Design and Experimental Evaluation of Secure Overlay Networks for Docker Microservices Architectures

Damiano Perri[1(✉)], Osvaldo Gervasi[1], and Francesco Bietolini[2]

[1] Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy
{damiano.perri,osvaldo.gervasi}@unipg.it
[2] Perugia, Italy

Abstract. Overlay networks are today a key technology for enabling communication across distributed microservices in containerised environments. However, in standard Docker Swarm deployments, these networks offer logical isolation but lack default encryption, leaving east-west traffic vulnerable and highlighting the challenge of balancing security with efficient connectivity. This work addresses this challenge by systematically evaluating secure and insecure overlay network configurations in a controlled three-node Docker Swarm cluster. We compare four scenarios: a physical network, a standard VXLAN overlay, an IPsec-encrypted overlay, and WireGuard tunnelling for secure inter-node communication. The analysis considers latency, TCP and UDP throughput, jitter, and traffic observability via Wireshark captures. The experimental results highlight that native overlay encryption introduces a measurable overhead compared to the physical baseline, whilst WireGuard provides broader host-level protection with a different performance impact profile. In particular, significant reductions in TCP throughput and variable increases in latency are observed, depending on the solution adopted. The results highlight the trade-off between security and performance in containerised architectures and show that the WireGuard implementation yields greater benefits than other configurations, making it the most suitable choice for this project.

Keywords: Docker · Cloud Computing · Cybersecurity · Networking · Microservices

1 Introduction

Container-based microservices architectures are now the most widespread, significant and dominant [1]. The use of orchestration platforms such as Docker Swarm, in our case, allows services to be deployed across different nodes, ensuring scalability, isolation and operational flexibility; in such environments, communication between distributed microservices typically takes place via overlay

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16767, pp. 236–250, 2027.
https://doi.org/10.1007/978-3-032-30491-9_16

networks, which create a logical topology superimposed on the physical infrastructure. Overlay networks, which are generally based on encapsulation technologies using the VXLAN protocol, simplify the management of connectivity between distributed containers by providing isolation. However, in the standard configuration, they do not guarantee the encryption of application traffic, and this is particularly critical in the context of so-called east-west traffic—that is, internal communication between services within the same cluster—as it may carry sensitive information or application control data [2]. The evolution of Zero Trust models and the focus on the security of internal communications have highlighted how east-west traffic constitutes a significant attack surface in distributed architectures. In the absence of adequate protection mechanisms, an attacker could gain access to the physical network or a compromised node and consequently intercept, analyse or manipulate traffic between microservices, which is a critical risk in domains requiring high data integrity and confidentiality, such as secure electronic assessment systems [3]. Docker Swarm offers native IPsec-based overlay network encryption, which can be enabled using the –opt encrypted option. Alternatively, external solutions based on Virtual Private Networks (VPNs), such as WireGuard, can be used; these establish tunnels that provide end-to-end encryption between cluster nodes. Although both solutions ensure traffic confidentiality, they operate at different levels of the network stack and can affect performance in different ways. Although the extensive literature on containerised network security and VPN protocol performance is fundamental, there is a lack of systematic experimental analysis to directly compare the native encryption of Docker overlay networks with a host-level solution based on WireGuard in a real-world Docker Swarm environment. In particular, the trade-off between security and performance in terms of latency, throughput and jitter is of limited relevance in the context of east-west traffic between microservices. This work aims to explore this important concept through a comparative experimental evaluation conducted on a three-node Docker Swarm cluster, running on three different VMware virtual machines. Four scenarios related to the four configurations were analysed: the reference physical network, a standard overlay network, an IPsec-encrypted overlay network, and secure communication using WireGuard. The main contributions of this work are as follows. First, we systematically evaluate Docker overlay network performance, both with and without encryption, to determine measurable impacts. Second, we provide a direct, side-by-side comparison between Docker Swarm's native IPsec encryption and a host-level WireGuard solution, highlighting their respective performance and security implications. Third, we quantitatively analyse the effect of these security mechanisms on latency, TCP/UDP throughput, and jitter. Finally, we discuss the architectural implications of selecting specific security mechanisms, offering clear recommendations for cloud-native environments. The results provide practical guidance for system designers and administrators in selecting appropriate security strategies for distributed microservices infrastructures.

2 Related Works

The security and performance of communications in cloud-native architectures have evolved significantly in recent years, particularly with the rise of microservices and containers. The literature can be divided into four main areas: the performance of containerised and overlay networks, the security of Docker networks, performance analysis of modern VPN protocols, and the protection of east-west traffic in microservices systems. The cloud computing is today an important technology that allow users to reach remote content and services, and to use them in a flexible way, without the need to manage the underlying infrastructure [4,5]. It is also important to note that the modern Artificial Intelligence (AI) systems are increasingly being deployed in cloud environments, and a microservices architecture is often used to manage the complexity of these systems and to ensure scalability, flexibility and security [6–8].

