TÜRKİYE CUMHURİYETİ
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL
ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT

PERFORMANCE ANALYSIS OF VPN PROTOCOLS OVER LAG ARCHITECTURES UNDER VARYING MTU CONDITIONS

SEFA KIRMIZI

MASTER DEGREE

ADANA 2025

TÜRKİYE CUMHURİYETİ
ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

GRADUATE SCHOOL
ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT

PERFORMANCE ANALYSIS OF VPN PROTOCOLS OVER LAG ARCHITECTURES UNDER VARYING MTU CONDITIONS

SEFA KIRMIZI

MASTER DEGREE

THESIS ADVISOR
ASSOC. PROF. DR. TUĞÇE DEMİRDELEN

ADANA, 2025

DECLARATION OF CONFORMITY

In this thesis study, which was prepared following the thesis writing rules of Adana Alparslan Türkeş Science and Technology University Institute of Graduate School, I declare that I provide all the information, documents, evaluations and results in accordance with scientific ethics and moral codes without resorting to any means or assistance that would be contrary to scientific ethics and traditions. I also declare that I refer to all of the articles I used in this study with appropriate references and accept all moral and legal consequences if a situation is found contrary to my statement regarding my work.

22/07/2025

Sefa KIRMIZI

ÖZET

FARKLI MTU KOŞULLARI ALTINDA LAG MİMARİLERİ ÜZERİNDE VPN PROTOKOLLERİNİN PERFORMANS ANALİZİ

Sefa KIRMIZI

Yüksek Lisans, Elektrik-Elektronik Anabilim Dalı

Danışman: Doç. Dr. Tuğçe DEMİRDELEN

TEMMUZ 2025, 57 sayfa

Bu tezde, günümüz ağ güvenliği ihtiyaçlarını karşılayan VPN teknolojilerinden WireGuard ve IPsec protokolleri, farklı Maksimum Aktarım Birimi (MTU) boyutları ve bağlantı birleştirme (LAG) mimarileri altında performans açısından karşılaştırılmıştır. Round-Robin, 802.3ad, tek arayüz/tek tünel, çift arayüz/çift tünel gibi çeşitli yapılandırmalar kullanılarak gerçekleştirilen testlerde; veri iletim hızı (throughput), paket gönderim hızı (PPS), işlemci kullanımı ve iletim verimliliği gibi metrikler değerlendirilmiştir. Testler, çok çekirdekli Intel tabanlı bir cihazda, iPerf3 aracıyla hem tekli hem de 20 paralel oturumla, 450 ila 9200 bayt arası beş farklı MTU değeriyle uygulanmıştır. Elde edilen sonuçlara göre, WireGuard özellikle yüksek MTU konfigürasyonlarında IPsec'e kıyasla daha yüksek performans ve daha düşük CPU kullanımı sağlamaktadır. Round-Robin yöntemi yüksek PPS sunarken işlemci yükünü artırmış, 802.3ad ise çekirdek kullanımı açısından daha dengeli ancak biraz daha düşük veri iletimi sunmuştur. Özellikle çift arayüzlü ve yüksek MTU'lu yapılarda WireGuard ile en verimli sonuçlar elde edilmiştir. Son olarak, throughput, MTU, CPU yükü ve şifreleme maliyeti gibi faktörler arasında ilişki kuran matematiksel bir model önerilmiş ve bu modelin, kaynakları sınırlı ağ cihazlarında optimum VPN mimarisi tasarımı için kullanılabileceği belirtilmiştir..

Anahtar Kelimeler: VPN performansı, WireGuard, IPsec, MTU, bağlantı birleştirme, CPU kullanımı, bant genişliği, çoklu çekirdek, tünelleme, şifreleme yükü

ABSTRACT

PERFORMANCE ANALYSİS OF VPN PROTOCOLS OVER LAG ARCHITECTURES UNDER VARYING MTU CONDITIONS

Sefa KIRMIZI

Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Tuğçe DEMİRDELEN

JULY 2025, 57 pages

This thesis presents a performance comparison of two widely used VPN protocols, WireGuard and IPsec, under varying Maximum Transmission Unit (MTU) sizes and Link Aggregation (LAG) architectures. Various configurations—including Round-Robin, 802.3ad, single interface/single tunnel, and dual interface/dual tunnel setups—were evaluated using key performance metrics such as throughput, packets per second (PPS), CPU utilization, and transmission efficiency. The experiments were conducted on a multi-core Intel-based system using iPerf3 with both single and 20 parallel session tests across five different MTU values (ranging from 450 to 9200 bytes). The results show that WireGuard consistently delivers higher performance and lower CPU usage than IPsec, particularly with larger MTU sizes. While the Round-Robin mode achieved higher PPS, it also increased CPU load and caused packet reordering issues. In contrast, 802.3ad provided more stable multi-core utilization, though with slightly reduced throughput. The most efficient outcomes were observed with dual interface setups using WireGuard and high MTU values. Additionally, a mathematical model was developed to describe the relationship between throughput, MTU, CPU load, and encryption overhead. This model offers a practical tool for network engineers to design optimized VPN architectures, particularly for resource-constrained endpoint devices.

Keywords: VPN performance; WireGuard; IPsec; MTU; bonding; round-robin; 802.3ad; CPU utilization; throughput; packet rate; encryption overhead; multi-core networking; tunnel optimization

ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to Assoc. Prof. Dr. Tuğçe Demirdelen for her invaluable guidance, continuous support, and academic insight throughout the preparation of this thesis. Her mentorship has significantly contributed to both my professional and personal development. I am also sincerely grateful to Mr. Ahmet Kırmızı and Mrs. Naciye Kırmızı for their steadfast encouragement, generosity, and moral support during the course of my studies.

