Годишник Телекомуникации 2024, том 11, с. 53-60

Yearbook Telecommunications 2024, vol. 11, p. 53-60

eISSN 2534-854X https://telecommunications.nbu.bg/bg/godishnik-telekomunikacii

DOI: https://doi.org/10.33919/YTelecomm.24.11.6

AUTOMATED MONITORING AND ANALYSIS OF NETWORK TRAFFIC IN LINUX USING SCRIPTS AND SYSTEM TOOLS

Stanislav Georgiev Peltekov

1. INTRODUCTION

Effective network monitoring is vital for ensuring optimal system performance, reducing latency, diagnosing bottlenecks, and detecting anomalous behavior that could indicate security threats. With its flexibility, reliability, and ubiquity, Linux serves as an ideal platform for deploying robust monitoring solutions tailored to diverse system needs [1, 2]. This project focuses on automating the collection and analysis of network traffic data in Linux environments. By leveraging only built-in system tools and utilities, the solution provides a cost-effective, lightweight, and easily deployable monitoring framework. The integration of real-time data logging and graphical visualization ensures actionable insights for system administrators and IT professionals, reducing manual intervention and enabling proactive management. In addition to leveraging Linux tools, the development process of this project was significantly accelerated through the use of ChatGPT. By employing ChatGPT as a programming assistant, we were able to:

- **Enhance Productivity:** Generate code snippets for repetitive or complex tasks, such as JSON parsing, data formatting, and log management.
- **Debug and Refine Scripts:** Quickly identify and resolve issues, validate logic, and optimize script performance with ChatGPT's ability to analyze and suggest improvements.
- **Documentation and Learning:** Create clear, professional documentation for scripts and methodologies, while also gaining insights into advanced usage of tools like vnstat, jq, and gnuplot.
- **Streamline Collaboration:** Foster a more efficient workflow by generating templates, flowcharts, and explanations for team members unfamiliar with specific tools.

By combining Linux's inherent capabilities with ChatGPT's programming support, this project demonstrates how automation and AI-powered assistance can together address complex monitoring challenges. This synergy not only reduces development time but also ensures a robust, scalable solution that can adapt to evolving network monitoring requirements.

2. SCRIPTS

The proposed system is based on two custom scripts: scritpt.sh and GNU.sh, which collectively enable real-time network monitoring and visual performance analysis [3]. **scritpt.sh:** This script is designed to measure and log critical network metrics, including latency, incoming and outgoing traffic, and the number of active network connections. Running at 10-second intervals, it collects data in real time and appends it to a log file for subsequent analysis. **GNU.sh:** Leveraging the data collected by scritpt.sh, this script utilizes the powerful gnuplot tool to generate graphical representations of network performance. These visualizations include:

- Network latency (measured in milliseconds).
- Incoming and outgoing traffic (quantified in megabytes).

- Number of active network connections (at any given interval).

By automating both data collection and analysis, this project provides a streamlined and reliable approach to network performance monitoring using fast and simple techniques. The integration of real-time logging with graphical analysis enables system administrators to quickly identify trends, diagnose issues, and optimize network configurations. Furthermore, the automated nature of this solution reduces the need for constant manual intervention, allowing IT professionals to focus on higher-level tasks. Through this project, the aim is to demonstrate how simple yet effective tools in Linux can be combined to address complex monitoring challenges, offering both technical efficiency and user-friendly insights into network performance. Together, these scripts provide a complete solution for monitoring, logging, and visualizing network performance in Linux. They are particularly useful for system administrators and IT professionals who need to identify trends, diagnose issues, or optimize network configurations with minimal manual intervention

2.1. Script 1: scritpt.sh

The first script, **scritpt.sh**, is designed to automate the process of monitoring and logging key network performance metrics in real-time. These metrics are stored in a log file (GNU.txt) for subsequent analysis. Flowchart of code is shown on Fig. 1. The script operates at 10-second intervals and performs the following key tasks:

