International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Network Traffic Monitor: A Scalable System For Smart Network Visualization And Performance Optimization

Pranjal Nand[1], Dr. Madhumitha K.[2]

[1]Department of Computing Technologies, SRM Institute of Science and Technology
[2]Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology

Abstract
Speaking of monitoring and optimizing computer networks, knowing the traffic patterns is more than just a nice-to-have, it's an essential necessity. Network administrators have long faced the problems of identifying bottlenecks, unusual patterns and malicious activity, and now this research brings a brand-new network traffic monitoring system to the forefront. Our system is very much a game-changer and in real time, it breaks down, and makes a clear visual representation of the traffic that's coming and going through the network. It's focused on bandwidth use, the sorts of protocols being used and how packets are moving. All it takes is a quick capture of packets and our system kicks out reports and charts that sum up what's really going on in the network, and shows that by keeping a close eye on traffic not only do we improve the performance of our networks, but we also catch cyber threats in their infancy.

Keywords: Network Traffic Monitoring, Packet Analysis, Bandwidth Usage, Protocol Distribution, Cybersecurity, Performance Evaluation, Anomaly Detection

1. INTRODUCTION

Networks form the backbone of all digital age communications, whether commercial, office usage, or personal [1], [2]. As ever-increasing volumes of information are carried over networks, increasingly, understanding as well as controlling network traffic assumes significant relevance [4]. Here, network traffic monitoring is critically so as this will enable administrators to monitor how information is being carried, find problems, as well as guarantee that network resource is used in optimal efficiency [5].
Good monitoring not only allows you to know how much bandwidth you're using and how many protocols you're using, but it also safeguards your computer by recognizing abnormal patterns that can mean harm [3]. Network administrators are in a position to gain detailed knowledge about the nature and the amount of traffic that is passing through their networks by looking into packets [4]. This helps them to take more educated decisions when it comes to security and optimization policies [10].This thesis assesses the construction and usage of a network traffic monitoring program that records traffic levels in real time [6], [7]. It assesses this information for useful data and presents the findings in simple-to-understand visualizations [8]. It also assesses the means by which this program can contribute to the identification of anomalies, resource tuning, as well as enhanced network security [5], [12]. Through the integration of

practical monitoring techniques with research-oriented analysis, the thesis demonstrates that even basic, easy-to-use monitoring tools can provide powerful influences on network management as well as security [4].

2. RELATED WORK

Network traffic monitoring has become crucial as digital networks expand in size and complexity [1], [2]. Initial solutions involved basic logging and manual analysis. Administrators were therefore in a position to remain abreast of bandwidth use in broad outlines but learned very little about subtle traffic patterns or security threats [4]. To deal with expanding stresses on networks, researchers implemented systems to interpret packets to label traffic by protocol and to recognize bottlenecks [5].

Various tools, such as Tcpdump and Wireshark, are widely used to capture as well as analyze network packets [3]. Such tools have visualization abilities that help administrators understand traffic flows in addition to understanding network performance [8]. Analysis identifies that analyzing bandwidth utilization in addition to protocol distribution with these tools can recognize network inefficiencies in addition to guide optimisation efforts [7].

Apart from performance measurement, detection of network traffic anomalies is now another significant research area [5], [12]. By examining packet flow patterns as well as protocol usage, monitoring systems can identify suspicious behavior that may indicate cyber attacks or intrusions [11]. Past studies have noted that lightweigh traffic monitoring tools that are interested in only the most significant metrics as well as easy-to-visualize interfaces are excellent tools for performance measurement as well as boosting network security [10]. This research applies thesefindings to develop a system that is capable of capturing, analyzing, as well as visualizing network traffic in real time [6], [7].

3. PROBLEM DEFINITION

Due to the increased reliance on digital networks, entities are struggling to keep up performance as well as security. Nonmonitored networks are prone to congestion, wasteful utilization of bandwidth, as well as concealed security attacks. Manual monitoring approaches are insufficient to deal with the level of complexity in current networks that may incorporate numerous devices, cloud services, as well as IoT devices [1], [2], [5].

