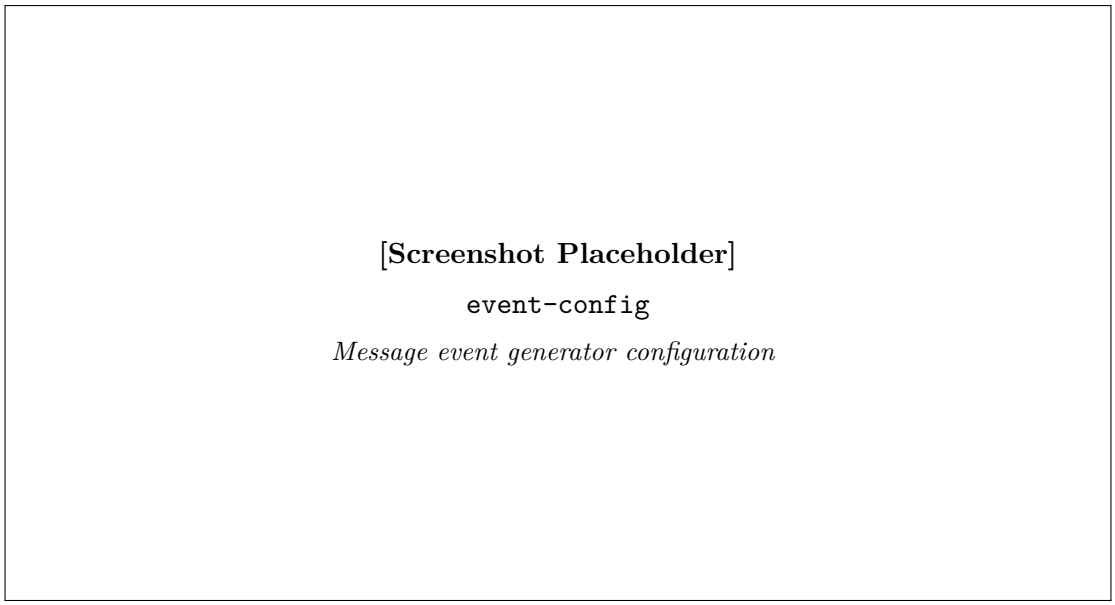

Figure 9: Message event generator configuration

3.6 Report Selection

Choose which reports the ONE simulator should generate:

1. Browse the available report types
2. Select desired reports (e.g., MessageStatsReport, DeliveredMessagesReport)
3. Configure report-specific parameters if needed
4. Set the report output directory