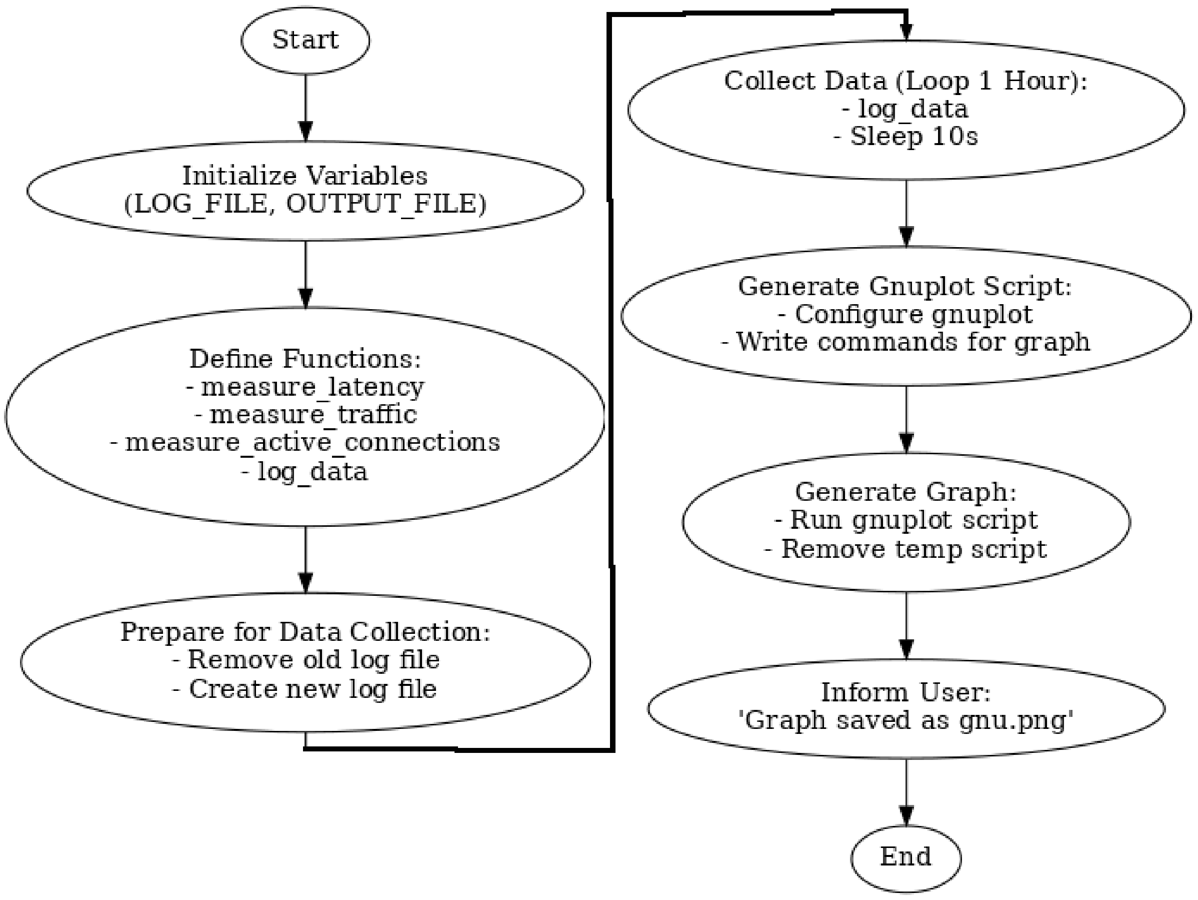
Start
Initialize Variables (LOG_FILE, OUTPUT_FILE)
Define Functions: - measure_latency - measure_traffic - measure_active_connections - log_data
Prepare for Data Collection: - Remove old log file - Create new log file
Collect Data (Loop 1 Hour): - log_data - Sleep 10s
Generate Gnuplot Script: - Configure gnuplot - Write commands for graph
Generate Graph: - Run gnuplot script - Remove temp script
Inform User: 'Graph saved as gnu.png'
End

Figure 1. Flowchart of main script **scritpt.sh**

2.1.1. Key Functions of scritpt.sh:

The `scritpt.sh` script is designed to automate the real-time collection of essential network performance metrics. It leverages widely used Linux utilities to provide accurate, actionable data. Below are the key functions of the script:

1. **Measure Network Latency**
   This function uses the `ping` command to calculate the average round-trip time (RTT) to a remote server (Google). By extracting the "average latency" from multiple pings, the script provides a reliable metric for evaluating network responsiveness. The output is recorded in milliseconds (ms), offering insights into the quality of the network connection and potential delays [2].