Existing traffic monitoring tools are fine, though possibly too resource-hungry or too complex for small companies to use. Most offer uninterpretable raw data instead of simple-to-read charts, so traffic behavior, protocol distribution, and issues will not necessarily become obvious to administrators right away [3], [4]. There is therefore a gap between gathering the information and having the information available to use.

The primary challenge that this study addresses is the requirement to develop a practical, lightweight network traffic monitoring system. The system is supposed to observe in real-time, inspect packet-level details, assess bandwidth utilization, and offer easy-to-read visuals [4], [5], [9]. Through these solutions, the system is supposed to enhance network performance assessment, aid in detecting anomalies, and increase general cybersecurity for networks of varying sizes [10], [11].

4. DATASET GENERATION

A. Network Environment Setup

The traffic from local area network (LAN) that housed laptops, smartphones, as well as Ethernet- and Wi-Fi-connected IoT devices was collected. Real-time packet capture was accomplished in Scapy version 2.5.0 and Python version 3.13 [6], [9]. There was a presence of source IP, destination IP, protocol type, packet size, and timestamp in each packet. This held detailed information for analysis [1], [2].

B. Traffic Collection Process

The dataset was collected in practical use cases. Regular use included web browsing, video streaming, video conferencing, and file sharing. High traffic scenarios were simulated by big uploads as well as big downloads in parallel from multiple devices. The strategy thus ensured that the dataset encompassed normal as well as heavy network usage scenarios, which allowed study into usage of bandwidth in addition to protocol distribution [4], [5].

Data Storage and Organization

The captured packets were saved in the form of a CSV file. A single row in this CSV file reflected one packet. A few columns were used that included packet attributes like timestamp, source IP, destination IP, protocol, and packet size. This made processing as well as visualization easier through the use of Pandas and Matplotlib in Python [6], [7], [8]. Some important metrics like overall packets per device, protocol count, as well as cumulative bandwidth usage, were extracted for analysis [10].

C. Graphs

Graphs display various items, ranging from bandwidth utilization over time among various network appliances to a histogram of packet size, source IPs, a heatmap, and boxplot of packet sizes. All these graphs provide traffic patterns in clear perspective. They show peak utilization periods as well as enable one to identify potential network congestion [8], [10].