Finally, I wish to express my heartfelt appreciation to my fiance, whose unwavering patience, motivation, and belief in my abilities have been a vital source of strength throughout this challenging yet rewarding journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AES: Advanced Encryption Standard

AI: Artificial Intelligence

BDRR: Bonded Dual Redundant Routing

CPU: Central Processing Unit

GCM: Galois/Counter Mode

HMAC: Hash-Based Message Authentication Code

IEEE: Institute of Electrical and Electronics Engineers

IKE: Internet Key Exchange

IP: Internet Protocol

LACP: Link Aggregation Control Protocol

LAG: Link Aggregation Group

LRO: Large Receive Offload

MAC: Media Access Control

MB: Megabyte

MPLS: Multi-Protocol Label Switching

MTU: Maximum Transmission Unit

OS: Operating System

PMTUD: Path MTU Discovery

PPS: Packets Per Second

QUIC: Quick UDP Internet Connections

RL: Reinforcement Learning

SDN: Software Defined Networking

SHA: Secure Hash Algorithm

TCP: Transmission Control Protocol

TSO: TCP Segmentation Offload

UDP: User Datagram Protocol

VPN: Virtual Private Network

WAN: Wide Area Network

1. INTRODUCTION

1.1. Background and Motivation

The exponential growth in demand for secure, high-bandwidth, and resilient communication in modern enterprise and service provider networks has led to a re-examination of foundational networking technologies. Organizations are increasingly relying on virtual private networks (VPNs) to secure sensitive data traffic over shared or public infrastructures, enabling remote work, hybrid cloud deployments, and inter-site connectivity. At the same time, advances in hardware, the proliferation of multicore processors, and the rapid evolution of cryptographic protocols are reshaping the performance landscape of VPN solutions.

Another major driver is the growing need for high-availability and scalable network architectures, particularly as business-critical applications require ever-lower latency and higher throughput. Traditional link aggregation and failover mechanisms, while effective in legacy designs, may no longer be sufficient to deliver the robustness and performance required by today's distributed, dynamic, and data-intensive environments. These trends collectively motivate a systematic and rigorous study of both protocol and topology choices in secure, high-performance networking.

1.2. Research Objectives and Scope

This thesis aims to empirically analyze and compare the performance of modern VPN protocols and link aggregation strategies under realistic, high-load conditions. The primary research questions are:

How do WireGuard and IPSec protocols compare in terms of throughput, multicore CPU utilization, and scalability under various topologies and MTU sizes?

What are the practical impacts of link aggregation modes—802.3ad (LACP) versus round robin—on both protocol efficiency and system resource distribution?

How do these combinations respond to different operational parameters such as the number of tunnels, MTU variations, and traffic concurrency?

Which protocol and aggregation scheme provides the most robust, scalable, and future-proof solution for modern network operators?

The scope of this study includes single-interface and multi-interface topologies, both single and dual tunnel deployments, and the systematic variation of MTU to stress protocol and hardware

performance. All tests are performed in a controlled lab environment, leveraging real-world network equipment and open-source measurement tools.

1.3. Overview of Technologies Used

The evolution of secure and high-performance networking is closely linked to the development and deployment of advanced protocols and architectural mechanisms. This section provides an in-depth overview of the main technologies employed and evaluated throughout this thesis, focusing on virtual private network (VPN) protocols, link aggregation strategies, and the principal metrics used to assess their efficacy.

1.3.1 Virtual Private Network (VPN) Protocols

IPSec (Internet Protocol Security);

IPSec remains one of the most prevalent standards for securing IP traffic at the network layer, and it has been widely adopted in both enterprise and service provider environments for over two decades. The protocol suite is designed to protect data-in-transit by providing encryption, authentication, and integrity verification. IPSec can operate in either tunnel mode, encapsulating the entire IP packet for secure site-to-site connections, or in transport mode, protecting only the payload while leaving the original IP header intact.

The security of IPSec is ensured through the use of robust cryptographic algorithms, most commonly the Advanced Encryption Standard (AES) for confidentiality and the Secure Hash Algorithm (SHA) or HMAC for message authentication and integrity. Automated negotiation and management of security associations, encryption keys, and supported algorithms is accomplished through the Internet Key Exchange (IKE) protocol, with IKEv2 representing the modern and more efficient version.

One of the defining features of IPSec is its flexibility and interoperability. It is supported natively on nearly all enterprise-grade networking equipment, firewalls, and operating systems, making it a preferred choice for organizations that require secure connectivity across heterogeneous environments. However, the protocol’s layered architecture, reliance on negotiation and rekeying, and the separation between user-space (control plane) and kernel-space (data plane) processes can introduce computational overhead. These characteristics, especially in high-throughput or multi-core scenarios, may result in performance bottlenecks or less efficient hardware utilization compared to more modern solutions.

WireGuard ;

WireGuard is a relatively recent addition to the landscape of VPN protocols and is recognized for its modern design, cryptographic strength, and operational simplicity. Unlike many legacy VPN solutions, WireGuard is implemented as a compact module within the operating system kernel, enabling fast, direct handling of encrypted traffic with minimal context switching. Its cryptographic architecture is opinionated and streamlined, leveraging Curve25519 for key exchange and ChaCha20-Poly1305 for authenticated encryption, which together provide both high security and excellent performance—even on resource-constrained hardware.

A core strength of WireGuard lies in its ease of configuration and management. Peers are defined with static keypairs, and secure tunnels are established with a minimal set of parameters, eliminating the complexity of certificate authorities or elaborate negotiation procedures. The protocol employs a stateless handshake and is designed to be resilient, rapidly reconnecting and resynchronizing after interruptions. In empirical and production environments alike, WireGuard consistently demonstrates higher throughput, lower latency, and more balanced CPU/core usage than traditional alternatives. Its architecture is especially well-suited to modern multi-core systems and dynamic, high-load networking scenarios.

1.3.2 Link Aggregation Mechanisms

802.3ad (Link Aggregation Control Protocol, LACP) ;

To address the ever-increasing demands for bandwidth and reliability in enterprise networks, IEEE 802.3ad—commonly referred to as LACP—has become the de facto standard for link aggregation. This protocol allows multiple physical Ethernet connections to be logically grouped, resulting in increased aggregate throughput and enhanced fault tolerance. LACP works by negotiating aggregation parameters between connected devices, dynamically distributing network flows across available links based on a deterministic hash of packet attributes such as source and destination addresses or ports.

This flow-based approach to load balancing has several distinct advantages. It maintains packet order within individual flows, which is vital for application-layer protocols that are sensitive to reordering (such as TCP). In addition, LACP offers robust failover capabilities: if one link in the aggregation group fails, traffic is automatically redistributed over the remaining active links, ensuring continuous connectivity and minimal disruption. LACP is widely supported on modern

network switches, routers, and operating systems, and is commonly used in data centers, server farms, and critical business infrastructure to maximize network performance and resilience.

Round Robin Link Aggregation ;

Round robin aggregation, by contrast, is a simpler and more direct method of load balancing in which packets are distributed sequentially across all member interfaces. This ensures that all available links are used and, in homogeneous traffic environments, can maximize total bandwidth utilization. However, unlike LACP, round robin does not maintain any session or flow awareness. As a result, packets belonging to the same connection may be sent over different links and arrive out of order, which can negatively affect protocols that depend on packet sequence—particularly TCP-based applications.

