engineering proceedings

MDPI

Proceeding Paper

Architecture of Cross-Platform Distributed Hierarchical System for Real-Time Network Monitoring †

Veneta Aleksieva * and Hristo Valchanov

Computer Science and Engineering Department, Technical University of Varna, 9010 Varna, Bulgaria; hristo@tu-varna.bg

* Correspondence: valeksieva@tu-varna.bg

† Presented at the International Conference on Electronics, Engineering Physics and Earth Science (EEPES2026), Bandirma, Turkey, 24–27 June 2026.

Abstract

In today's digital infrastructure, organizations and enterprises increasingly rely on distributed and remote IT environments. These environments may include multiple network sites, remote branch offices, employee home networks, or servers hosted in geographically distant locations. Monitoring and managing these networks presents a critical challenge, especially when administrators are expected to maintain security, performance, and uptime across all nodes. This paper presents an architecture for a cross-platform, multi-tenant network monitoring and management system that enables centralized visibility and control over a distributed infrastructure.

Keywords: cross-platform; network monitoring; real-time; SNMP

1. Introduction

In today's hyper-connected digital world, the number of devices connected to local and global networks continues to grow at an unprecedented rate. According to Statista [1] there were over 19.8 billion connected Internet of Things (IoT) devices worldwide in 2025, and this number is expected to grow to over 40 billion by 2034. This explosive growth in connected devices brings not only convenience and automation, but also new challenges in cybersecurity, network performance, and administrative oversight. Many organizations—especially educational institutions, small businesses, public sector agencies, and even private home networks—lack affordable, easy-to-use tools to effectively monitor, manage, and secure their local networks. While large enterprises have access to sophisticated and expensive network monitoring systems like Zabbix, SolarWinds, or PRTG, smaller environments are often left with manual methods or outdated tools that fail to provide real-time information.

At the same time, cybersecurity threats are becoming more complex and frequent. IBM's Cost of a Data Breach (2023) report [2] notes that the average time to detect and contain a breach was 241 days, with an average cost of $4.44 million per incident worldwide. A common vector for these attacks [3] is through poorly secured devices or unmonitored DNS traffic, which allows malicious actors to infiltrate data or inject malware without being noticed. DNS queries, SNMP-based device communication, and router-level traffic provide some of the most important low-level data for understanding what is happening on the network. Passive DNS logging can reveal access to malicious domains, SNMP monitoring can help identify unexpected spikes in bandwidth usage, and automated device discovery

Check for updates

Academic Editors: Teodor Iliev, Ivaylo Stoyanov, Grigor Mihaylov and Selahattin Kosunalp

Published: 2 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

can alert administrators to unknown devices appearing on the network. These capabilities are essential for maintaining network visibility and control.

Furthermore, the rise of decentralized networks—such as organizations with multiple physical locations (branch offices, schools, clinics, etc.)—creates the need for a system that can operate independently locally and synchronize critical data with a central management server. This hybrid model provides continuity and decentralization while allowing centralized policy enforcement, updates, and visibility across sites.

In the field of network monitoring, dozens of approaches and tools have been developed over the years, often targeting enterprise or data center environments. While many of them are technically capable, most are not sufficient in the context of small-to-medium-sized deployments, especially when the use case involves site-level monitoring for distributed users. For example, Zabbix and Nagios [4] focus on enterprise monitoring and infrastructure notifications. They provide agent-based or agentless monitoring with SNMP support. However, they are also complex to implement, with an outdated interface and complex static configuration. LibreNMS [5] provides SNMP-based monitoring, but is designed for use with a central network OS. Netdata [6] uses real-time metrics with useful graphs, but is oriented to a single host without the ability to have a centralized dashboard for site management. GrayLog [7] focuses on aggregation of log files with subsequent detailed analysis, but is not designed for DNS/traffic monitoring or real-time site tracking. There is also research and development using multi-agent systems [8] for the purpose of flexible monitoring; platforms for monitoring and incident detection using machine learning [9,10]; and use of proxy server architectures [11]. The characteristic is that there is an emphasis on single-platform implementations, mainly data collection via SNMP, lack of possibility of secure remote connections of administrators to monitoring systems.