2. **Measure Network Traffic**
   The script collects network traffic statistics using `vnstat` in JSON format. These statistics include incoming (`rx`) and outgoing (`tx`) traffic volumes in bytes. The data is processed using `jq`, a lightweight JSON processor, to extract relevant fields. To enhance readability and usability, these byte values are converted into megabytes (MB), allowing system administrators to better interpret network usage trends.

3. **Count Active Network Connections**
   The `netstat` command is utilized to identify and count active network connections in the `ESTABLISHED` state. This metric provides a real-time snapshot of the number of ongoing communication sessions, enabling administrators to monitor connection loads and detect unusual patterns indicative of potential issues such as unauthorized access or system overload.

4. **Log Data with Timestamps**
   To facilitate seamless analysis and visualization, the script logs all collected data in a space-separated format, optimized for compatibility with `gnuplot`. Each log entry includes:

   - A timestamp to indicate when the data was collected.
   - Metrics such as latency, incoming traffic, outgoing traffic, and active connections.

   The logging mechanism ensures that data is well-structured and easily parsable for generating detailed graphical reports.

5. **Automate Data Collection**
   To ensure continuous monitoring, the script operates in an automated loop, collecting data at fixed intervals (10 seconds by default). This functionality allows for:

   - Consistent monitoring over a predefined period (e.g., 1 hour).
   - A comprehensive, time-based view of network activity, making it easier to identify performance trends and anomalies.

The automated data collection not only reduces the need for manual intervention but also ensures that all critical network metrics are captured systematically, enabling robust analysis and troubleshooting.

2.1.2. Code for scritpt.sh

```
#!/bin/bash
# Define the log file paths
LOG_FILE="./GNU.txt"
OUTPUT_FILE="gnu.png"
# Function to measure latency
measure_latency() {
ping -c 5 google.com | grep ’avg’ | awk -F’/’ ’{print $5}’
}
11
# Function to measure traffic (incoming and outgoing) using jq with vnstat JSON output
measure_traffic() {
# Use vnstat in JSON format and parse with jq to get only incoming and outgoing data
vnstat --json | jq ’.interfaces[0].traffic.total.rx, .interfaces[0].traffic.total.tx’
}
# Function to measure active connections
measure_active_connections() {
netstat -an | grep ’ESTABLISHED’ | wc -l
}

# Function to log data to a file
log_data() {
timestamp=$(date "+%Y-%m-%d %H:%M:%S")
latency=$(measure_latency)
27
# Separate incoming and outgoing traffic, convert from bytes to MB
traffic=($(measure_traffic))
incoming_traffic=$(printf "%.2f" "$(echo "${traffic[0]} / 1048576" | bc -l)") # Convert to MB
outgoing_traffic=$(printf "%.2f" "$(echo "${traffic[1]} / 1048576" | bc -l)") # Convert to MB
32
                                 33    active_connections=$(measure_active_connections)
34
# Log the data in space-separated format for easy gnuplot parsing
echo "$timestamp $latency $incoming_traffic $outgoing_traffic $active_connections" >> "
    a     $LOG_FILE"
}
38
# Remove the old log file if it exists to create a new one
rm -f "$LOG_FILE"
41
42 # Ensure the log file exists (create if not) 43
touch "$LOG_FILE"
44
# Run the log_data function every 10 seconds
echo "Collecting data... Press Ctrl+C to stop."
47
# Collect data for a period of time (e.g., 1 hour) for testing purposes
for ((i=0; i<3600; i+=10)); do
log_data
sleep 10
done
53
```