2.1 Performance of Container and Overlay Networks

Several studies have analysed the impact of virtualisation technologies and overlay networks on network performance. Wiedner and Helm [9] evaluate container performance in low-latency virtualised environments, highlighting how virtual networking mechanisms can introduce overhead compared to direct communication over a physical network. More recent studies have specifically analysed the performance of VXLAN-based overlay networks, showing that Layer 2 encapsulation over IP infrastructure results in increased latency and reduced throughput compared to direct host-to-host communication [10,11]. Lin [12] proposes a new low-overhead overlay network architecture for containerised environments, demonstrating how optimising forwarding mechanisms can significantly reduce the performance impact. In parallel, other works have investigated the impact of the Linux kernel's virtual networking mechanisms. Abani [13] analyses the latencies introduced by network namespaces, virtual bridges and veth interfaces, showing how these components can influence the behaviour of distributed applications.

2.2 Container Network Security

The security of communications between containers has been analysed in great depth recently. Lee, for example [14], conducts an experimental analysis of attacks on Docker networks, demonstrating that traffic between containers can be intercepted and manipulated in the absence of adequate protection mechanisms. A survey on container security [15] highlights how the increase in the attack surface in cloud-native architectures is closely linked to the complexity of communications between distributed services. In the context of microservices architectures [16,17], several studies have analysed the security issues associated with communication between services. Berardi [18] presents a systematic review of security in microservices, highlighting the need to protect internal communications between application components. Similarly, Nasab [19] conducts an

empirical study on security techniques in microservice systems, seeking to understand the numerous vulnerabilities linked to the management of internal communications. This is particularly critical when microservices are integrated into broader distributed architectures, such as IoT systems monitoring public areas, where data flow securely between edge nodes and the cloud [20].

2.3 VPN Protocols and Communication Encryption

In recent years, several VPN protocols have been proposed, designed to improve performance and ease of configuration. WireGuard is now one of the most recent and promising approaches, thanks to its minimalist architecture and the use of modern cryptographic primitives [21]. Several experimental studies have compared WireGuard with traditional VPN solutions. Anyam [22] conducts a comparative performance analysis between WireGuard and OpenVPN in cloud and virtualised environments, highlighting significant differences in terms of throughput and latency. Other works analyse the comparison between WireGuard and IPsec in different network configurations. Recent studies show that the performance of the two protocols depends heavily on the deployment context and the characteristics of the underlying infrastructure [23].

2.4 Protecting East-West Traffic in Microservices

The protection of internal communications in microservices architectures is often addressed through application-level mechanisms. Chellu [24] discusses the use of mutual TLS to ensure mutual authentication and end-to-end encryption between containerised services. In parallel, several studies have analysed the performance impact of service mesh architectures used to implement such mechanisms. Barr [25] compares different service-mesh implementations, highlighting that using sidecar proxies can introduce significant latency and resource-consumption overhead. Other works investigate alternative techniques to improve the security of east-west traffic. Hussain [26] proposes an eBPF-based approach to prevent spoofing attacks in containerised clusters, whilst recent studies analyse access control and authentication models in microservices architectures [27]. A review of the literature reveals that numerous studies have separately investigated the performance of overlay networks, the security of container networks, and the performance of modern VPN protocols. However, there are few studies that directly and systematically compare the native encryption of Docker overlay networks with host-level VPN solutions such as WireGuard in the context of a Docker Swarm cluster. This work fits within this area of research by providing a comparative experimental evaluation that combines security analysis and performance measurements in the context of east-west traffic between containerised microservices.

3 System Architecture and Experimental Methodology

This section analyses the experimental architecture used in the study, the test environment, the methodology for data collection, and the results obtained

across the different network scenarios. The aim of the experiments is to assess the impact of different traffic protection solutions on east-west communication between distributed, containerised microservices. The experimental infrastructure was implemented using a Docker Swarm cluster consisting of three virtualised nodes. One node acts as the manager and is responsible for service orchestration and cluster state management, whilst the other two nodes act as workers and host the application containers. Each node runs Docker Engine and participates in the cluster network via a standard IP connection. Communication between containers distributed across different nodes is achieved via a Docker overlay network based on VXLAN, which creates an independent virtual structure superimposed on the underlying physical network. The network architecture can be viewed as a multi-layered structure, where the first layer is the underlay network – that is, the underlying IP network that connects the virtual machines hosting the cluster nodes; the overlay network is the VXLAN-based Docker virtual network that enables communication between containers distributed across different nodes; the encrypted overlay, which is a variant of the Docker overlay with native IPsec-based encryption; and finally, WireGuard, a host-level VPN tunnel that ensures fully encrypted traffic between the cluster nodes. This multi-tiered structure enables comparison of different communication security strategies across microservices. Figure 1 shows the logical architecture of the experimental system.