This paper presents the architecture of a cross-platform, multi-tenant network monitoring and management system that enables centralized visibility and control over a distributed infrastructure.

2. Prerequisites for a New Solution

In today's heterogeneous networks, the challenge is no longer just managing a single static network, but providing security, performance, and visibility across multiple dynamic and often geographically dispersed nodes. These issues are compounded by:

Lack of centralized monitoring

Remote sites often operate independently, making it difficult to track the health of their network. Administrators are blind to bandwidth abuse, DNS activity, uptime, or even the number of active devices at a given site unless they manually connect to each environment.

Fragmented tools and inconsistent interfaces

Existing tools such as SNMP platforms, log aggregators, and device discovery tools offer partial solutions, but require significant training, complex setup, and lack of integration with each other.

Remote access security

Providing access to internal devices (routers, switches, servers) at a remote site typically requires opening firewall ports, risking exposure. While VPNs exist, they are not directly integrated into the monitoring layer and require manual configuration for each site.

No scalable multi-tenancy architecture

Most available solutions are either designed for single-site deployments or do not offer a true multi-tenancy experience. There is little or no hierarchical role control, making managing large environments cumbersome.

No visibility into end-user behavior

https://doi.org/10.3390/engproc2026154025

Especially in shared or campus environments, it is important to see which websites or services are being queried. DNS queries can provide valuable information about potential threats, misconfigurations, or misuse of network resources. In most cases, DNS sniffing and device correlation are rarely performed in real time.

Complexity of deployment

Many existing solutions assume prior experience in Linux administration and a deep understanding of network protocols. A simpler, more intuitive interface is needed that abstracts this complexity, allowing administrators to deploy and use the system with minimal configuration.

No notifications or site lifecycle management

If a remote site goes offline, gets compromised, or quietly stops reporting data, administrators often won't know unless they check manually. A real system should support heartbed, alerts, and site health checks.

3. Architecture of the Proposed System

The architecture of the proposed system is designed to support modular deployment, seamless synchronization between distributed nodes, and a clear separation of responsibilities between server logic, data storage, and user interface. The user interface is web-based, allowing administrators to access the system in a unified manner.

The system consists of two main layers (Figure 1):

- Central Control Server (CCS)—this component performs centralized control by managing sites, global device data, users, and receiving updates from Local Site Servers (LSS). The information is stored in a central SQLite database.
- Local Site Server—a component that is installed at a given site on an edge router running Linux or Windows Server. It performs monitoring and synchronization tasks with the central server.

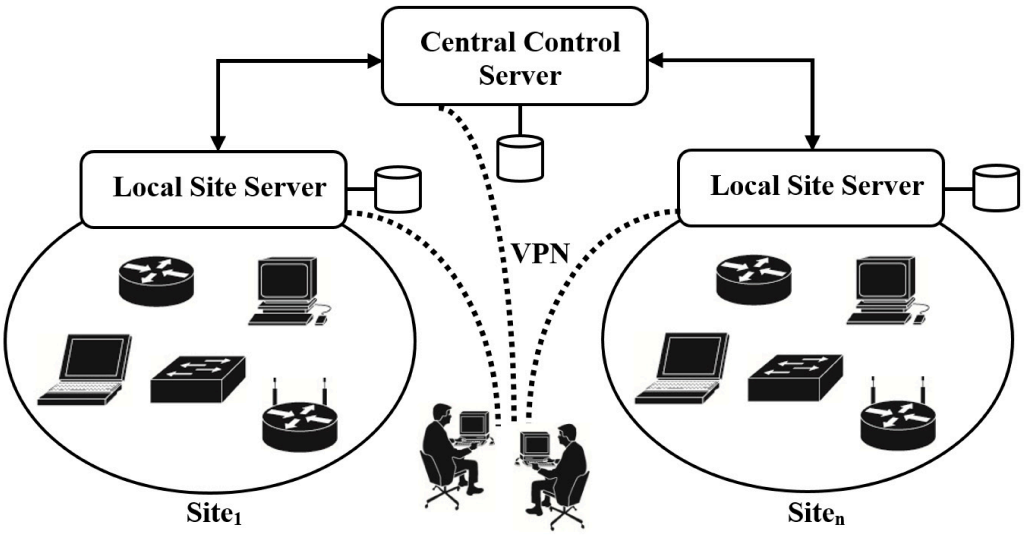


Figure 1. Architecture of the proposed system.