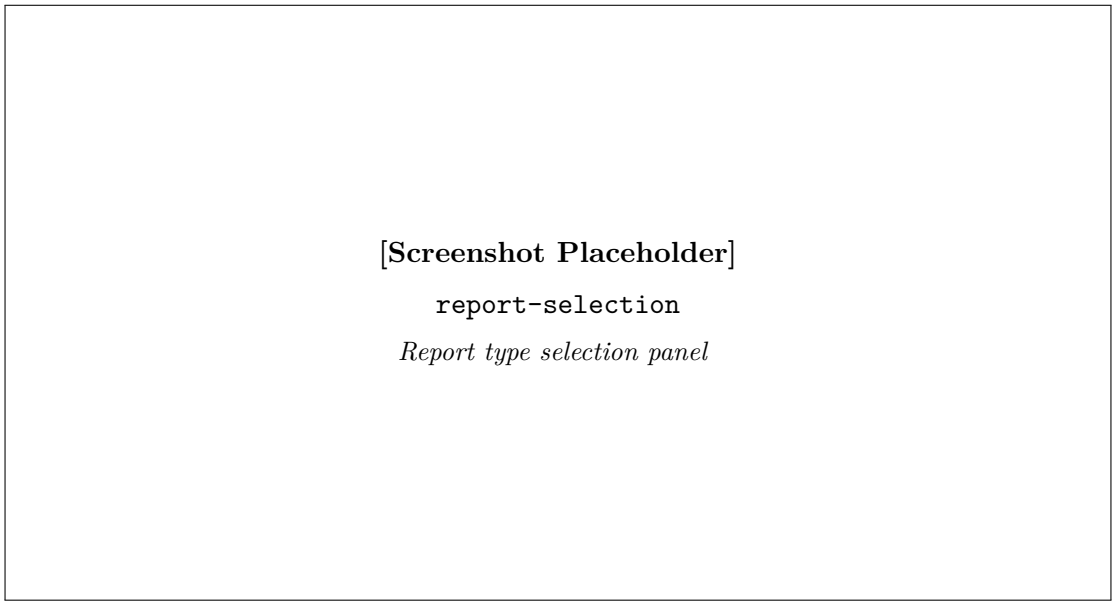

Figure 10: Report type selection panel

3.7 Saving Configuration

After configuring all parameters:

1. Click **Save Config** to save the current configuration
2. The configuration is saved as a `.txt` file compatible with ONE
3. OppNDA also maintains JSON backups of all settings

4 Running the ONE Simulator

OppNDA provides integrated execution of the ONE simulator with real-time output monitoring.

4.1 Starting a Simulation

1. Ensure your configuration is saved
2. Click the **Run ONE** button
3. OppNDA will:
   - Save the current configuration
   - Launch the ONE simulator with appropriate parameters
   - Display real-time console output
   - Automatically trigger post-processing upon completion

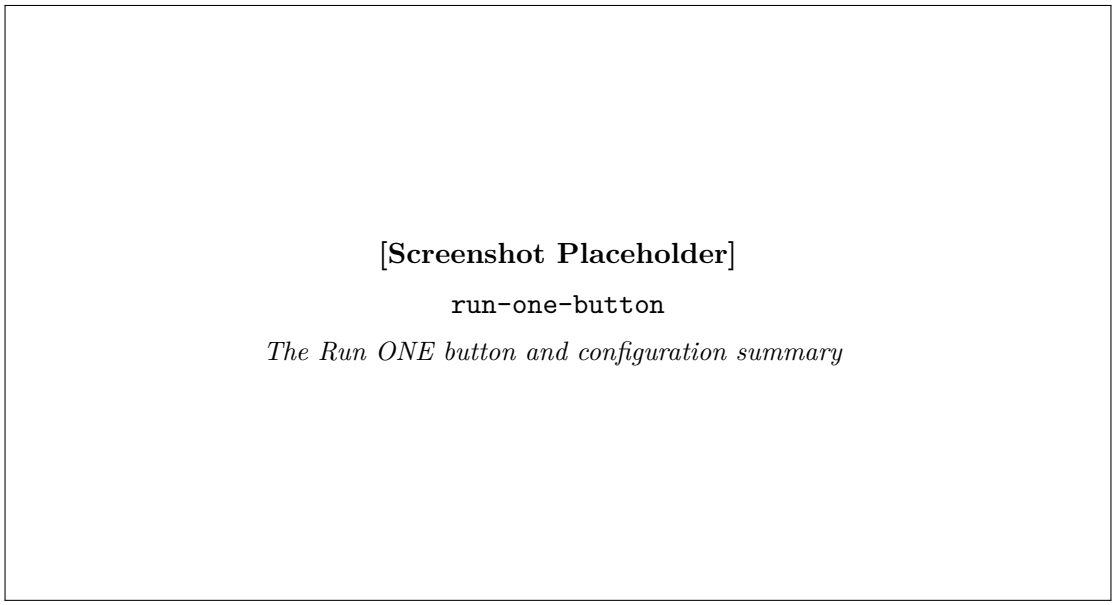

Figure 11: The Run ONE button and configuration summary

4.2 Monitoring Progress

The console panel displays real-time output from the simulator:

- Simulation progress percentage

- Current simulation time
- Messages created and delivered
- Any warnings or errors