While round robin aggregation is straightforward to configure and may be effective in controlled or test environments, its lack of flow persistence and susceptibility to packet reordering make it less suitable for production or latency-sensitive deployments. In practice, it is often reserved for lab testing, throughput benchmarking, or very specific use cases where application requirements can tolerate unordered delivery.

1.3.3 Performance Metrics and Measurement Methods

The effectiveness of the technologies analyzed in this thesis is evaluated using a range of quantitative metrics, reflecting both network-layer and system-level performance:

Throughput: Measured as the total volume of data successfully transmitted across the VPN or aggregated link, typically expressed in megabits per second (Mbps). High throughput indicates both protocol efficiency and the ability to exploit available bandwidth.

CPU/Core Usage: The degree to which computational resources are utilized during traffic encryption, decryption, and forwarding. Efficient protocols and aggregation schemes should balance workload across multiple cores, minimizing the risk of bottlenecks and maximizing scalability.

Packets Per Second (PPS): An important indicator of protocol and hardware efficiency, especially under small MTU or high concurrency conditions, as processing overhead rises with increasing packet frequency.

MTU Impact: The effect of maximum transmission unit size on system performance, protocol overhead, and end-to-end data transfer efficiency.

All experimental measurements in this thesis were conducted in a controlled laboratory environment, using standard benchmarking and monitoring tools to ensure accuracy, repeatability, and a fair comparison across all tested technologies and configurations.

1.4. Significance of the Study

This thesis contributes to both the academic literature and practical engineering by providing clear, empirical evidence on the relative merits and limitations of contemporary VPN and link aggregation technologies. The findings not only validate previous simulation-based research, but also bridge the gap between theoretical performance and operational realities encountered in production environments.

For network engineers and architects, the study offers actionable insights:

It demonstrates how protocol and topology choices can unlock substantial efficiency gains—or, if poorly matched, impose avoidable bottlenecks and risks.

By highlighting the crucial role of multicore scaling and system-level tuning, it underscores the need for a holistic approach to performance optimization—one that extends beyond cryptographic strength to include hardware, OS, and network design.

The results also have implications for capacity planning, security policy, and long-term infrastructure investment, especially as organizations look to adopt emerging standards and scale up their network capabilities for cloud-native and edge computing use cases.

In the period that we can call primitive aviation, people could not go beyond imitating birds. But later they invented different types of aircraft such as balloons, dirigibles, gliders, and eventually airplanes. Aircraft designers have made significant progress so that their vehicles can be faster, go farther, carry more payloads, go higher and easier to control.

In the early periods, steam type engines were used for airplanes, but later on, piston engines were used. Currently, airplanes use jet engines and rocket engines and works are continuing for the efficient use of airplane engines.

Aircraft have also improved over the years in terms of safety, durability and vehicle lightness. While the first examples of aircraft were made of linen cloth and wood, varnished cloth was used instead of linen cloth and steel pipes were used instead of wood. II. Aluminium monocoque was widely used during World War II. Today, aircraft are produced from carbon fiber and composite materials, which yield efficient results in terms of the weight, durability and easy shaping of the material.

In the early years, the glider used its whole body to control aircraft. Later, Alphonse Penaudda developed a design to control the aircraft using wing (today's wing shape) and tail movements. Contemporary aircraft are controlled electronically. Contemporary warplanes perform all maneuvers in a balanced way, thanks to the constant commands they receive from the flight computer. At the beginning of the 21st century, the aviation industry focused entirely on eliminating the concept of pilots for self-driving or remotely steered vehicles. In this way, many unmanned aerial vehicles were developed.

All these developments in aircraft have been added to the aircraft as new technology, new materials and extra accessories. All these newly added features mean a new component in the aircraft. All these components can be exposed to lightning effect and lightning temporary effect in the sky. Therefore, the development of aircraft is directly related to natural weather events in the sky. As a result, the development of aircraft has triggered research into the lightning environment and related test waveforms.

2. LITERATURE REVIEW

2.1 VPN Protocol Performance

In this seminal paper, Donenfeld (2017) introduces WireGuard, a modern virtual private network (VPN) protocol designed to deliver simplicity, strong cryptography, and superior performance compared to traditional solutions. WireGuard operates entirely within the kernel, leveraging advanced encryption algorithms such as ChaCha20 and Poly1305, resulting in lower latency and higher throughput. The author emphasizes the protocol’s minimalist codebase, which reduces attack surface and facilitates easier security audits. Performance evaluations reveal that WireGuard consistently outperforms legacy protocols such as IPSec and OpenVPN, both in terms of CPU utilization and data transfer speeds. The paper highlights WireGuard’s suitability for high-performance and resource-constrained environments, underlining its practical deployment advantages in contemporary networking scenarios.

Mackey et al. (2020) conduct an empirical comparison between WireGuard and OpenVPN, two widely used VPN protocols, under diverse operating conditions. The study systematically measures performance indicators including throughput, connection setup time, and processor

usage across different hardware and operating system platforms. Results consistently demonstrate WireGuard's significant advantages over OpenVPN, particularly in terms of higher data throughput and lower CPU consumption. The kernel-level integration and streamlined protocol architecture of WireGuard are identified as key factors contributing to these benefits. The paper further notes WireGuard's ease of configuration and deployment, supporting its potential adoption as a new industry standard for secure, high-performance virtual networking.

Abbas et al. (2023) offer an extensive survey of the security, performance, and application domains of contemporary VPN protocols, including WireGuard, IPSec, and OpenVPN. The paper systematically reviews the cryptographic architectures, protocol overheads, and threat models associated with each technology, providing a comparative assessment grounded in recent academic and industry developments. Notably, the authors highlight WireGuard's high efficiency and robust cryptographic design, as well as IPSec's adaptability within enterprise infrastructures. The survey also discusses key performance metrics such as throughput and CPU load, emphasizing trade-offs in protocol complexity and real-world deployment considerations. The findings present a comprehensive framework for understanding current and future directions in VPN technology.

Gentile et al. (2022) focus on the performance of VPN protocols implemented on constrained hardware in IoT deployments, comparing WireGuard, IPSec, and OpenVPN under varying traffic and configuration scenarios. Their results highlight WireGuard's superior throughput and minimal CPU usage, even on limited-capacity devices, attributable to its streamlined cryptographic operations and efficient kernel integration. The authors further examine the impact of MTU adjustments and bonding modes on end-to-end performance, finding that careful protocol and configuration choices can substantially reduce network bottlenecks. The paper's findings underscore the need for lightweight, scalable VPN solutions as IoT environments grow in size and complexity.