Site

A site represents a logical or physical entity, such as a branch, a customer deployment, or a local network domain. Each site has a unique *site_id*, a descriptive *site_name*, and metadata such as *owner_email*, *created_by*, timestamps such as *created_at*, *last_seen*, etc. All users, devices, servers, and log files are encompassed by a site. By isolating data, boundaries are enforced between customer environments, ensuring that operations and data visibility are only accessible within their respective contexts.

https://doi.org/10.3390/engproc2026154025

The conceptual framework of the proposed system allows a site to have multiple users and devices, but a single user or device is strictly tied to only one site. The differentiation of users and devices by site is based on the site_id key in the CCS database. In the local LSS database, users and devices are differentiated by their id keys.

Devices

Devices include network infrastructure components such as hosts, switches, and routers. Each device is identified by its IP address, MAC address, and name. Each device belongs to a single site, allowing for visualizations and audit trails across the entire site. The system ensures uniqueness and consistency by reconciling records using both MAC and IP addresses at the time of discovery.

Local Site Server

The Local Site Server is responsible for performing tasks such as DNS sniffing, logging SNMP traffic, and heartbeat reporting. It communicates with the central server at regular intervals to send updates and receive configuration changes. Each local server is assigned to a single site. The information is stored in a local SQLite database.

The LSS includes several main modules:

- Local Server Agent—acts as a background service and local data collector. It periodically sends heartbeat updates and synchronization to the central server using background threads. It also receives updated site information or policies from the administrator side. The communication is based on the Push/Pull synchronization model.
- DNS sniffer—performs the interception of DNS requests and their registration. These DNS requests are normalized and registered in the database along with metadata such as the IP address of the request, time and site ID. This allows visualization of searches and trends in domain access by user, device or time window.
- SNMP Agent—performs device discovery in the site, traffic logging, measurement of incoming and outgoing data on router interfaces and monitoring of bandwidth consumption. Differences between leak detection values are calculated and historical usage trends are displayed on a visual chart.
- TrafficLog—stores historical bandwidth usage information obtained from routers via SNMP. Each log entry includes raw input/output counters, calculated deltas, and accumulated traffic totals. Provides time series visualizations that track upload/download trends at the site level. Logs are generated at regular intervals by a dedicated thread in the local server agent.

Administrators use management software to configure and monitor the system status. When certain events occur, the LSS modules generate alerts that are sent by messages to the management software and visualized by the administrator. Alert sources differ according to their origin. The Central Control Center sends an alert when a site is down (lack of heartbeat from the Local Server agent of a site). The SNMP Agent sends alerts in two cases (i) when a new device is discovered and (ii) when a message is received via the SNMP TRAP protocol from a device in the site that a certain threshold has been crossed. Alert thresholds themselves are not configurable. A threshold itself is configured by the administrator via the SNMP protocol.

LSS periodically synchronizes its database with that of CCS using timestamps. If the connection between them is lost, LSS continues to store data in its database, operating in an autonomous mode. When the connection is restored, LSS starts synchronization and sends the data received from the moment of the breakdown of communication with CSS.

https://doi.org/10.3390/engproc2026154025

4. Experimental Study and Results

Experimental studies of the functionality of the proposed system were conducted on a virtual platform under VMware 6.5. A virtual infrastructure with three sites with three hosts each running under the Linux operating system was implemented. The two Local Site Servers are under Linux, and the third under Windows Server 2012. The virtual infrastructure has a connection to the Internet, which allows for monitoring and analysis of real DNS requests.

After starting the system servers, they register with the Central Control Server (Figure 2), after which the SNMP agents send messages to discover devices in the corresponding site. The received information is visualized by the administrator (Figure 3).