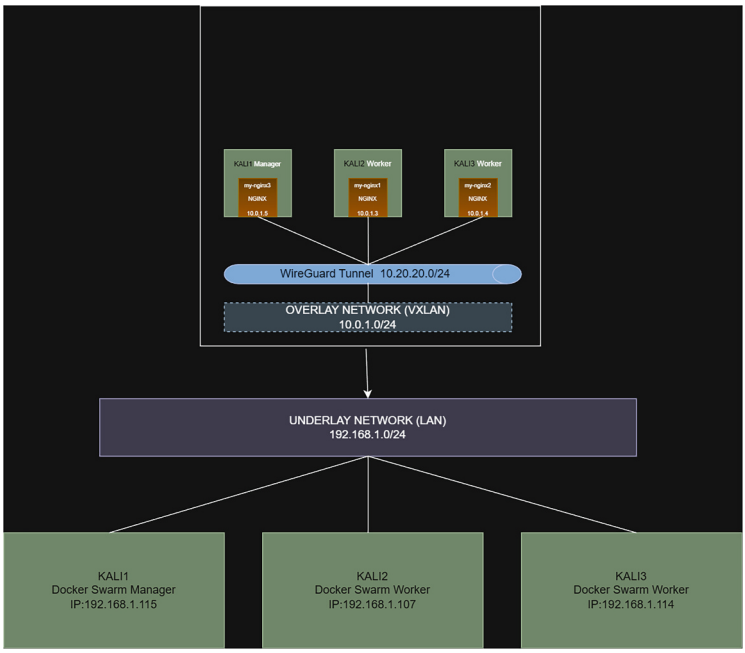

Fig. 1. System network architecture

3.1 Experimental Environment and Methodology

The experiments were conducted using three virtual machines configured as nodes in a Docker Swarm cluster. Each node runs Linux and has IP connectivity within the same virtualised local network. Application containers were deployed within the cluster and replicated across hosts to generate east-west traffic between nodes. Specifically, containers based on `nginx` images were used to simulate web services, and `alpine` containers were used to run testing and diagnostic tools. Communication between the containers was achieved via the Docker overlay network, which was dynamically created within the cluster. This approach allows containers to communicate using IP addresses from the overlay network, regardless of the physical host on which they run. The infrastructure was designed to closely resemble a realistic scenario and, above all, to be fully reproducible, utilising open-source tools and official Docker and WireGuard configurations. The aim of the experiments is to assess the impact of different encryption solutions on communication between distributed containers; for this reason, various test scenarios were defined in which the level of protection applied to communication between nodes varies. Measurement tools were used, such as tcpdump – a tool that allows packets to be captured at the interface level, useful for verifying traffic visibility – and Wireshark, one of the world's most famous and widely used tools for the graphical analysis of network activity, from flow reconstruction to payload inspection; the collected data was saved in .pcap files; Ping is a diagnostic tool based on two types of messages: the echo request and the echo reply, i.e. the request and the subsequent response to confirm connectivity; it uses the ICMP protocol and was used here to measure the Round Trip Time (RTT); curl is a tool that measures the total application response time (time_total), and finally iperf3, which measures TCP/UDP throughput, i.e. the actual data transfer rate. In addition to the tools, the study focused on analysing certain key metrics, namely RTT (i.e. average, minimum and maximum latency, with standard deviation); HTTP time (i.e. the total time taken to complete the application request); and TCP throughput (with average bitrate), and, where applicable, UDP jitter and packet loss.

3.2 Network Scenarios

The experiments were conducted by comparing four distinct network scenarios:

- Scenario A – Baseline underlay. This particular scenario represents the absolute baseline of the infrastructure, in which communication takes place directly between virtual machines without any overlay or tunnelling; it allows the performance of the physical network to be measured and provides a reference point for interpreting subsequent overheads, as this is where the overhead is lowest. In the absence of end-to-end encryption, containerised networking can expose east-west traffic to interception and analysis; it is therefore assumed that an adversary capable of capturing packets on a cluster node or a virtual network segment could compromise the system due to poor protection or misconfigurations.

- Scenario B – VXLAN Overlay. In this scenario, microservices communicate via a Docker overlay network based on VXLAN encapsulation. The overlay network enables connectivity between containers deployed on different nodes, but does not provide traffic encryption. An Alpine test container was used to enable communication between the nodes.
- Scenario C – Encrypted Overlay. The Docker overlay network is configured with native IPsec encryption. In this case, VXLAN traffic between the cluster nodes is automatically encrypted.
- Scenario D – WireGuard. In this scenario, traffic between nodes is protected by a tunnel, providing network-level encryption and integrity. Measurements were taken on hosts outside the container.
- Scenario E – Wireshark. This section analyses the observability of traffic across the three network layers used in the experimental lab: underlay, overlay and overlay with WireGuard. The aim is to demonstrate how the capture point influences traffic visibility and how different network mechanisms affect confidentiality and isolation.