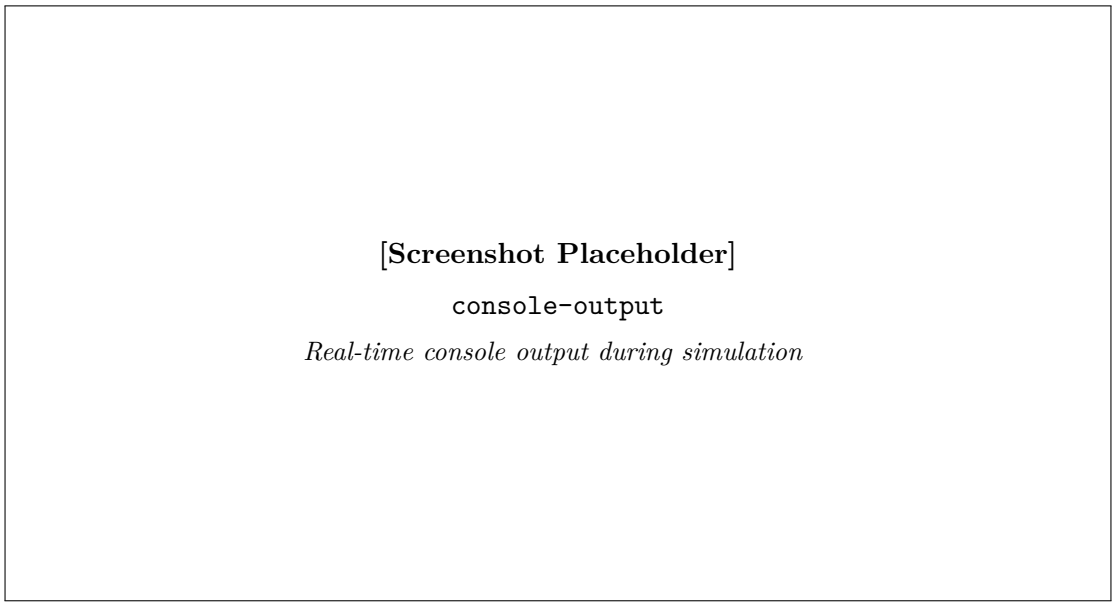

Figure 12: Real-time console output during simulation

Tip

For batch simulations with multiple parameter combinations, you can configure seed ranges and parameter variations. OppNDA will execute simulations sequentially and aggregate results.

4.3 Terminating a Simulation

If needed, you can stop a running simulation:

1. Click the **Terminate** button next to the Run ONE button
2. Confirm the termination
3. Any partial results will be preserved

5 Post-Processing: Report Averaging

The averaging module aggregates results across multiple simulation runs (seeds) to produce statistically meaningful data.

5.1 Understanding Filename Patterns

OppNDA uses configurable filename patterns to group and average reports. A typical ONE report filename follows this structure:

```
RouterType_TTL_BufferSize_Seed_ReportType.txt
```

For example: `Epidemic_300_5M_1_MessageStatsReport.txt`

5.2 Configuring the Averager

1. Navigate to the **Post-Processing** section
2. Set the **Input Directory** containing raw report files
3. Configure the **Filename Pattern** using the drag-and-drop pattern builder
4. Set **Grouping Parameters** to define which parameters to group by (e.g., Router, TTL, Buffer)
5. Set the **Seed Position** to identify which part of the filename represents the seed number