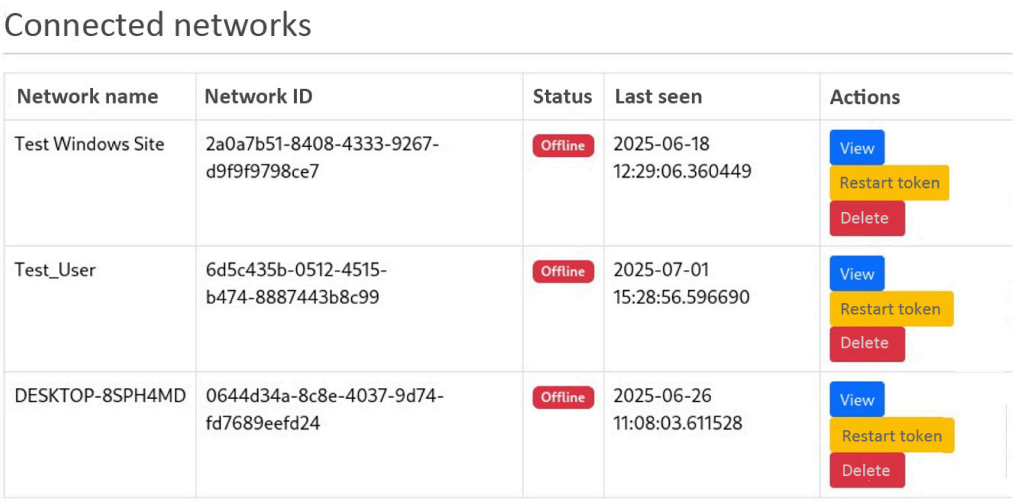


Figure 2. Registered sites.

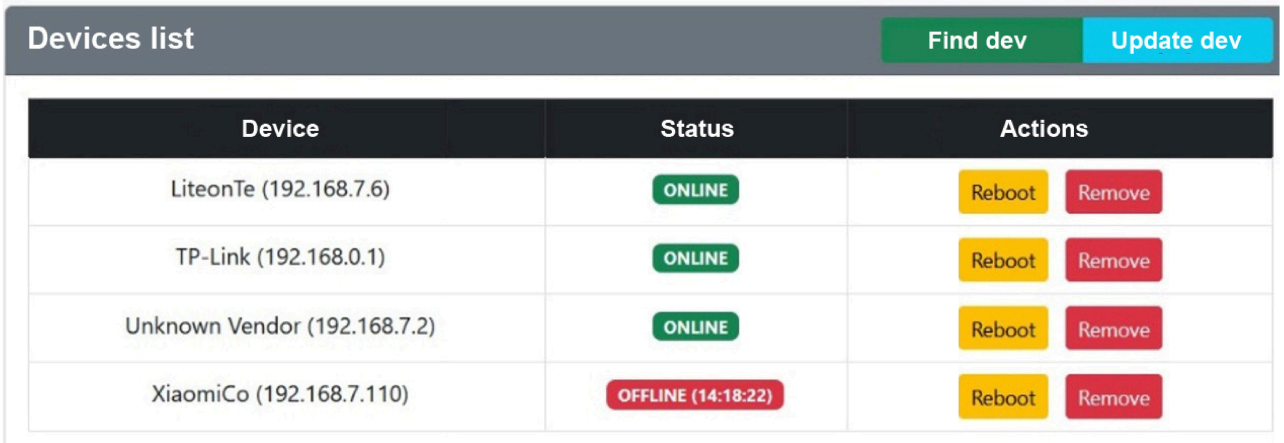


Figure 3. Recognized devices.

The system monitors incoming and outgoing traffic for each site, and the data received from the TraficLog module is visualized in graphical form, with the possibility of viewing in a specific time window as shown in Figure 4.

To test the capabilities of monitoring DNS traffic, requests are sent from hosts to the DNS system on the Internet. The results obtained provide information about the accessed domains in both tabular and graphical form (Figure 5).