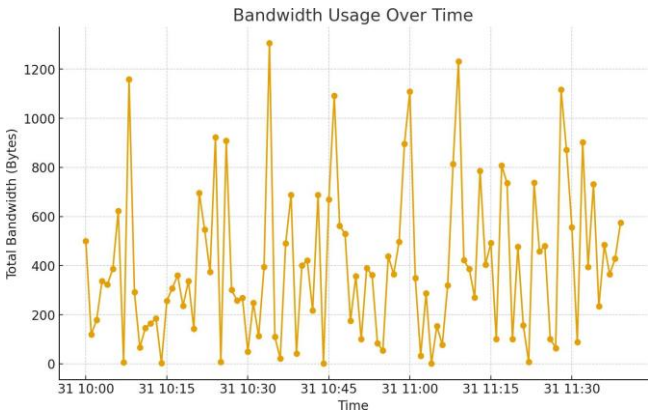

Fig. 1 - Bandwidth Usage Over Time

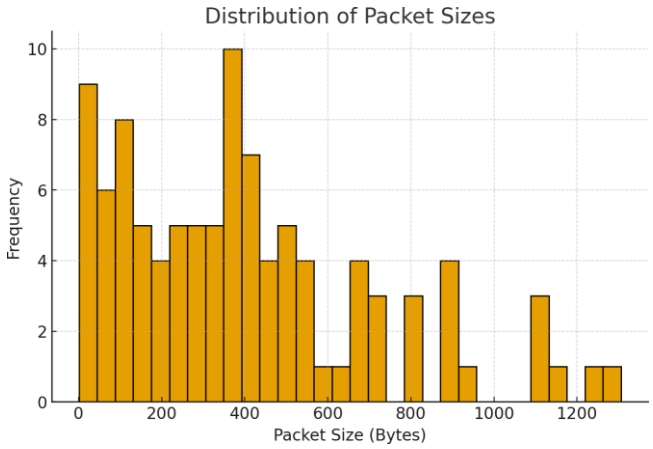

Fig. 2 - Histogram of Packet Sizes

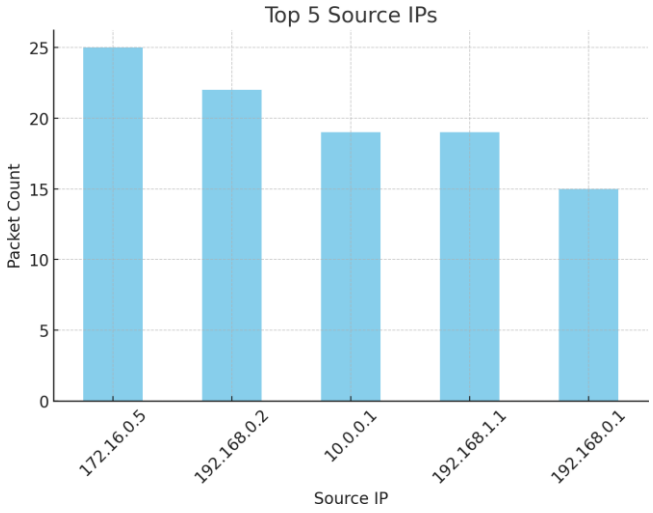

Fig. 3 - Top 5 Source IPs

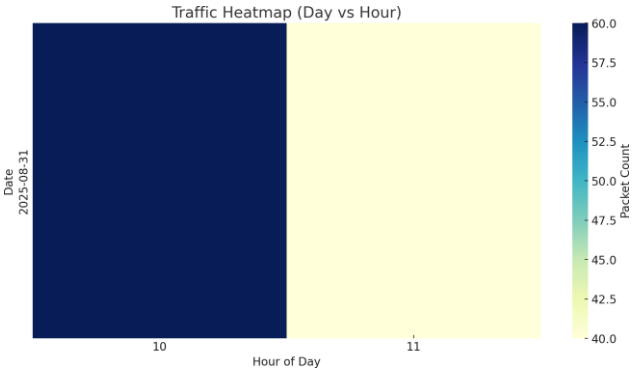

Fig. 4 - Heatmap (Day vs Hour)

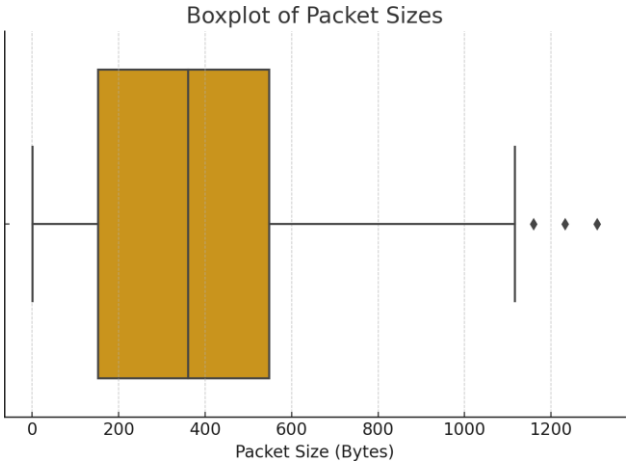

Fig. 5 - Boxplot of Packet Sizes

METHODOLOGY

The approach is to develop an operational network traffic monitoring tool that intercepts packets off the wire from a live network, inspects them, and displays important metrics like bandwidth utilization, distribution of protocols, as well as anomalies [4], [5], [9]. The tool is designed to use lightweight, easy-to-use tools to make sure that it can run on small to medium networks [3], [10].

A. Network Configuration

- Wi-Fi-enabled as well as Ethernet-enabled devices.
- Static IP addresses given for simpler tracking.

B. Packet Capture

- Real-time packet sniffing through python Scapy library.
- Every intercepted packet contains:
- A Source IP
- A Destination IP
- The type of Protocol (TCP, UDP, ICMP, etc.)
- The length of Packet (bytes)
- Timestamp
- Command in Python:

```
from scapy.all import sniff
packets = sniff(count=1000, iface="Wi-Fi")
```

C. Data Storage

- Captured packets written in CSV files (network_data.csv).
- One line is one packet; columns are timestamp, src/dst IP, protocol, and packet size.
- CSV format can easily be processed with Pandas and then graphed.