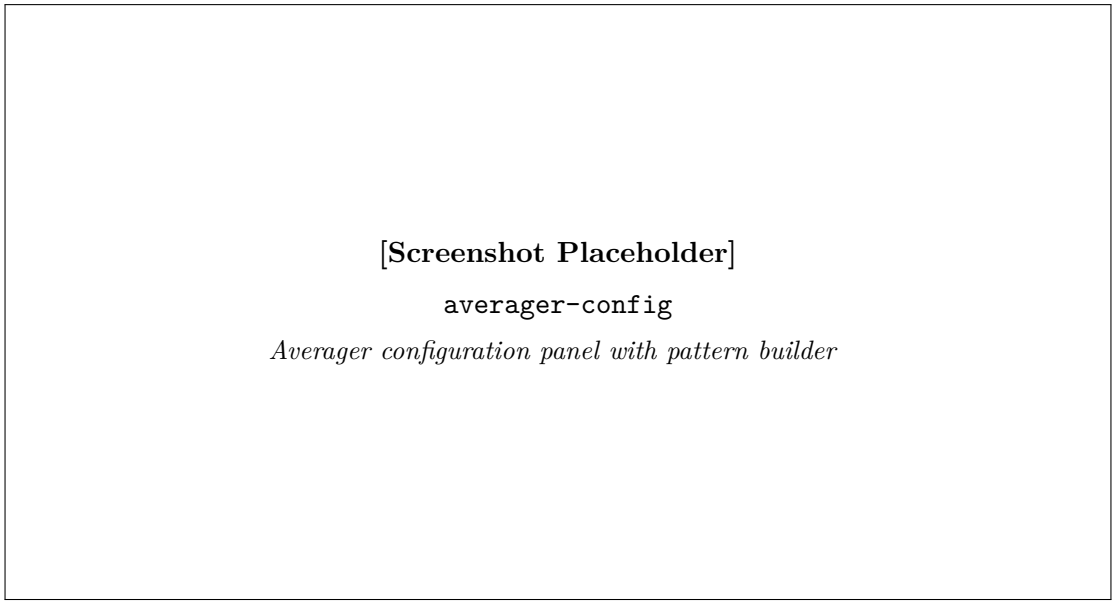

Figure 13: Averager configuration panel with pattern builder

5.3 Pattern Builder

The interactive pattern builder helps define filename patterns:

1. Drag components (Router, TTL, Buffer, Seed, etc.) into position
2. Set delimiters between components (underscore, hyphen, etc.)
3. Preview how patterns match your actual files

5.4 Running the Averager

1. Click **Run Averager**
2. Monitor progress in the console panel
3. Averaged files are saved with `averaged_` prefix in the output directory

[Screenshot Placeholder]

`pattern-builder`

Interactive pattern builder interface

Figure 14: Interactive pattern builder interface

[Screenshot Placeholder]

`averager-output`

Averager console output showing grouped files

Figure 15: Averager console output showing grouped files

5.5 Understanding Averaged Output

The averaged output files contain:

- Mean values across all seeds
- Standard deviation for each metric
- Count of samples averaged

6 Post-Processing: Visualization and Analysis

The analysis module generates publication-ready visualizations from averaged data.

6.1 Configuring Analysis

1. Select the **Input Directory** containing averaged reports
2. Choose **Report Types** to analyze
3. Configure axis parameters (X, Y, Z variables)
4. Set desired **Plot Types**