Годишник Телекомуникации 2024, том 11, с. 53-60

Yearbook Telecommunications 2024, vol. 11, p. 53-60

eISSN 2534-854X https://telecommunications.nbu.bg/bg/godishnik-telekomunikacii

DOI: https://doi.org/10.33919/YTelecomm.24.11.6

```
          54 # Generate the gnuplot script for creating the graph 55
GNUPLOT_SCRIPT=$(mktemp)
          56
          # Write gnuplot commands to the temporary script
          cat << EOF > "$GNUPLOT_SCRIPT"
          set terminal pngcairo enhanced size 800,600
          set output "$OUTPUT_FILE"
          set title "Network Monitoring Data"
          set xlabel "Time"
          set ylabel "Latency (ms)"
          set y2label "Traffic (MB) / Active Connections"
          set y2tics
          66
          # Format the timestamp on the x-axis
          set xdata time
          set timefmt "%Y-%m-%d %H:%M:%S"
          set format x "%H:%M:%S" 71 set xtics rotate by -45
          72
          73 # Set logarithmic scale for the secondary y-axis (y2) for Traffic and Active Connections 74 set logscale y2
          75
          # Plot latency on the primary y-axis and traffic on the secondary y-axis
          plot "$LOG_FILE" using 1:2 title "Latency (ms)" with lines lw 2, \
          "$LOG_FILE" using 1:3 axes x1y2 title "Incoming Traffic (MB)" with lines lw 2, \
          "$LOG_FILE" using 1:4 axes x1y2 title "Outgoing Traffic (MB)" with lines lw 2, \
          "$LOG_FILE" using 1:5 axes x1y2 title "Active Connections" with lines lw 2 81 EOF
          82
          83 # Run the gnuplot script to generate the PNG file 84
gnuplot "$GNUPLOT_SCRIPT"
          85
          # Remove the temporary gnuplot script
          rm -f "$GNUPLOT_SCRIPT"
          88
          # Inform the user about the output
          echo "Graph has been saved as $OUTPUT_FILE"
```

2.2.Script 2: GNU.sh

The second script, GNU.sh, processes the log data generated by scritpt.sh and creates a graphical representation of the network performance metrics using gnuplot. This script is designed to simplify the visualization of data for system administrators, enabling them to quickly analyze trends and detect potential issues. Full text of this script is shown on Fig. 2.

2.2.1. Key Functions of GNU.sh

The `GNU.sh` script is designed to streamline the process of visualizing network data by leveraging the powerful graphing capabilities of `gnuplot`. Below are its primary functions:

1. Validate Log File

Before proceeding with any operations, the script verifies the existence of the GNU.txt log file, which contains the necessary network performance data. If the file is missing or inaccessible, the script immediately alerts the user and halts further execution. This validation step prevents errors and ensures that the graphing process is based on valid input data.

2. Generate Graph Using gnuplot

The script dynamically creates a temporary gnuplot script to define the parameters required for graph generation. Title and Labels:

Customizes the graph's title, x-axis, and y-axis labels to enhance interpretability. Axis Configuration: Configures scaling and ranges to accommodate the logged data effectively. Plots multiple metrics, such as:

- Latency (measured in milliseconds).
- Incoming and Outgoing Traffic (converted to megabytes).
- Active Connections (number of established sessions).

Output Format: Saves the generated graph as a PNG file, ensuring it is easily viewable and shareable. The automation of these steps simplifies the visualization process, making it accessible even to users with minimal gnuplot experience.

3. Cleanup and User Notification

After successfully generating the graph, the script performs a cleanup to remove the temporary gnuplot script used during the process. This ensures that no residual files clutter the system. Additionally, the user is notified of the successful graph creation, including details about the output file location. This final step enhances usability by confirming task completion and facilitating access to the results.

2.2.2. Code for GNU.sh

```
#!/bin/bash
2
# Define the input and output files
LOG_FILE="./GNU.txt"
OUTPUT_FILE="gnu.png"
6
# Check if the log file exists
if [ ! -f "$LOG_FILE" ]; then
echo "Log file $LOG_FILE not found. Please ensure the log data has been collected."
exit 1
fi
12
# Create the gnuplot script 14 GNUPLOT_SCRIPT=$(mktemp)
15
# Write gnuplot commands to the temporary script
cat << EOF > "$GNUPLOT_SCRIPT"
set terminal pngcairo enhanced size 800,600
set output "$OUTPUT_FILE"
set title "Network Monitoring Data"
set xlabel "Time"
set ylabel "Latency (ms)"
set y2label "Traffic (MB) / Active Connections"
```