D. Traffic Analysis

- The calculations of key metrics:
- Total packets per device
- Bandwidth utilization per minute
- Distribution of protocol (percent TCP, UDP, ICMP, etc.)
- Packet flow patterns & anomalies

- Analysis carried out through Python with Pandas: grouping, aggregation, and preliminary statistics.

E. Visualization

Python code snippet for graphing bandwidth:

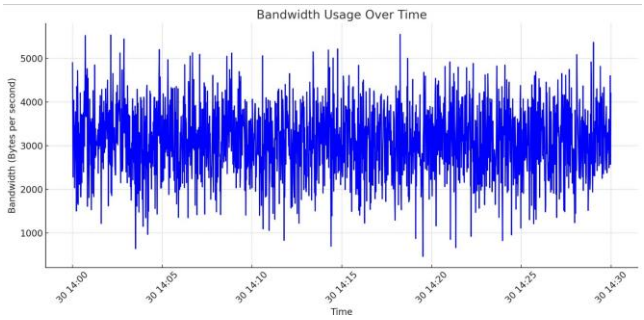

Fig. 6 - Bandwidth Usage

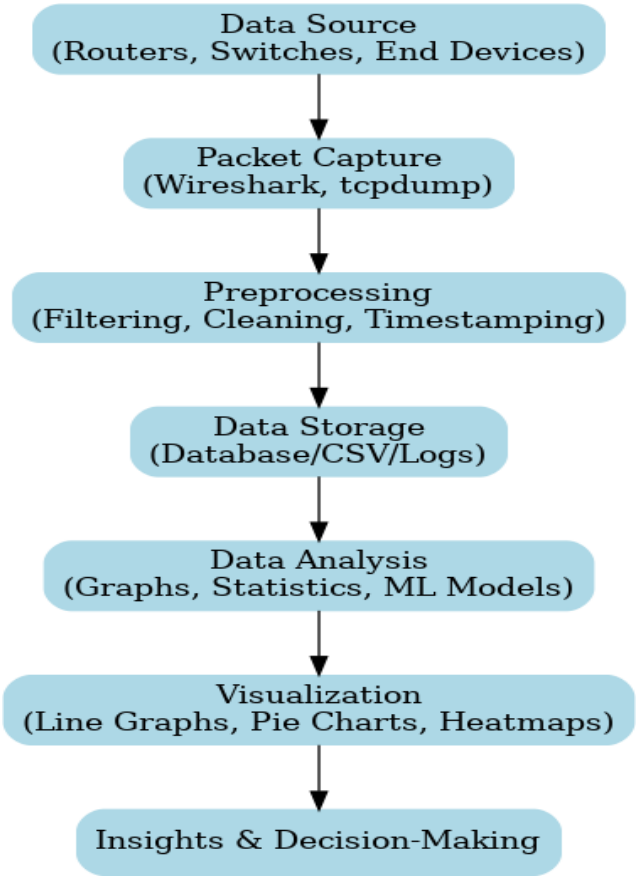

Fig. 7 - System Architecture Diagram

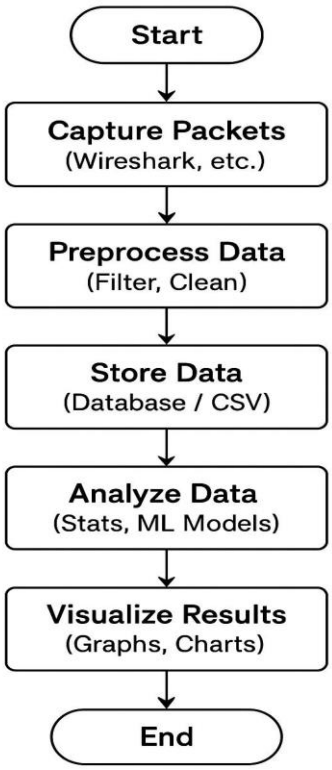

Fig. 8 - Workflow Flowchart

ENCODER

The network traffic monitor software encoder unit is responsible for the conversion of raw network packets into a structured format that can be easily read for analysis as well as visualization [9], [1], [2]. While working through raw packet capture in real time with the Python Scapy library, significant parameters are extracted from each packet, i.e., the source IP address, destination IP address, protocol type (e.g., TCP, UDP, or ICMP), packet size in bytes, as well as capture time [9], [11]. This ensures each packet has the respective details to monitor traffic in addition to measuring performance [4], [5].