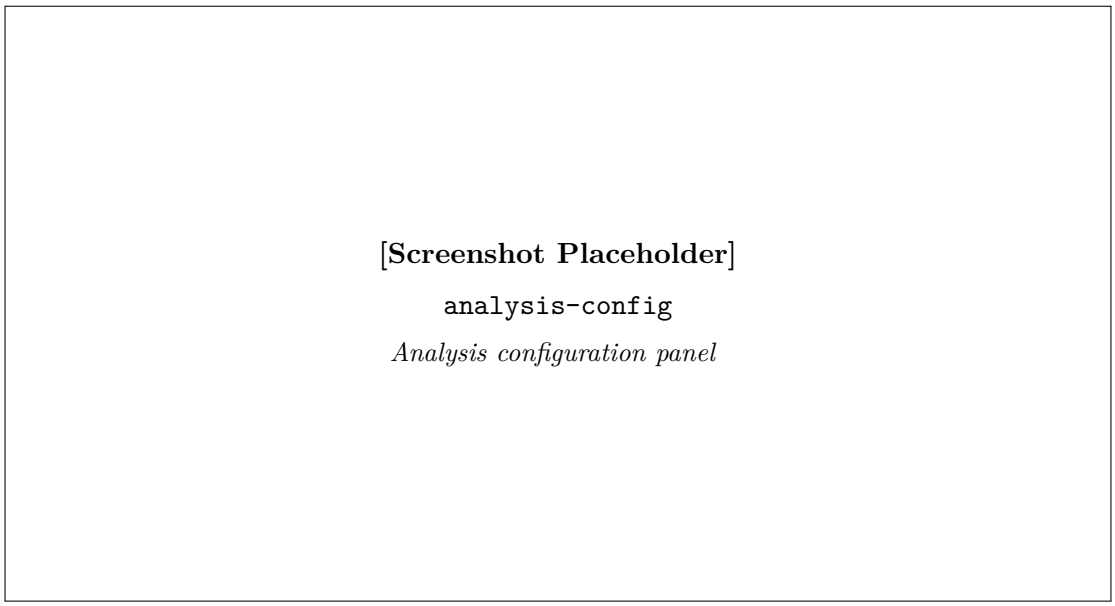

Figure 16: Analysis configuration panel

6.2 Available Plot Types

OppNDA supports multiple visualization types:

- **3D Surface Plots**: Visualize relationships between three variables
- **Line Plots**: Show trends across parameter values
- **Heatmaps**: Display intensity matrices
- **Violin Plots**: Show data distribution
- **Pair Plots**: Explore correlations between metrics

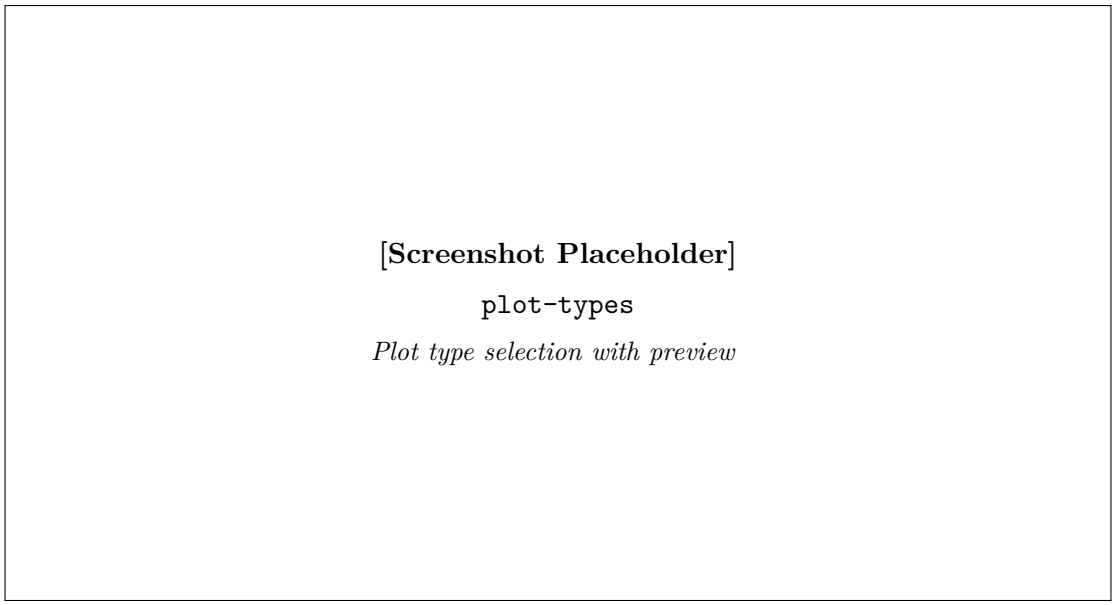

Figure 17: Plot type selection with preview

6.3 Plot Settings

Customize plot appearance:

1. **Figure Size**: Set dimensions in inches
2. **Font Sizes**: Configure title, label, and tick fonts
3. **Color Scheme**: Choose from available color palettes
4. **DPI**: Set resolution for saved images