eISSN 2534-854X https://telecommunications.nbu.bg/bg/godishnik-telekomunikacii
DOI: https://doi.org/10.33919/YTelecomm.24.11.6

```
set y2tics
25
# Format the timestamp on the x-axis
set xdata time
set timefmt "%Y-%m-%d %H:%M:%S"
set format x "%H:%M:%S" 30 set xtics rotate by -45
31
# Set logarithmic scale for the secondary y-axis (y2) for Traffic and Active Connections 33 set logscale y2
34
# Plot latency on the primary y-axis and traffic on the secondary y-axis
plot "$LOG_FILE" using 1:2 title "Latency (ms)" with lines lw 2, \
"$LOG_FILE" using 1:3 axes x1y2 title "Incoming Traffic (MB)" with lines lw 2, \
"$LOG_FILE" using 1:4 axes x1y2 title "Outgoing Traffic (MB)" with lines lw 2, \
"$LOG_FILE" using 1:5 axes x1y2 title "Active Connections" with lines lw 2 40 EOF
41
42 # Run the gnuplot script to generate the PNG file 43 gnuplot "$GNUPLOT_SCRIPT"
44
# Remove the temporary gnuplot script
rm -f "$GNUPLOT_SCRIPT"
47
# Inform the user about the output
echo "Graph has been saved as $OUTPUT_FILE"
```

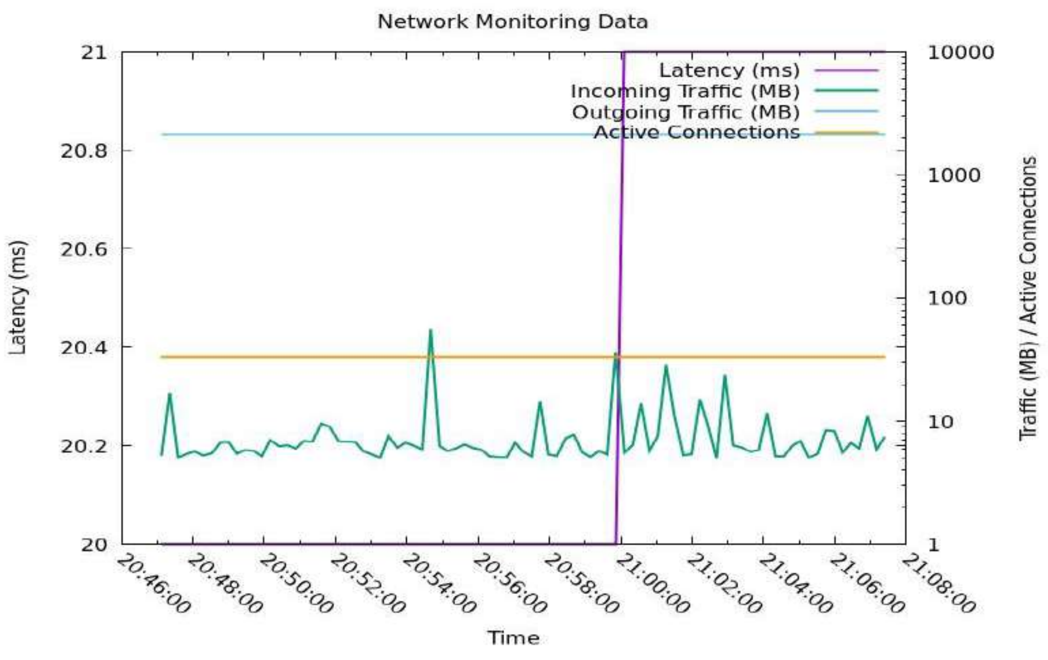

Figure 2. Graph generated from the data collected by scritpt.sh and processed by GNU.sh.