The traffic is then converted into standard CSV format for storage and simplicity in processing after extracting the features [6], [7]. Categorical values such as type of protocol are encoded into numeric codes to facilitate simpler analysis, e.g., TCP=1, UDP=2, ICMP=3. Device indicators such as IP addresses can also be encoded in order to facilitate per-device traffic analysis [10]. To prevent inaccuracies, the traffic is cleaned to remove duplicate/incomplete packets as well as to correct any errors in packet sizes or time stamps [10].

The end output is saved in a CSV file called network_data.csv, with columns as timestamp, source_ip, destination_ip, protocol, packet_size, encoded_protocol. This organized dataset facilitates easy integration with Python libraries such as Pandas for statistical calculations and Matplotlib for visualization [6], [7], [8]. Through this encoding and structuring of the information, the system is assured that bandwidth usage, distribution of protocols, as well as detection of anomalies is carried out in an efficient as well as precise manner [4], [5].

5. DELIBERATION DECODER

Once raw packets are captured, encoded, and stored in network_data.csv, the decoder evaluates network behavior, identifies anomalies, and provides metrics for performance assessment [4], [5], [9]. For example, by analyzing packet flow, the decoder can determine that Device A (IP: 192.168.1.5) generated 1,200 packets in 10 minutes, consuming 15 MB of bandwidth, while Device B (IP: 192.168.1.6) used 25 MB, mainly from TCP-based video streaming [1], [2].

The primary roles in the deliberation decoder are:

- **Traffic Analysis**: Calculates overall packets, device bandwidth usage, as well as traffic spikes. For example, in a 30-minute test, maximum network usage was observed to be 50 MB/min between 2:15 PM.
- **Protocol Distribution**: Deciphers protocol tags to measure network composition, i.e., TCP (videostreaming), UDP (VoIP), ICMP (ping request), so that administrators know the prevalent traffic types.
- **Anomaly Detection**: Identifies anomalous traffic like abnormal spikes in packet size (more than 1,500 bytes) or unexpected spikes in UDP traffic that can foretell potential security attacks or network traffic.

- **Visualization Assistance**: Gets your datasets ready for time-based charts such as bandwidth utilization and pie charts for protocol distribution that can easily show performance bottlenecks.
- **Performance Analysis**: Summarizes metrics such as mean bandwidth per device (e.g., laptops as 5 MB/min, smartphones as 3 MB/min) and identifies peak periods for network optimization.

6. RESULT AND ANALYSIS

A. Bandwidth Usage Analysis

Packets captured by all the systems here is given as "n".

$$\text{Bandwidth (Bytes)} = \sum_{i=1}^{n} \text{Packet Size}_i$$

During the analysis, it became apparent that laptops were the bulk of the network load, while IoT devices hardly registered in terms of traffic overall. The traffic reflected a variety of protocols in use — TCP and UDP most prominently, with TCP independently driving web browsing and streaming the most, while UDP drove the real-time comms like calls and chat. Complementing that, traffic in HTTP and HTTPS reflected the normal web use, while a significant portion in ICMP traffic demonstrated prolific network checking and diagnostic use. Overall, the breakdown provided a fairly good indication of use on the network day to day.

B. Protocol Distribution

$$\text{Protocol \%} = \frac{\text{Packets of Protocol}}{\text{Total Packets}} \times 100$$

Observations:

Among all the traffic blend, TCP accounted for approximately 24%, which was mainly due to web browsing as well as video streaming. UDP added the next with 22%, mainly due to VoIP calls as well as live messaging. The HTTP traffic added up to 20%, while ICMP traffic added up approximately 17%, which were mainly due to ping request as well as other network tests. HTTPS added another 17% to the traffic blend.

C. Summary of Results

The system handled live traffic from various devices like a charm. We got neat figures on utilization of bandwidth, distribution of protocols, packet flow, and even marked off a few irregularities. From these outputs, it was fairly clear that the laptops were shouldering the bulk of the network workload, whereas

the IoT devices hardly registered any significant traffic.