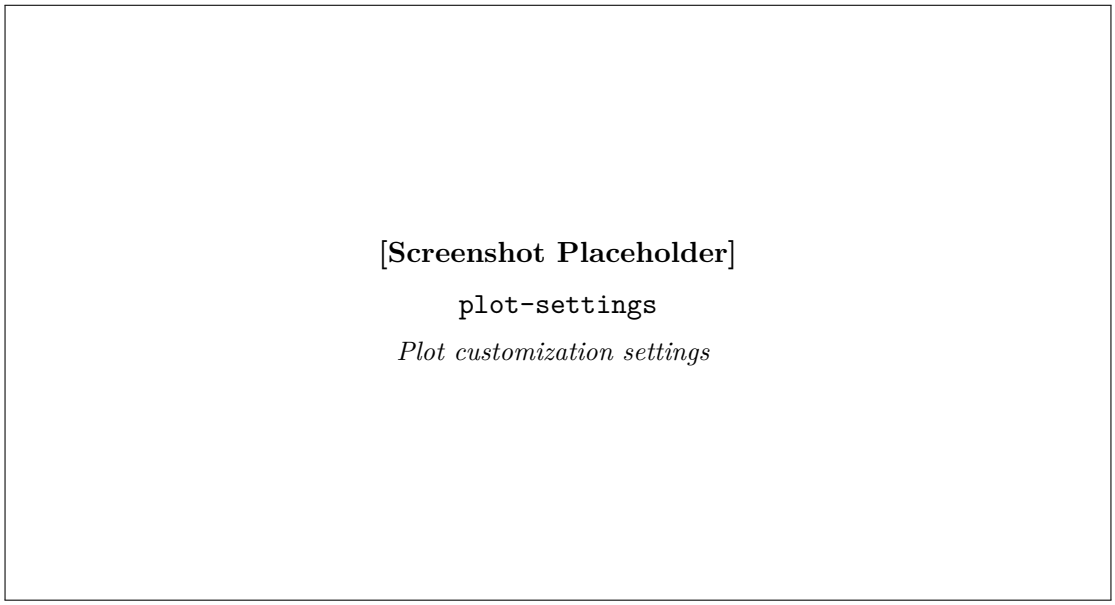

Figure 18: Plot customization settings

6.4 Running Analysis

1. Click **Run Analysis**
2. Monitor progress as plots are generated
3. Generated plots are saved to the `plots/` directory

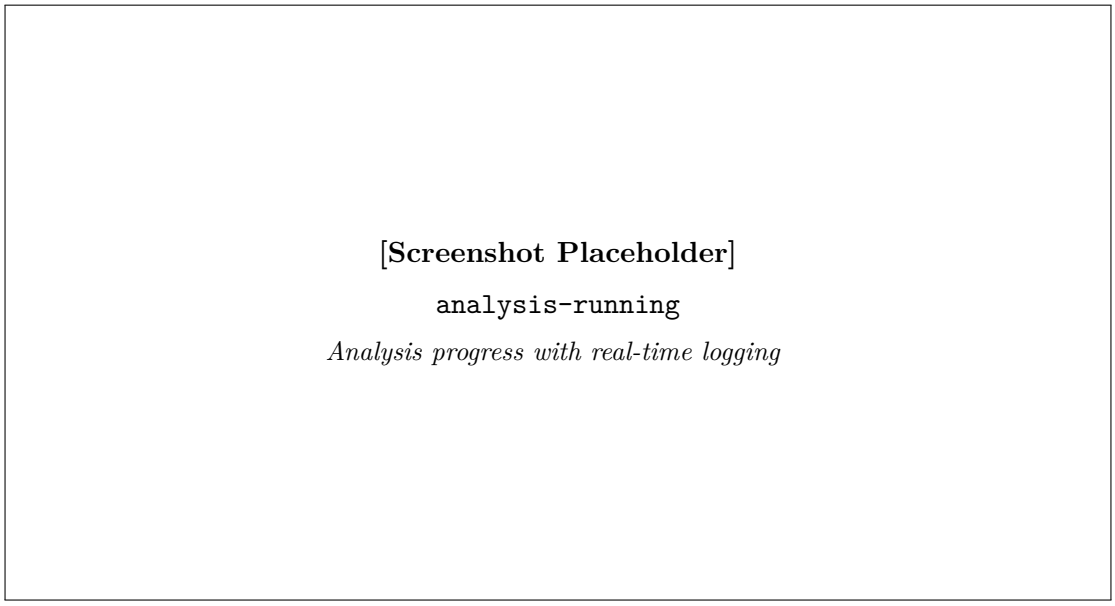

Figure 19: Analysis progress with real-time logging

6.5 Viewing Generated Plots

After analysis completes:

- View plots directly in the web interface
- Access high-resolution versions in the `plots/` directory
- Download plots in PNG or PDF format

7 Regression Analysis

The regression module trains machine learning models to understand and predict network performance metrics.