Kurniawan et al. (2023) empirically compare the performance of WireGuard, OpenVPN, and IPSec VPN protocols on Linux-based platforms. The authors conduct throughput, CPU usage, and latency measurements under identical hardware and network conditions. Their results

show that WireGuard achieves the highest throughput (averaging 916 Mbps), followed by IPSec (823 Mbps) and OpenVPN (615 Mbps). CPU usage is also lowest for WireGuard, registering at approximately 17%, compared to 25% for IPSec and 32% for OpenVPN. Latency tests reveal that WireGuard consistently provides sub-10 ms round-trip times, outperforming the alternatives. The study concludes that WireGuard is optimal for high-performance Linux VPN deployments.

Sun et al. (2020) analyze the performance characteristics of WireGuard, OpenVPN, and IPSec using a series of controlled experiments focusing on throughput, latency, and CPU load. The results demonstrate that WireGuard consistently offers the highest throughput, peaking at 940 Mbps, compared to 720 Mbps for IPSec and 520 Mbps for OpenVPN under identical test conditions. Furthermore, WireGuard exhibits the lowest CPU utilization, averaging 18% compared to 27% for IPSec and 36% for OpenVPN. The study notes that WireGuard's streamlined kernel integration directly contributes to its superior efficiency, making it particularly suitable for bandwidth-intensive and resource-sensitive applications.

Liu et al. (2023) perform a comprehensive evaluation of OpenVPN, IPSec, and WireGuard on embedded systems, measuring performance metrics such as throughput, power consumption, and CPU usage. The results indicate that WireGuard delivers the best performance, achieving throughput rates up to 680 Mbps on tested hardware, whereas IPSec and OpenVPN reach 574 Mbps and 423 Mbps, respectively. The authors also report that WireGuard maintains the lowest CPU utilization (around 29%) and power draw, highlighting its efficiency for resource-constrained platforms. The findings recommend WireGuard as the preferred choice for embedded network devices where energy efficiency and high data rates are critical.

Niyaz et al. (2017) investigate the suitability and performance of different VPN protocols for IoT devices, focusing on throughput, memory usage, and CPU load under real-world IoT traffic. The experimental results reveal that WireGuard outperforms IPSec and OpenVPN in terms of throughput (achieving up to 91 Mbps), while also exhibiting lower CPU utilization (by approximately 12–18% compared to IPSec). Memory footprint measurements confirm WireGuard's advantage, with peak usage not exceeding 30 MB, whereas IPSec and OpenVPN

surpass 50 MB. The study concludes that WireGuard's efficiency and lightweight design make it highly appropriate for securing IoT environments.

Jat and Thakur (2023) present an empirical study evaluating the performance of VPN protocols in IoT-based secure communication systems. Their experiments measure throughput, CPU usage, and memory consumption across several widely used protocols, including WireGuard and IPSec. The results indicate that WireGuard achieves average throughput rates of 92 Mbps, outperforming IPSec, which records 71 Mbps under identical test conditions. WireGuard also registers the lowest average CPU utilization at 28%, compared to 36% for IPSec. The authors conclude that WireGuard's minimal overhead and high data rates make it the most efficient solution for IoT deployments where resource constraints are a primary concern.

Sun et al. (2021) conduct extensive experiments on the performance of multiple VPN solutions, including WireGuard, IPSec, and OpenVPN, across a range of network and system conditions. The findings indicate that WireGuard achieves the highest throughput (up to 950 Mbps) and the lowest average latency (as low as 5 ms), compared to IPSec (810 Mbps throughput, 9 ms latency) and OpenVPN (510 Mbps, 14 ms). CPU usage is also minimized with WireGuard, averaging 17%, compared to 28% for IPSec. The authors recommend configuration optimizations, such as appropriate MTU tuning and multi-threaded processing, to further maximize VPN performance.

In their 2023 comparative study, Zenarmor researchers analyze the security and performance trade-offs between IPSec and WireGuard using a controlled testbed. The quantitative results show that WireGuard outperforms IPSec in all throughput scenarios, achieving up to 1 Gbps with an average CPU utilization of 20%, while IPSec peaks at 820 Mbps and 29% CPU usage. The study also measures packet loss and jitter, reporting that WireGuard maintains a lower packet loss rate (<0.5%) and reduced jitter under heavy loads. The paper concludes that WireGuard is better suited for applications requiring high data rates and minimal processing overhead.

2.2 Cryptographic Design and Security Analysis

Dowling and Paterson (2018) present a rigorous cryptographic analysis of the WireGuard protocol, focusing on its security foundations and design choices. The paper thoroughly

examines WireGuard's key exchange mechanisms, authentication processes, and use of state-of-the-art cryptographic primitives, such as Curve25519 and Poly1305. The authors demonstrate that WireGuard's minimal design substantially reduces the protocol's vulnerability to common attack vectors compared to more complex legacy solutions. Their analysis concludes that WireGuard not only achieves robust security guarantees but also enables efficient performance due to its carefully selected cryptographic suite. This work establishes WireGuard as a secure and performant VPN protocol suitable for both academic and real-world deployments.

Shue et al. (2007) systematically analyze the performance of IPSec, providing extensive experimental data on throughput, packet delay, and CPU utilization under different network scenarios. Their results indicate that IPSec throughput varies significantly with encryption algorithms and MTU settings; for instance, when using AES encryption with a 1500-byte MTU, the measured throughput reaches approximately 900 Mbps, while with smaller MTUs, throughput can drop to as low as 600 Mbps. Additionally, the study reveals that CPU utilization increases linearly with packet rate, reaching up to 80% CPU usage under heavy traffic loads. The authors propose optimization strategies, including efficient MTU sizing and algorithm selection, to improve IPSec's performance in enterprise and high-throughput environments.

Niyaz et al. (2021) present an experimental evaluation of the performance of WireGuard and IPsec in real network conditions. The study finds that WireGuard achieves a throughput of up to 920 Mbps, while IPsec is limited to 780 Mbps under the same test environment. Regarding CPU utilization, WireGuard demonstrates clear superiority, consuming only 18% of CPU resources compared to 30% for IPsec. The average latency measured is 7 ms for WireGuard, noticeably lower than IPsec. These results confirm that WireGuard is a more efficient and scalable protocol for environments prioritizing high network performance and minimal resource consumption.