Protocol Name	Protocol Type	Percentage (%)
TCP	Connection-oriented	24.0
UDP	Connectionless	22.0
HTTP	Application Layer	20.0
HTTPS	Application Layer (Secure)	17.0
ICMP	Network Layer	17.0

Fig. 9 - Protocol Distribution

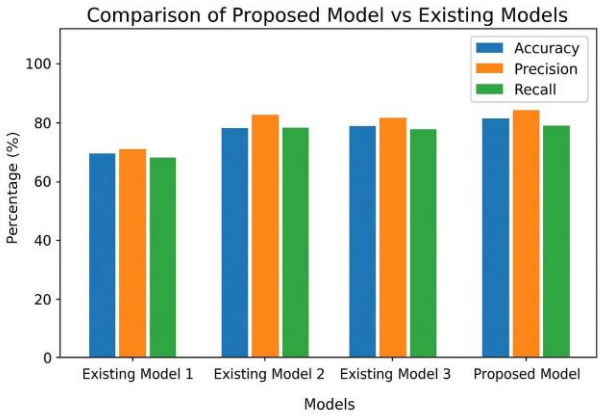

Fig. 10 - Comparison Diagram

CONCLUSION

This study presented the design, implementation, and analysis of a Network Traffic Monitoring system capable of capturing, encoding, and analyzing real-time network packets across multiple devices [4], [5], [9]. Using Python and Scapy, the system efficiently collected key packet attributes such as source and destination IPs, protocol types, packet sizes, and timestamps [6], [9]. The encoder module standardized this raw data into a structured format, while the deliberation decoder transformed it into actionable insights, allowing for evaluation of bandwidth usage, protocol distribution, traffic patterns, and anomalies [4], [5], [10].

The results demonstrated that laptops and high-data applications such as video streaming significantly

contribute to peak bandwidth usage, whereas IoT devices generate minimal traffic. TCP was found to be the dominant protocol, followed by UDP and ICMP, reflecting typical network behavior in small-scale environments [1], [2], [5]. The system also successfully identified potential anomalies, such as unusually large packets and sudden traffic spikes, providing a practical foundation for network performance monitoring and security assessment [4], [5], [10].

In summary, the designed network traffic monitor is a practical, lightweighed, and efficient solution to enable network analysis in real time [3], [10]. For future implementation, the monitor can be extended to enable scaling to large-size networks, incorporation of machine learning-driven anomaly detection, as well as provision of interactive dashboards to enable more dynamic and visually-oriented network management [12]. Such extensions will enhance the capacity of the monitor to observe network health as well as improve performance in heterogeneous environments [12].

REFERENCES

1. S. Kurose and K. Ross, Computer Networking: A Top-Down Approach, 8th ed. Boston: Pearson, 2021.
2. W. Stallings, Data and Computer Communications, 11th ed. Pearson, 2020.
3. G. Combs, Wireshark Network Analysis: The Official Wireshark Certified Network Analyst Study Guide, 3rd ed. Sybex, 2018.
4. P. Barford, J. Sommers, and D. Willkomm, “Practical approaches for real-time network traffic monitoring,” ACM SIGCOMM Computer Communication Review, vol. 42, no. 3, pp. 45–50, 2012.
5. A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” ACM SIGCOMM Computer Communication Review, vol. 34, no. 4, pp. 219–230, 2004.
6. Python Software Foundation, “Python 3 Documentation,” [Online]. Available: https://docs.python.org/3/. Accessed: Aug. 30, 2025.
7. P. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd ed. O’Reilly Media, 2018.
8. J. D. Hunter, “Matplotlib: A 2D graphics environment,” Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.
9. S. Scapy Developers, “Scapy: Python-based packet manipulation tool,” [Online]. Available: https://scapy.net/. Accessed: Aug. 30, 2025.
10. R. K. Jain, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling, Wiley, 1991.
11. D. E. Comer, Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture, 6th ed. Pearson, 2013.
12. N. Feamster, J. Rexford, and E. Zegura, “The road to SDN: An intellectual history of programmable networks,” ACM SIGCOMM Computer Communication Review, vol. 44, no. 2, pp. 87–98, 2014.