7.1 Preparing Input Data

Regression analysis uses CSV files generated during the analysis phase:

1. Ensure analysis has been run to generate CSV output
2. The CSV files contain structured data suitable for ML training

[Screenshot Placeholder]

`generated-3d-plot`

Example 3D surface plot showing delivery ratio

Figure 20: Example 3D surface plot showing delivery ratio

[Screenshot Placeholder]

`generated-heatmap`

Example heatmap visualization

Figure 21: Example heatmap visualization

7.2 Configuring Regression

1. Navigate to the **Regression** section
2. Select the **Input CSV File**
3. Choose **Target Variable(s)** to predict (e.g., delivery_ratio, latency)
4. Select **Predictor Variables** (e.g., TTL, buffer_size, num_hosts)
5. Choose **ML Models** to train

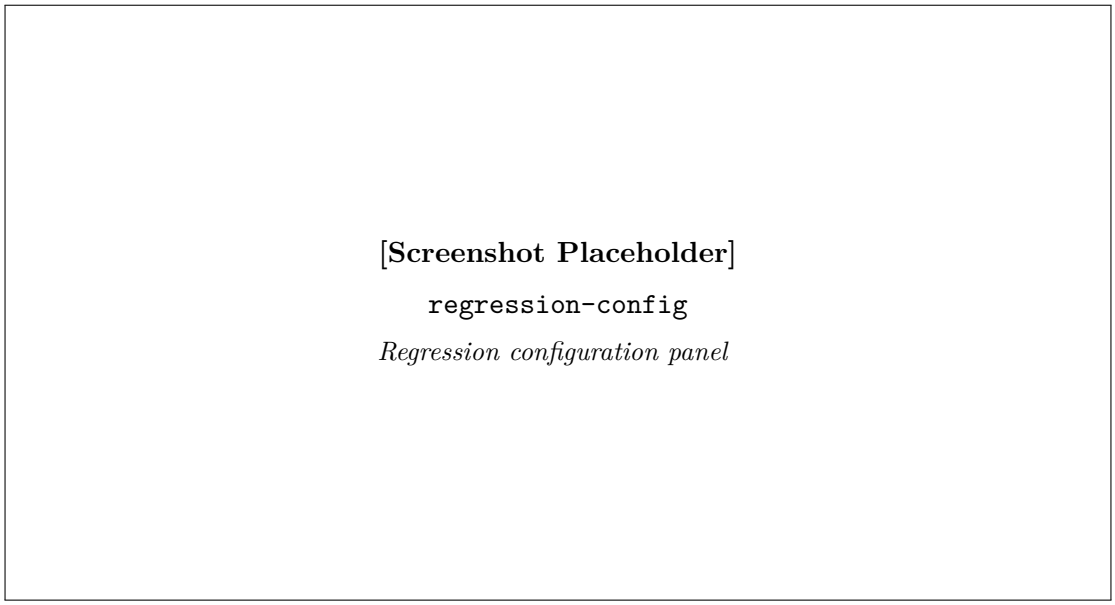

Figure 22: Regression configuration panel

7.3 Available ML Models

OppNDA supports multiple regression algorithms:

- **Linear Regression**: Simple interpretable model
- **Ridge Regression**: L2 regularized linear model
- **Lasso Regression**: L1 regularized for feature selection
- **Random Forest**: Ensemble tree-based method
- **Gradient Boosting**: Sequential ensemble method
- **Support Vector Regression**: Kernel-based method