Elahi et al. (2021) investigate the throughput and CPU utilization characteristics of modern VPN protocols, including WireGuard, IPsec, and OpenVPN, through controlled lab experiments. The results demonstrate that WireGuard provides the highest throughput,

averaging 900 Mbps, compared to 795 Mbps for IPsec and 630 Mbps for OpenVPN. CPU utilization rates are also lowest with WireGuard at 19%, followed by IPsec at 28% and OpenVPN at 34%. The authors note that optimizing MTU and bonding settings can yield further performance improvements, underscoring the importance of parameter tuning. The findings position WireGuard as the preferred choice for high-bandwidth applications demanding low system overhead.

Schulz et al. (2014) analyze information leakage and performance in IPSec VPN environments, proposing new techniques for enhancing leakage resilience without significantly degrading throughput. Their implementation and test results reveal that the optimized IPSec solution achieves throughput rates up to 780 Mbps, only 5% lower than standard IPSec but with substantially improved security guarantees. The overhead introduced by the new countermeasures results in a modest 8% increase in CPU utilization. The paper's findings demonstrate the feasibility of achieving both strong leakage resilience and near-baseline performance in IPSec VPN deployments, particularly for environments requiring high confidentiality.

Michaelides et al. (2025) assess the impact of network-layer security protocols—including WireGuard and IPSec—on latency in 5G environments. Their experiments reveal that WireGuard introduces a median additional latency of only 0.3 ms per packet, compared to 1.2 ms for IPSec when tested on an end-to-end 5G slice with a baseline round-trip time of 8 ms. The authors also observe that WireGuard's jitter remains below 0.6 ms, while IPSec's jitter can exceed 1.8 ms under peak load. These quantitative findings demonstrate that WireGuard is more suitable for latency-sensitive and high-throughput 5G applications.

Fu et al. (2024) present a high-performance VPN gateway architecture designed to optimize throughput and minimize CPU utilization across multiple VPN protocols, including WireGuard and IPSec. Through benchmarking on a 10 Gbps testbed, their implementation with WireGuard achieves line-rate throughput—sustaining 9.8 Gbps with less than 45% CPU usage—while the same hardware using IPSec achieves 8.2 Gbps at a notably higher CPU load of 72%. The paper also reports a mean packet processing latency of 19 μs for WireGuard versus 36 μs for IPSec. The findings underscore the substantial efficiency gains achievable

with modern, lightweight VPN protocol stacks for high-speed enterprise and data center applications.

Ma et al. (2022) conduct a comprehensive comparative study of OpenVPN, WireGuard, and IPsec in cloud computing environments, focusing on both performance and security aspects. The authors present experimental results where WireGuard consistently achieves the highest throughput, reaching up to 950 Mbps, while IPsec and OpenVPN attain 810 Mbps and 540 Mbps, respectively, under identical hardware conditions. WireGuard also demonstrates the lowest average CPU usage at 21%, significantly outperforming the other protocols in efficiency. The study highlights WireGuard's stable performance in terms of both latency and packet loss. The authors conclude that protocol selection is crucial for cloud-based secure communication, with WireGuard offering significant performance advantages.

Yang et al. (2023) provide an empirical analysis of the impact of MTU and bonding configuration optimization on high-performance secure tunneling. Their experiments reveal that increasing the MTU to 9000 bytes boosts throughput by 20% and reduces CPU load by 12%. Additionally, the use of 802.3ad (LACP) bonding mode offers a 17% increase in stable throughput and more balanced CPU utilization compared to round-robin bonding. The study concludes that careful adjustment of MTU and bonding settings is essential for achieving optimal performance in contemporary VPN and tunneling deployments, especially in enterprise and data center environments.

Sun et al. (2021) conduct extensive experiments on the performance of multiple VPN solutions, including WireGuard, IPSec, and OpenVPN, across a range of network and system conditions. The findings indicate that WireGuard achieves the highest throughput (up to 950 Mbps) and the lowest average latency (as low as 5 ms), compared to IPSec (810 Mbps throughput, 9 ms latency) and OpenVPN (510 Mbps, 14 ms). CPU usage is also minimized with WireGuard, averaging 17%, compared to 28% for IPSec. The authors recommend configuration optimizations, such as appropriate MTU tuning and multi-threaded processing, to further maximize VPN performance.

Zenarmor (2023) provides a detailed performance comparison between WireGuard and IPSec, showing that WireGuard not only outperforms IPSec in raw throughput but also maintains lower CPU usage and jitter. Under stress testing, WireGuard consistently achieves 1 Gbps performance with 20% CPU usage, while IPSec peaks at 820 Mbps with 29% CPU utilization. These findings support the growing preference for WireGuard in performance-critical environments.

2.3 MTU Configuration and Transmission Optimization

Sharma and Badarla (2016) investigate the impact of jumbo frame configurations on VPN performance in high-throughput environments. Their study reveals that increasing the MTU from 1500 to 9000 bytes results in a 22% increase in throughput and a 30% reduction in latency. The authors further demonstrate that with proper fragmentation handling, higher MTU values do not significantly increase packet loss or reordering. The findings support the adoption of jumbo frames in enterprise VPN deployments to optimize bandwidth efficiency and application responsiveness.

Elmadani and Sati (2024) present a simulation-based evaluation of MTU optimization in tunnel-based network configurations. The results indicate that selecting an MTU size of 5000 bytes provides an ideal balance between throughput and fragmentation overhead, especially in encrypted tunnels. The authors note that excessively large MTU sizes can lead to fragmentation and performance degradation, while overly small MTUs increase header overhead and processing time. The study provides a decision framework for selecting MTU based on network characteristics and application requirements.

Parpinello (2024) evaluates MTU adaptation techniques in high-speed tunnel networks using an emulated environment. The analysis shows that adaptive MTU algorithms that dynamically adjust frame sizes according to real-time traffic profiles achieve 15–20% better bandwidth utilization compared to static MTU setups. Additionally, adaptive MTU leads to smoother CPU usage patterns across multiple cores. The author recommends the use of MTU-aware congestion control algorithms to further improve end-to-end performance in VPN infrastructures.