Comparing these scenarios enables a systematic analysis of the trade-off between performance and security in communications between containerised microservices.

3.3 Results

We now present the results obtained from the experiments conducted across the different network scenarios, focusing on the key performance metrics of latency, throughput, and application response time, as well as traffic observability through packet captures.

Latency. The detailed results of the average latency measurements are shown in the graph below in Fig. 2 As can be seen, starting with the underlay scenario – i.e. the baseline physical network without any additional security or encryption – the average latency is optimal, at around 0.33 ms; as we move through the scenarios and add security and encryption, the latency increases. For example, in the second scenario, there is a slight increase due to VXLAN, which adds an additional header, resulting in an average latency of 0.44 ms. In the third scenario, there is a further, albeit modest, increase due to the use of IPsec, which introduces cryptographic encryption and decryption operations, with a latency of around 0.52 ms. The most significant increase in latency is seen in the WireGuard scenario, where we reach 0.78 ms, but this is still an optimal value as we remain below the 1 ms threshold; this is due to cryptographic operations but, above all, to the introduction of this VPN tunnel, which the packet must traverse, thereby lengthening the data path.

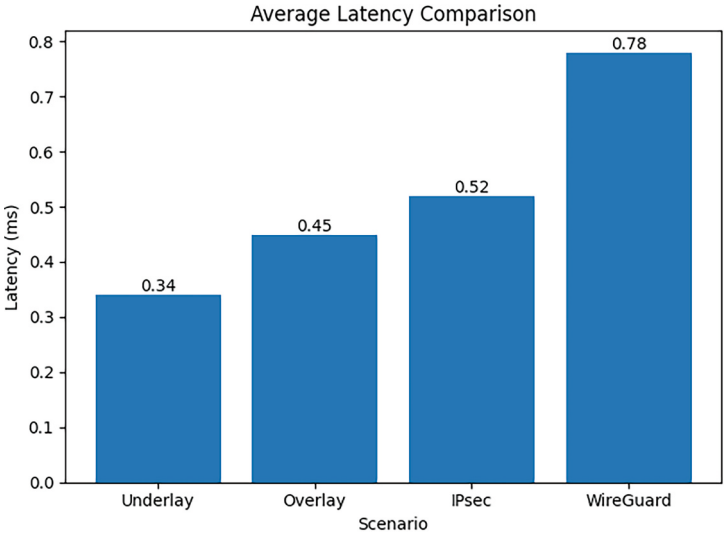

Fig. 2. Average latency results by scenario

Throughput. The detailed results of the TCP and UDP throughput measurements are shown in the graph below in Fig. 3

Starting again with the first underlay network scenario, we see a very high TCP throughput of around 3.07 Gbits/sec – which is therefore optimal – and a UDP throughput of around 222 Mbits/sec; at this point, simply introducing the overlay network with VXLAN drastically reduces the TCP value in the second scenario to 934 Mbits/sec and slightly reduces the UDP value to 198 Mbits/sec as well; this is due to the encapsulation of the protocol's additional headers; in the third scenario with IPsec, greater overhead is introduced and, consequently, both TCP throughput (639 Mbits/sec) and UDP throughput (170 Mbits/sec) are reduced. As can also be seen from the graph, WireGuard appears to be the best scenario in terms of performance, as whilst the TCP value does decrease further to 560 Mbits/sec, the UDP value increases for the first time compared to the previous scenarios to 478 Mbits/sec, confirming that WireGuard is the best solution to date in terms of the security-performance trade-off.

Curl. The detailed results of the application curl test are shown in the graph below in Fig. 4

As regards the application analysis, the response times measured using curl are in the order of a few milliseconds, indicating that the overhead introduced by the overlay is limited for simple, sequential HTTP requests. It is therefore important to clarify that this observation is valid in the specific context of the test performed, namely with synchronous, non-concurrent requests and a small