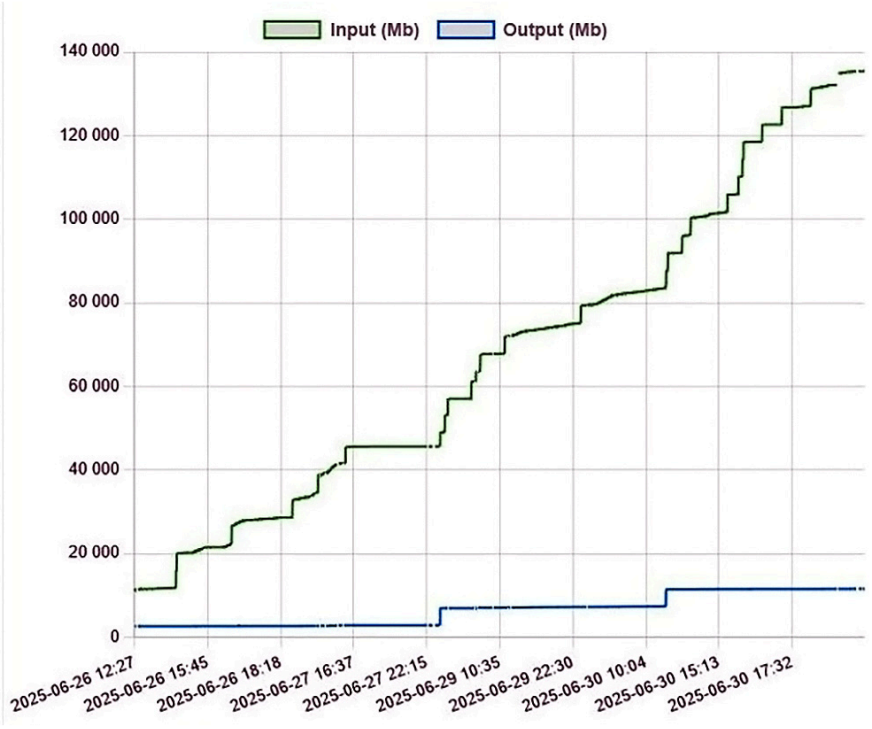


Figure 4. Ingress and egress site traffic.

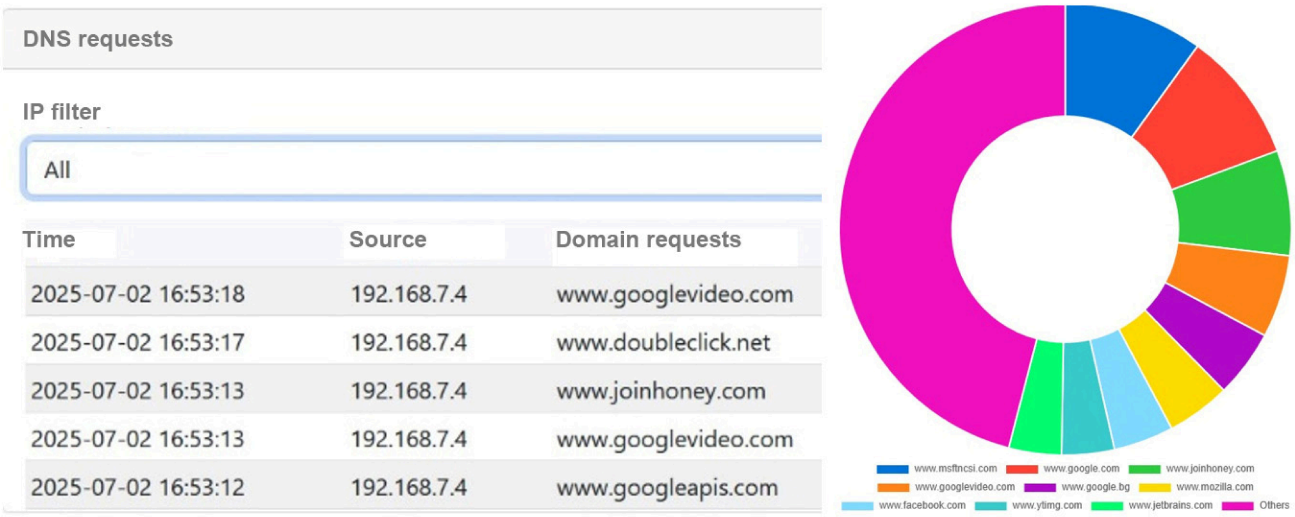


Figure 5. Monitoring of DNS traffic.

5. Conclusions

This paper presents the architecture of a cross-platform system for secure, distributed network monitoring using a hybrid client-server architecture. The system allows both local autonomy and centralized control. It allows administrators to monitor multiple sites, track traffic via SNMP, log DNS queries, and manage connected devices in real time.