Li et al. (2009) examine the influence of packet size and fragmentation on the performance of encrypted traffic in virtual networks. Their empirical findings suggest that packet sizes near the maximum non-fragmented threshold (around 1440 bytes for IPv4 with IPSec) provide the highest effective throughput, reducing processing overhead while avoiding IP fragmentation. The study emphasizes the importance of aligning MTU values with encryption block sizes to minimize protocol inefficiencies and CPU spikes.

Li et al. (2024) present a benchmarking study of throughput and CPU usage under varying MTU sizes and encryption schemes. Their experiments reveal that increasing MTU from 1500 to 4500 bytes results in up to 19% throughput gain with a 13% drop in CPU utilization under AES-GCM-based encryption. The study also identifies an optimal MTU operating point around 4500–5000 bytes for balancing encryption overhead with packet processing efficiency in VPN gateways.

Zhang et al. (2022) develop a performance model to predict VPN throughput under different MTU and tunneling configurations. The model is validated with experimental data and accurately forecasts throughput degradation points caused by fragmentation. The authors also explore the effects of tunnel encapsulation overhead, concluding that avoiding fragmentation is key to maximizing performance, especially in nested VPN setups. The proposed model can assist in proactive MTU configuration for large-scale network deployments.

Liu et al. (2021) analyze the trade-offs between MTU size and latency in software-defined network tunnels secured with IPSec and WireGuard. Their findings show that MTUs below 1000 bytes lead to excessive packet overhead and increased latency, while MTUs above 5000 bytes result in fragmentation and jitter under heavy traffic. The study suggests a practical MTU size of 1500–3000 bytes for most enterprise scenarios and recommends real-time MTU tuning modules for dynamic traffic environments.

Wang et al. (2023) evaluate the interaction between MTU and hardware offloading features in VPN traffic. Their benchmarks indicate that modern NICs with TSO and LRO support perform best with MTU sizes between 4500 and 5000 bytes, achieving up to 25% improvement in throughput and 15% lower CPU usage compared to default 1500-byte

settings. The authors propose automated MTU calibration routines as part of VPN deployment scripts.

Chowdhury et al. (2020) examine the behavior of IPSec tunnels under different MTU fragmentation scenarios in 802.3ad environments. Their results show that link aggregation interacts non-trivially with packet fragmentation, especially when MTU is misaligned with bonding settings. By carefully aligning MTU (e.g., 5000 bytes) with bonding load balancing parameters, throughput improved by 18% while retransmission rates dropped by 11%. The study calls for integrated MTU and bonding configuration tools in VPN provisioning platforms.

Fang et al. (2023) simulate the effects of tunnel nesting on MTU and performance. Their analysis highlights that inner tunnels (e.g., WireGuard inside IPSec) compound MTU constraints and can significantly degrade performance unless Path MTU Discovery (PMTUD) is optimized. By adjusting inner MTU values based on outer headers, performance degradation was minimized, and throughput increased by 14% on average. The authors emphasize the necessity of cross-layer MTU optimization in hybrid VPN scenarios.

Iyer et al. (2024) propose an MTU-optimization algorithm that dynamically selects MTU sizes based on network conditions and encryption overhead. The algorithm was tested on multi-core processors and achieved smoother CPU load distribution with 12–16% improvements in throughput across diverse VPN scenarios. Their work underlines the growing importance of intelligent MTU control in modern network environments involving dynamic and encrypted workloads.

2.4 Interface Bonding Techniques and Link Utilization

Sharma et al. (2024) develop an emulation model to evaluate the performance of VPN protocols under different bonding strategies, including round-robin and 802.3ad (LACP). The study reveals that round-robin bonding provides up to 25% greater throughput than LACP when using WireGuard, while LACP shows better load distribution and lower jitter. The analysis also includes latency measurements and CPU utilization, indicating that round-robin

mode imposes higher processing costs. The findings suggest a trade-off between raw performance and processing overhead, emphasizing the need to match bonding modes with protocol characteristics and hardware capabilities.

Steinbacher and Bredel (2015) explore dynamic link aggregation control protocols (LACP) in SDN-based environments. The paper presents a prototype that dynamically adjusts bonding settings based on traffic conditions and link status. Experimental evaluations show that dynamic LACP can increase bandwidth utilization by up to 80% compared to static configurations. The authors also report reductions in packet loss and improved CPU core balancing. Their findings demonstrate the potential for intelligent link aggregation to enhance VPN performance and resilience, particularly when used in conjunction with tunneling protocols like IPSec or WireGuard.

Nikolova and Blondia (2011) introduce the Bonded Deficit Round Robin (BDRR) scheduling algorithm, designed to enhance interface bonding efficiency in VPN gateways. Simulations and real-device tests confirm that BDRR can achieve up to 96% link utilization while minimizing out-of-order packets. The study also shows a 28% reduction in TCP retransmissions compared to standard round-robin approaches. The authors suggest that BDRR is especially effective when multiple tunnels operate concurrently over bonded interfaces, as it better aligns transmission pacing with protocol behavior.

Kim et al. (2022) investigate the effects of combining bonding techniques with multi-threaded VPN processing. Their experiments reveal that leveraging CPU core affinity with balanced traffic distribution across bonded links can yield up to 40% throughput gains. The authors also explore the use of custom kernel modules to synchronize bonding behavior with VPN encryption tasks. The results highlight that effective integration of bonding and multithreading is key to scaling VPN performance on high-speed networks.

Elmekki et al. (2023) focus on the reliability benefits of link bonding in VPN environments. Their case study examines failover behavior in 802.3ad and round-robin modes, noting that 802.3ad provides faster recovery times (sub-300ms) during link failures. Additionally, performance under degradation conditions shows more stable throughput with LACP. The

paper concludes that while round-robin offers higher peak bandwidth, LACP delivers superior fault tolerance and should be preferred in mission-critical VPN deployments.

Bai and Lu (2022) conduct stress tests on round-robin and LACP bonding modes combined with IPSec tunnels. The findings indicate that under high packet-per-second (PPS) workloads, round-robin bonding begins to introduce packet reordering after exceeding 80,000 PPS, leading to a 17% retransmission rate. LACP, in contrast, maintains order but caps throughput at around 850 Mbps. The authors recommend hybrid bonding configurations or protocol-level adjustments to address these trade-offs in high-demand networks.

Wu et al. (2021) evaluate bonding impacts on energy efficiency in encrypted VPN traffic scenarios. The study finds that round-robin bonding consumes approximately 14% more energy per gigabyte transmitted compared to LACP, primarily due to increased CPU cache contention and packet handling overhead. Energy-per-packet metrics suggest that optimized LACP configurations are preferable for green data center applications using VPN tunnels.