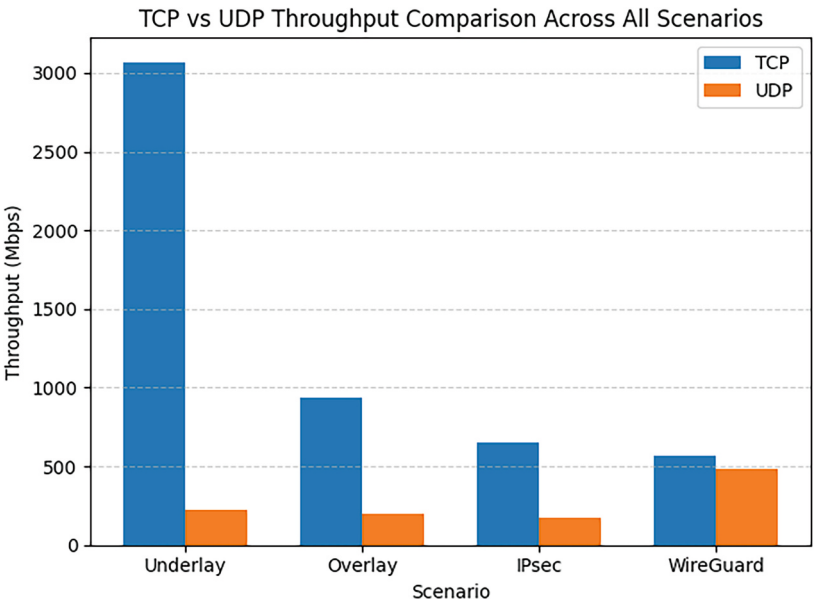

Fig. 3. TCP and UDP throughput results by scenario

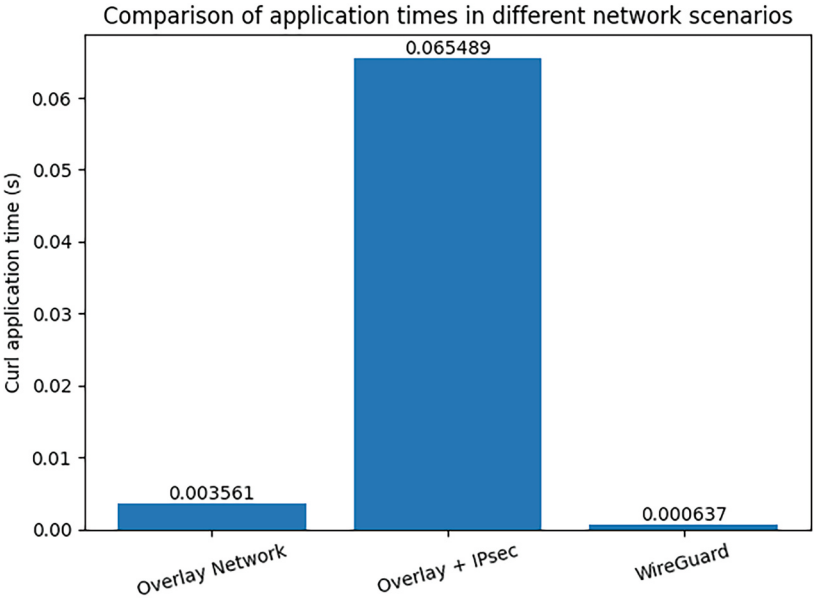

Fig. 4. Results for application time using curl

payload, which do not saturate the datapath or stress the container scheduling system. Under these conditions, the cost of VXLAN encapsulation is negligible compared to the overall application processing time. In real-world scenarios characterised by high concurrency, persistent connections, HTTP pipelines, or heavy loads, the overlay overhead could, however, become more significant. Therefore, the result obtained should be interpreted as valid for light, sequential workloads, whilst it does not claim to generalise the behaviour to intensive or highly concurrent application loads.

Wireshark. For the network analysis, we used Wireshark, which enabled us to examine the behaviour of two configurations: the first, the unencrypted overlay network shown in Fig. 5, and the second, WireGuard, shown in Fig. 6. In the first case, if we focus on the 'Protocol' section, we can see that the protocols are described and displayed in plain text, for example TCP and HTTP, and by opening the packet we can also see the corresponding TCP headers and payloads in plain text; on the right-hand side of the information, however, we can clearly see a GET request for HTTP and, a few lines below, the corresponding response, all in a transparent manner and readable by anyone; this makes it a highly vulnerable situation, prone to attacks such as sniffing. In the second case, again in the 'protocol' section, we see that only WireGuard appears in a generalised form, rather than the specific individual protocol, and on the right-hand side of the information, we no longer see the request and response; instead, all traffic is fully encrypted.