7.4 Multi-Target Regression

OppNDA supports training models on multiple target variables simultaneously:

1. Select multiple target variables from the dropdown
2. Models are trained independently for each target
3. Results are aggregated for comparison

7.5 Running Regression

1. Click **Run Regression**
2. Monitor training progress
3. View model performance metrics (Rš, RMSE, MAE)

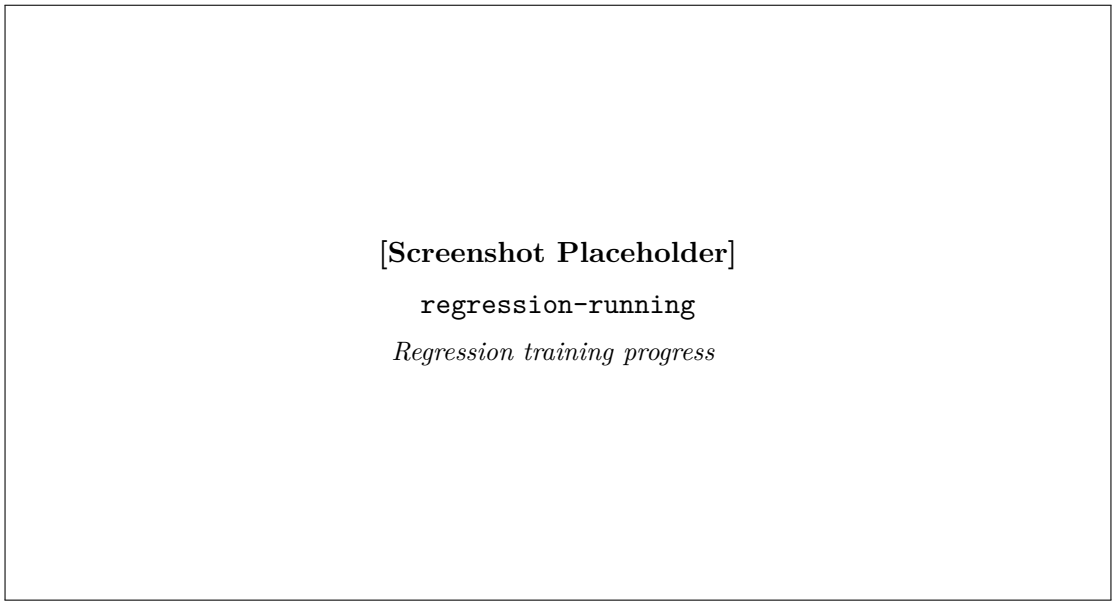

Figure 23: Regression training progress

7.6 Interpreting Results

The regression results panel displays:

- Model performance comparison table
- Feature importance rankings
- Actual vs. predicted scatter plots
- Cross-validation scores

Tip

Use feature importance to identify which simulation parameters have the greatest impact on network performance metrics. This can guide future simulation parameter selection.

8 Conclusion

This tutorial has covered the complete OppNDA workflow from installation through regression analysis. Key takeaways:

1. **Flexible Installation**: Choose between scripts, manual setup, or Docker
2. **Intuitive Configuration**: Web-based GUI simplifies ONE simulator setup

[Screenshot Placeholder]

`regression-results`

Regression results and model comparison

Figure 24: Regression results and model comparison

3. **Config Import**: Easily import and modify existing configuration files
4. **Integrated Execution**: Run simulations and monitor in real-time
5. **Automated Post-Processing**: Seamless averaging and visualization
6. **Advanced Analytics**: ML-based regression for performance insights

8.1 Tips for Advanced Usage

- Use the performance optimization settings to handle large datasets
- Leverage the dynamic memory management for parallel processing
- Customize plot styles through the analysis configuration JSON
- Export configurations for reproducible experiments

8.2 Further Resources

- OppNDA Documentation: https://oppnda.readthedocs.io
- ONE Simulator: https://github.com/akeranen/the-one
- GitHub Repository: https://github.com/nafisshahriar/oppnda

DHMAI Network Research Group
https://dhmairg.net
2026