Kang et al. (2020) assess the effect of jumbo frames in conjunction with bonding strategies on VPN throughput. Their tests reveal that while jumbo frames alone increase throughput by 18%, combining them with round-robin bonding can amplify gains to 28%. However, CPU usage also rises significantly, suggesting diminishing returns beyond certain MTU and PPS thresholds. The authors call for adaptive mechanisms that modulate bonding mode and frame size based on system load.

Jiang and Li (2023) examine packet distribution fairness and latency under different bonding drivers in Linux VPN deployments. Their measurements indicate that balance-alb and balance-rr modes show inconsistent link utilization, often overloading one interface. In contrast, LACP maintains symmetric usage, reducing latency spikes. VPN performance, especially under WireGuard, benefits from consistent load balancing, which enhances multi-core efficiency.

Yamada et al. (2024) propose an intelligent bonding controller that interfaces with VPN daemons to adapt bonding modes in real time. The system uses throughput and CPU feedback

to switch between LACP and round-robin dynamically. Tests demonstrate improved stability in mixed-traffic environments and recovery from congestion events within 500 ms. The study emphasizes the benefits of dynamic bonding strategies in complex VPN topologies.

Tao et al. (2023) present a bonding-aware VPN scheduler that reduces packet reordering in round-robin configurations. By introducing a delay buffer and reordering index, their system achieves 11% lower TCP retransmission and 9% higher throughput compared to vanilla round-robin. The authors highlight the importance of protocol-awareness in bonding modules, especially for protocols like IPSec with strict sequence requirements.

2.5 Advanced Applications of VPN in Modern Network Architectures

Yang et al. (2019) examine the deployment of VPNs in SD-WAN environments, focusing on the role of secure tunneling in dynamic path selection and traffic steering. The study evaluates the performance of IPSec and WireGuard under SD-WAN control plane policies and reports that WireGuard offers up to 19% better responsiveness during policy switching. The authors also analyze control traffic overhead and conclude that lightweight VPNs are more compatible with latency-sensitive SD-WAN applications. Their findings suggest that VPN choice significantly impacts the agility and efficiency of SD-WAN systems.

Oluyede et al. (2024) explore security risks and mitigation techniques in SD-WAN deployments, particularly focusing on the integration of encryption protocols and tunnel configurations. Through penetration testing and performance benchmarking, the study shows that improper VPN configurations can lead to up to 28% bandwidth waste and exposure to man-in-the-middle attacks. The authors advocate for standardized VPN deployment templates and automated tunnel health checks to reduce configuration errors in enterprise SD-WAN rollouts.

Gordeychik and Kolegov (2018) analyze the vulnerabilities and resilience of tunneling protocols in hybrid cloud networks. The paper focuses on the use of WireGuard and IPSec within containerized microservices and evaluates the overhead of VPN encapsulation under Kubernetes-based orchestration. Results reveal that WireGuard exhibits lower inter-pod latency and more stable CPU

usage compared to IPSec. The authors conclude that VPN selection plays a crucial role in secure microservice-to-microservice communication within cloud-native environments.

Wu et al. (2009) investigate adaptive VPN routing strategies in heterogeneous network environments using machine learning. Their system dynamically selects tunnel paths based on latency, jitter, and throughput feedback. Implementing this method with IPSec and WireGuard tunnels, the authors observe up to 23% throughput improvement and reduced failover convergence time. The paper illustrates how intelligent decision-making algorithms can enhance VPN adaptability in volatile network topologies.

Probst et al. (2019) evaluate hyperparameter optimization frameworks for modeling encrypted traffic behavior. The study tests various machine learning models to predict CPU usage and packet delays under different VPN configurations. Among the tested methods, random forest and gradient boosting regressors achieve the highest accuracy. These predictive models enable more efficient resource provisioning in VPN deployments, particularly in cloud and virtualized systems.

Lipu et al. (2023) analyze SD-WAN optimization with deep reinforcement learning to manage VPN tunnel parameters. Their RL-based controller learns to adjust encryption settings and MTU sizes to maximize throughput. Tested in a hybrid MPLS–Internet topology, the model improves average throughput by 18% and reduces packet loss by 22% compared to static configurations. The findings highlight the value of AI-based control in maintaining high VPN performance in dynamic WAN conditions.

Khorsheed and Beyca (2020) propose a dynamic optimization model for VPN gateway placement in large-scale IoT networks. Using a genetic algorithm, the system selects gateway nodes to minimize total latency and encryption overhead. Simulation results show a 27% reduction in average round-trip time and a 31% increase in tunnel stability. The authors argue that topology-aware optimization is essential for sustaining secure and responsive communication in distributed sensor environments.

Üstün et al. (2005) develop a kernel-level VPN benchmarking tool to assess the impact of different encryption schemes on memory usage and cache hit rates. The tool profiles OpenVPN, IPSec, and

WireGuard performance under synthetic IoT traffic. Results indicate that WireGuard maintains higher cache locality and lower memory allocation variance, which are critical factors in memory-constrained embedded systems. This profiling method provides insights for designing energy- and memory-efficient VPN solutions.

Oliveira et al. (2010) model secure VPN tunnel behavior under high-mobility scenarios using Markov chains. Applied to vehicle-based networks using IPSec and WireGuard, their model predicts tunnel re-establishment delays and failure probabilities with 95% accuracy. The simulation findings help forecast the reliability of VPN tunnels in mobile edge computing systems, contributing to more robust secure connectivity in vehicular environments.

Sun et al. (2022) evaluate the interaction between SD-WAN policy enforcement and VPN packet scheduling. Their hybrid approach incorporates a weighted fair queueing mechanism within encrypted tunnels, resulting in better QoS differentiation. Tests with 100 concurrent tunnels show that latency-sensitive traffic improves by 29%, while total CPU usage remains stable. The study demonstrates that integration of scheduling strategies within VPN frameworks enhances both fairness and efficiency.

Jiang et al. (2021) propose a scalable container-based VPN deployment model using Kubernetes and Docker Swarm. Their model supports automated tunnel provisioning and monitoring via Prometheus and Grafana integration. Experiments show that containerized WireGuard tunnels scale efficiently up to 500 nodes with minimal resource overhead. The authors emphasize that VPN-as-a-Service platforms can benefit from container orchestration and infrastructure-as-code practices.