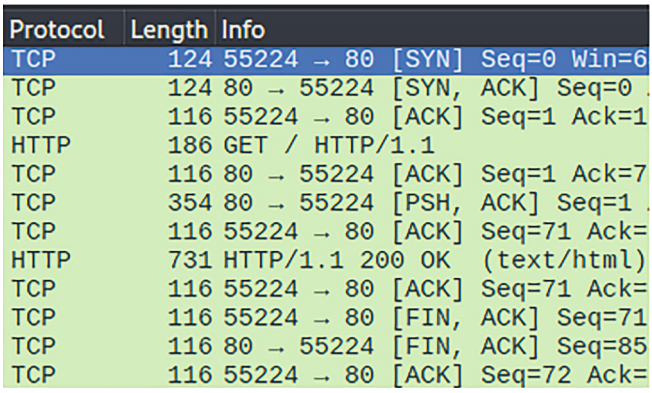

Fig. 5. Unencrypted overlay analysis

Protocol	Length	Info
WireGu…	74	Keepalive, receiver=0x3B1D2C
WireGu…	74	Keepalive, receiver=0xCA526B
WireGu…	74	Keepalive, receiver=0x1FD133
WireGu…	138	Transport Data, receiver=0xC
WireGu…	190	Handshake Initiation, sender
WireGu…	134	Handshake Response, sender=0
WireGu…	74	Keepalive, receiver=0x30C828
WireGu…	138	Transport Data, receiver=0x2
WireGu…	138	Transport Data, receiver=0x3
WireGu…	218	Transport Data, receiver=0x3
WireGu…	138	Transport Data, receiver=0xA
WireGu…	314	Transport Data, receiver=0xA

Fig. 6. WireGuard analysis

4 Discussion

The increase in latency and reduction in throughput observed in overlay and encrypted scenarios are not random phenomena, but the direct result of a combination of architectural and implementation factors that affect different network layers of the stack. The computational cost of encryption is significant. In the case of encrypted overlay networks (IPsec) and the WireGuard tunnel, each packet undergoes symmetric cryptographic operations (AES-GCM for IPsec and ChaCha-Poly1305 for WireGuard), which incur a per-packet computational cost. This cost manifests as increased processing time and a rise in the number of context switches between the kernel and user space, particularly evident in virtualised environments, where CPU resources are shared between hosts and virtual machines. The fact that TCP throughput suffers a more marked reduction than UDP is consistent with the greater number of control, retransmission, and state management operations required by reliable protocols. The reduction in effective MTU and fragmentation are two further important factors, as the use of overlays and VXLAN, as well as encrypted tunnels, introduces significant overhead due to encapsulation. Thus, approximately 50 bytes are added per packet with VXLAN, IPsec ESP adds 50–60 bytes, and WireGuard adds around 60 bytes. This effective reduction in the MTU therefore increases the likelihood of packet fragmentation or segmentation, particularly in TCP traffic, leading to more frames transmitted and placing additional load on the datapath. Does this explain why throughput does not decrease linearly but shows a measurable reduction relative to the baseline? The lengthening of the data path is also a factor; compared to the underlay, each packet traverses more components, such as the Docker virtual switch, VXLAN encapsulation, IPsec or WireGuard encryption, transmission over the physical underlay, and decapsulation and decryption upon reception. This lengthening directly contributes to the increase in end-to-end latency, but because the values remain below 1 ms, the observed overhead is primarily attributable to per-packet costs rather than congestion or packet loss.

In summary, the results show that the reduction in throughput is primarily due to computational overhead and fragmentation; the increase in latency is dominated by the lengthening of the data path and per-packet encryption; and the differences between IPsec and WireGuard can be explained by stack complexity and per-packet cost. It is important to compare the two security approaches and explain why it is insufficient to implement only the native IPsec method. The customised WireGuard approach must also be used. We can summarise this by saying that both are excellent security approaches, but we see that IPsec, in the configuration used, primarily protects the nodes that are part of the overlay network and therefore has a very limited scope and is mainly usable in this context, whilst WireGuard fully encrypts all traffic between nodes, offering greater flexibility even for different scenarios and outside the overlay context. The analysis of CPU usage, as shown in the graph in Fig. 7, based on the aggregate usage of CPU cores, highlights a different behaviour compared to, for example, an analysis of individual cores. In particular, WireGuard shows slightly higher overall CPU utilisation than IPsec. This is because WireGuard has an excellent ability to distribute the processing load across multiple processor cores, making more efficient use of multi-core architectures. Conversely, IPsec tends to concentrate a significant portion of the load on fewer cores, leading to greater saturation of individual cores but lower overall CPU utilisation across the system. Although it has this higher overhead, WireGuard remains a highly efficient solution thanks to its ability to handle parallel processing and the simplicity of its cryptographic design. When one also considers its greater flexibility in implementation and its integration with various network scenarios, a WireGuard-based configuration remains the most balanced choice in terms of the trade-off between security, performance, and scalability.