The architecture emphasizes modularity, with key background services such as DNS sniffing and SNMP logging running in isolated threads. The synchronization mechanism ensures that local servers can securely send data to and retrieve updates from the master server via secure tunneling.

The main strengths of the system are its extensibility, offline capabilities with delayed synchronization, and an easy-to-use interface for both administrators and local users. It

combines real-time monitoring with light system resource usage, making it suitable even for embedded or low-resource environments.

The authors envisage conducting future experimental studies that include detailed evaluations of the proposed system by implementing different scenarios to evaluate key metrics, such as CPU load, RAM usage, message latency and throughput. While the current work presents a conceptual validation of the proposed architecture through a virtual infrastructure, experimental studies will be performed with a real infrastructure to obtain adequate results. At the same time, this will also allow for more adequate testing for errors and boundary conditions.

Future work could include expanding protocol support beyond DNS and SNMP, implementing alerting and anomaly detection rules using machine learning, and improving the user interface to offer more customization for different roles. The system could also be adapted to work in industrial or IoT environments where real-time telemetry and secure site isolation are critical.

Author Contributions: Conceptualization, H.V. and V.A.; methodology, V.A.; software, H.V.; validation, H.V. and V.A.; investigation, H.V. and V.A.; data curation, H.V. and V.A.; writing—original draft preparation H.V.; writing—review and editing, H.V. and V.A.; visualization, H.V. and V.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available upon reasonable request to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
SNMP	Simple Network Management Protocol
CCS	Central Control Server
LSS	Local Site Servers

References

1. Number of IoT Connections Worldwide. Available online: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/?srsltid=AfmBOor4uRXQxIORlXszZmixVVM1JOqfJMuKMrzVfEcl7iL6ooXeXoay (accessed on 25 March 2026).
2. 2025 Cost of a Data Breach Report. Available online: https://www.ibm.com/think/x-force/2025-cost-of-a-data-breach-navigating-ai (accessed on 25 March 2026).
3. Ozkan-Okay, M.; Samet, R.; Aslan, Ö.; Kosunalp, S.; Iliev, T.; Stoyanov, I. A Novel Feature Selection Approach to Classify Intrusion Attacks in Network Communications. *Appl. Sci.* **2023**, *13*, 11067. [CrossRef]
4. Fernandes, H. Monitoring Hybrid Enterprise Infrastructure Using Nagios and Zabbix While Supporting Salesforce AI-Powered Customer Journeys. *Int. J. Sci. Eng. Technol.* **2022**, *10*, 1–7. [CrossRef]
5. Fachrurroi, N.; Wirabudi, A.; Rozano, S. Design of network monitoring system based on LibreNMS using Line Notify, Telegram, and Email notification. *SINERGI* **2023**, *27*, 111–122. [CrossRef]
6. Netdata. Available online: https://www.netdata.cloud (accessed on 25 March 2026).
7. Graylog. Available online: https://graylog.org (accessed on 25 March 2026).
8. Villalobos, R.; Triana, E.; Ceballos, H.; Triviño, J. Design and Implementation of Network Monitoring System for Campus Infrastructure Using Software Agents. *Ing. E Investig.* **2021**, *42*, e87564. [CrossRef]
9. Huang, W. Research on Computer Network Monitoring System Based on Artificial Intelligence. In Proceedings of the International Symposium on Intelligent Robotics and Systems (ISoIRS), Chengdu, China, 13–15 June 2025. [CrossRef]

10. Bellamkonda, S. Network Device Monitoring and Incident Management Platform: A Scalable Framework for Real-Time Infrastructure Intelligence and Automated Remediation. *Int. J. Recent Innov. Trends Comput. Commun.* **2022**, *10*, 76–86.
11. Melath, N.; Rachabattuni, R.; Cilamkoti, S.S.; Srinivas, D.; Chadaram, H.; Udayan, J. Robust and Scalable Network Monitoring System Using Apache Spark. In *Proceedings of 3rd International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*; Gunjan, V.K., Zurada, J.M., Eds.; Lecture Notes in Networks and Systems; Springer: Singapore, 2023; Volume 540. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.