3. METHOD

The methodology of the automated network monitoring system involves two key processes: data collection and data visualization. The scritpt.sh script automates the collection of network metrics at regular intervals, including latency (measured via ping), traffic (monitored using vnstat and parsed with jq), and active connections (counted using netstat). The collected data is logged into a timestamped file (GNU.txt) for further analysis. The GNU.sh script then processes this log file and generates a graphical representation of the data using gnuplot. The graph displays network latency on the primary y-axis and traffic and active connections on a secondary logarithmic y-axis. This automated system ensures continuous network monitoring and provides clear, actionable insights, helping administrators identify trends and network issues efficiently.

4. CONCLUSION

The automated network monitoring system, composed of the two scripts scritpt.sh and GNU.sh, provides a comprehensive solution for tracking and analyzing network performance in Linux environments. By automating the collection of key metrics such as latency, traffic, and active connections, the system enables real-time monitoring without requiring constant manual input. The structured logging mechanism ensures that data is stored in an easily interpretable format, while the integration of gnuplot transforms this raw data into actionable insights through visual representations. This system is particularly valuable for system administrators and IT professionals who need to maintain optimal network performance, identify potential issues, and streamline troubleshooting processes. Its modular design, with separate scripts for data collection and visualization, ensures flexibility and ease of use. By leveraging tools like vnstat, jq, and gnuplot, the solution demonstrates the power of automation in simplifying complex tasks, reducing manual effort, and enhancing the overall efficiency of network monitoring practices.

ИЗПОЛЗВАНИ ИЗТОЧНИЦИ (REFERENCES)

[1] KLEINROCK, L. An early history of the internet: History of Communications. *IEEE Communications Magazine* [online]. 2010, vol. 48(8), pp. 26-36 [viewed 24.03.2025]. ISSN 1558-1896. IEEE Xplore. Available from: 10.1109/MCOM.2010.5534584
[2] ПЕТРОВ, Георги, Филип АНДОНОВ и Тодор ДАЧЕВ. Разработка на приложения с отворени хардуерни платформи. София: Авангард Прима, 2015. ISBN 978-619-160-506-4. [PETROV, Georgi, Filip ANDONOV i Todor DACHEV. *Razrabotka na prilozhenia s otvoreni harduerni platformi*. Sofia: Avangard Prima, 2015. ISBN 978-619-160-506-4.]
[3] ПЕТРОВ, Георги и Иван БОГОМИЛОВ. Приложения за мрежов мониторинг и отстраняване на проблеми в Linux. *Годишник Телекомуникации* [онлайн]. 2017, (4), с. 1-7 [прегледан 24.03.2025]. eISSN 2534-854X. CEEOL. Достъпен на: https://www.ceeol.com/search/article-detail?id=784150 [PETROV, Georgi i Ivan BOGOMILOV. Prilozhenia za mrezhov monitoring i otstranyavane na problemi v Linux. *Godishnik Telekomunikatsii* [onlayn]. 2017, (4), s. 1-7 [pregledan 24.03.2025]. eISSN 2534-854X. CEEOL. Dostapen na: https://www.ceeol.com/search/article-detail?id=784150]

SYSTEM COMMANDS

[1] Ping (8) - Linux manual page. *Linux/UNIX system programming training* [online]. [viewed 24.03.2025]. Available from: https://man7.org/linux/man-pages/man8/ping.8.html
[2] VnStat. *Humdi.net* [online]. [viewed 24.03.2025]. Available from: https://humdi.net/vnstat/
[3] Netstat (8) - Linux manual page. *Linux/UNIX system programming training* [online]. [viewed 24.03.2025]. Available from: https://man7.org/linux/man-pages/man8/netstat.8.html
[4] Jq: Command-line JSON processor. *Jqlang.org* [online]. [viewed 24.03.2025]. Available from: https://stedolan.github.io/jq/
[5] Gnuplot: A command-line driven graphing utility. *Gnuplot.info* [online]. [viewed 24.03.2025]. Available from: http://www.gnuplot.info/
[6] Advanced Bash-Scripting Guide. *Tldp.org* [online]. [viewed 24.03.2025]. Available from: https://www.tldp.org/LDP/abs/html/

Contacts:
Stanislav Georgiev Peltekov, MS, New Bulgarian University, department "Telecommunications", peltekov.stanislav@gmail.com, +359 88 724 1552

Date of receipt of the manuscript: 28.09.2024
Date of adoption for publication: 30.09.2024