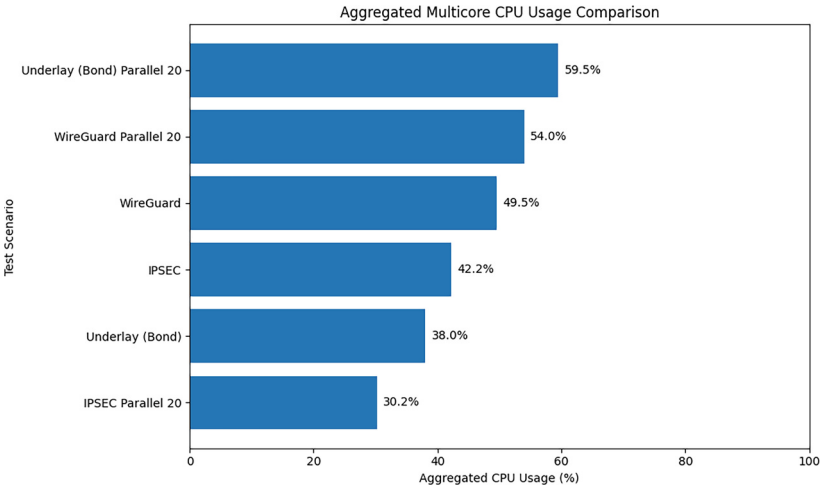

Fig. 7. CPU usage analysis

5 Conclusions

The study presented an experimental evaluation of performance and security properties across various network configurations in containerised environments based on Docker Swarm. Specifically, the study focused on analysing east-west traffic between distributed microservices, comparing four communication scenarios: a reference physical network, a VXLAN-based Docker overlay network, an IPsec-encrypted overlay, and communication secured via a WireGuard tunnel. The experimental results showed that the introduction of overlay networks and encryption mechanisms undoubtedly affects network performance. Direct communication over the physical network serves as the performance baseline, with high throughput and low latency. The use of the VXLAN overlay introduces an initial level of overhead due to encapsulation mechanisms, whilst the addition of encryption via IPsec or WireGuard tunnels entails further computational costs associated with cryptographic operations. From a security perspective, traffic analysis using packet-capture tools has shown that application traffic is easily observable in unencrypted configurations. Conversely, the use of encryption mechanisms renders the application payload unreadable in network captures, significantly reducing the attack surface associated with east-west traffic. Compared to the state of the art, this work provides a comparative experimental evaluation between the native encryption of Docker overlay networks and a host-level solution based on WireGuard in a real Docker Swarm cluster. The integrated analysis of performance and security highlights the trade-off between traffic protection and the overhead introduced by encryption mechanisms. From a practical perspective, the results suggest that the native encryption of the Docker overlay represents a simple solution that can be immediately integrated into existing infrastructures when the primary objective is to protect traffic between containers. Indeed, adopting a WireGuard tunnel between cluster nodes secures all inter-host traffic, providing a more comprehensive security model and making it particularly suitable in contexts where complete protection of internal communications is required. Overall, the study highlights that the choice of network protection mechanisms in microservice systems must be guided by a balance among security requirements, ease of management, and performance impact. Future developments could include extending the work to scenarios involving larger, geographically distributed clusters and to other widely used orchestration platforms such as Kubernetes, and incorporating metrics on CPU utilisation, energy consumption, and data-plane scalability. A further area of research concerns integration with application security mechanisms, such as service meshes, to assess the combined impact of network and end-to-end encryption in high-density microservice environments.

References

1. Perri, D., Simonetti, M., Gervasi, O.: Deploying efficiently modern applications on cloud. Electronics (Switzerland) **11**(3) (2022). https://doi.org/10.3390/electronics11030450

2. Fanfani, M., Palesi, L.A.I., Nesi, P.: Data-driven mobility demand-versus-offer matching by dynamic integration of public and sharing services. IEEE Access **14**, 23166–23193 (2026). https://doi.org/10.1109/ACCESS.2026.3661850
3. Gervasi, O., Perri, D., Simonetti, M.: Strategies and system implementations for secure electronic written exams. IEEE Access **10**, 20559–20570 (2022). https://doi.org/10.1109/ACCESS.2022.3150860. Green Open Access. All Open Access, Gold Open Access
4. Laganà, A., Gervasi, O., Tasso, S., Perri, D., Franciosa, F.: The ECTN virtual education community prosumer model for promoting and assessing chemical knowledge. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10964. LNCS (2018), pp. 533–548. https://doi.org/10.1007/978-3-319-95174-4_42. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85049988150&doi=10.1007%2f978-3-319-95174-4_42&partnerID=40&md5=23a9d4c7ef4ec100b6b06eeef01a25a6
5. Gervasi, O., Perri, D., Simonetti, M., Tasso, S.: Strategies for the digitalization of cultural heritage. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 13382. LNCS (2022). All Open Access, Hybrid Gold Open Access, pp. 486–502.https://doi.org/10.1007/978-3-031-10592-0_35. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85135788491&doi=10.1007%2f978-3-031-10592-0_35&partnerID=40&md5=51b7cb4fcebd5fbee1f71095b5b40a18
6. Fereidooni, Z., Palesi, L.A.I., Nesi, P.: Multi-agent optimizing traffic light signals using deep reinforcement learning. IEEE Access **13**, 106974–106988 (2025). https://doi.org/10.1109/ACCESS.2025.3578518
7. Dwivedi, R., et al.: Explainable AI (XAI): core ideas, techniques, and solutions. ACM Comput. Surv. **55**(9), 1–33 (2023)
8. Bilotta, S., Alessandro, L., Palesi, I., Nesi, P.: Exploiting open data for CO2 estimation via artificial intelligence and eXplainable AI. Expert Syst. Appl. **291**, 128598 (2025). issn: 0957-4174. https://doi.org/10.1016/j.eswa.2025.128598. https://www.sciencedirect.com/science/article/pii/S0957417425022171
9. Helm, M., Wiedner, F.: Performance evaluation of containers for low-latency packet processing in virtualized network environments, vol. 166. Elsevier (2024). https://doi.org/10.1016/j.peva.2024.102442
10. Gentile, A.F., et al.: Overlay and virtual private networks security performances analysis with open source infrastructure deployment. Future Internet **16**(8), 283 (2024). https://doi.org/10.3390/fi16080283
11. Ahmad, M.M., Haruna, Y., Yarima, K.I., Sani, M.A.: Analysis of docker networking and optimizing the overhead of docker overlay networks using OS kernel support. In: SCience Publishing Group (2022). https://doi.org/10.11648/j.net.20221002.11
12. Cao, P., Lin, S., Zhao, S., Han, D.: A cache-based low-overhead container overlay network. In: Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI) (2025)
13. Svensson, F., Abeni, L., Gustafsson, H., Cucinotta, T.: An investigation of linux virtual networking latencies. IEEE Trans. Network Serv. Manage. (2024)
14. Lee, J.-H., Lee, H., Kwon, S.: Experimental analysis of security attacks for docker container network in cloud computing. Electronics **12**(4), 940 (2023). https://doi.org/10.3390/electronics12040940
15. Dong, N., Jarkas, O., Ko, R., Mahmud, R.: A container security survey: exploits, attacks, and defenses. ACM Comput. Surv. (2025). https://doi.org/10.1145/3715001

16. Perri, D.: Dockerized architecture for a progressive web app: an Italian collocations dictionary. Lect. Notes Comput. Sci. **15891**, 33–43 (2026). https://doi.org/10.1007/978-3-031-97617-9_3. LNCS
17. Perri, D., Simonetti, M., Tasso, S., Ragni, F., Gervasi, O.: Implementing a scalable and elastic computing environment based on cloud containers. In: Gervasi, O., et al. ICCSA 2021. LNCS, vol. 12949. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86653-2_49
18. Mauro, J., Berardi, D., Giallorenzo, S.: Microservice security: a systematic literature review. J. Internet Serv. Appl. (2022). https://doi.org/10.7717/peerj-cs.779
19. Liang, P., Nasab, A.R., Shahin, M.: An empirical study of security practices for microservices systems. J. Syst. Softw. (2023). https://doi.org/10.1016/j.jss.2022.111563
20. Perri, D., Simonetti, M., Bordini, A., Cimarelli, S., Gervasi, O.: IoT to monitor people flow in areas of public interest. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **12958**, 658–672 (2021). https://doi.org/10.1007/978-3-030-87016-4_47. LNCS
21. Donenfeld, J.A.: WireGuard: next generation kernel network tunnel. In: Network and Distributed System Security Symposium (2017). https://api.semanticscholar.org/CorpusID:2590070
22. Larijani, H., Anyam, P., Singh, R.R.: Empirical performance analysis of WireGuard and OpenVPN in cloud and virtualised environments. Computers **14**(8), 326 (2025). https://doi.org/10.3390/computers14080326
23. Kırmızı, S., Demirdelen, T.: Performance evaluation of WireGuard and IPSec protocols in various network configurations. J. Inst. Sci. Technol. **8** (2025). https://doi.org/10.47495/okufbed.1660352
24. Chellu, R.: Secure Containerized Microservices Using PKI-Based Mutual TLS in Google Kubernetes Engine. Communications Networks and Information Security (2021)
25. Rampal, S., Barr, A.B., Tavori, J.: Performance comparison of service mesh frameworks: the mTLS test case. In: NOMS 2025-2025 IEEE Network Operations and Management Symposium (2025). https://doi.org/10.1109/NOMS57970.2025.11073712
26. Syed, H.J., Hussain, A., Aziz, A., Raza, S.: Preventing IP spoofing in kubernetes using eBPF. Computers, Mater. Continua (2025). https://doi.org/10.32604/cmc.2025.062628
27. Aldea, C.L., Bocu, R.: Authentication challenges and solutions in microservice architectures. Appl. Sci. **15**(22), 12088 (2025). https://doi.org/10.3390/app152212088