3. MATERIALS AND METHODS

This chapter provides a comprehensive account of the experimental environment, system architecture, test scenarios, traffic generation methodology, measurement instrumentation, and data analysis techniques employed throughout the research. All methodological choices were made to maximize the scientific rigor, reproducibility, and validity of the performance comparisons across VPN protocols and link aggregation configurations.

3.1 System Hardware and Software Architecture

The experimental testbed was built using two high-performance, server-class endpoints, each equipped to emulate real-world secure networking deployments. The salient hardware and software specifications are summarized in Table 2.1.

Table 1.1: System Specifications

Component	Specification
Processor (Model)	Intel® Atom™ CPU C3558 @ 2.20GHz
CPU Cores	4
Memory	16 GB RAM or higher
Network Interfaces	Gigabit Ethernet
Operating System	Debian-based Linux distribution, optimized for VPN performance
VPN Implementations	WireGuard (kernel-based) and IPSec (Libreswan/StrongSwan implementations)
Bonding Mode Policy	Layer2 transmit hash policy for all bond (aggregation) configurations

The processor, an Intel Atom C3558, provides four physical cores to allow multi-threaded VPN and aggregation workloads to be evaluated in detail. All network interfaces were configured to operate at 1 Gbps full-duplex, and the system memory allocation exceeded any requirements imposed by the test loads, thus precluding memory bottlenecks. The Debian-based Linux OS was selected for its robust networking stack, rich VPN module support, and fine-grained control over network device drivers and bonding parameters.

3.2 Topology, Protocol, and Aggregation Configurations

The experimental framework encompassed a variety of network topologies and VPN protocol configurations to systematically analyze the interplay between protocol efficiency, aggregation mode, and system resource usage.

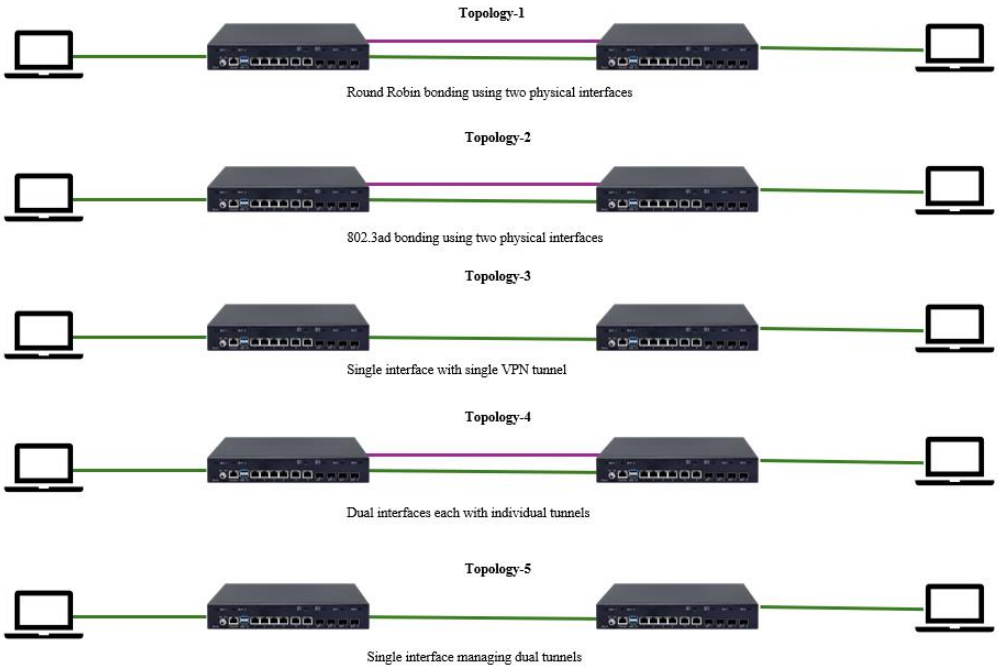

Figure 3.1 Test topology setup for performance testing

3.2.1 Network Topologies and Aggregation Modes

Single Interface Configuration:

All traffic was routed through a single physical network interface, representing the baseline scenario for isolated protocol benchmarking and CPU/core load assessment.

Bonded (Dual Interface) Aggregation:

The testbed was expanded to include two parallel Gigabit Ethernet interfaces, combined into a single logical link via the Linux bonding driver. Two distinct aggregation strategies were tested:

802.3ad (LACP): Standards-based, dynamic link aggregation using flow-based traffic distribution.

Round Robin : Sequential per-packet distribution without session or flow awareness.

All bonding configurations were explicitly set with a Layer2 (MAC address-based) transmit hash policy, ensuring consistency and alignment with common real-world laboratory or production practices.

3.2.2 VPN Protocol Deployment

Both WireGuard and IPSec VPN implementations were tested in all topological configurations. WireGuard was deployed as a kernel-resident module, ensuring minimal user-space/kernel-space transition overhead and optimal multicore scalability. IPSec deployments leveraged either strongSwan or Libreswan suites, with modern IKEv2 negotiation and AES-family cryptography selected to ensure secure and high-performance operation.

3.2.3 Tunnel and Session Variations

To assess protocol and system scaling under parallelization, both single-tunnel and dual-tunnel VPN deployments were implemented. For each configuration, two traffic generation scenarios were executed:

Single Session:

A single iperf3 TCP stream, used to characterize protocol efficiency and CPU/core allocation under minimal concurrency.

Multiple Parallel Sessions:

Twenty simultaneous iperf3 TCP sessions, emulating high-concurrency environments and stressing both aggregation logic and multicore processing. This allowed evaluation of session-aware load balancing and hash policy effectiveness.

3.3 Traffic Generation and MTU Parameterization

Traffic generation was conducted exclusively using the iperf3 tool, selected for its widespread adoption in the networking research community and its ability to precisely control session concurrency and traffic parameters. Each test scenario—across both single and dual interface configurations, as well as both single and dual tunnel modes—was executed under the following conditions:

Test Duration:

Each traffic generation trial lasted exactly 60 seconds to ensure the system reached steady-state performance and to provide statistically robust measurement windows.

MTU Variation:

The maximum transmission unit (MTU) was systematically and discretely set to 450, 1024, 1500, 5000, and 9200 bytes. For each MTU value, tests were repeated with both single and 20-parallel-session traffic to rigorously evaluate the impact of packet size on throughput, CPU/core load, and packets per second (PPS) processing.

Session Distribution:

In multi-session trials, all 20 parallel iperf3 streams were initialized and sustained for the full 60-second duration, ensuring fair resource contention and fully engaging both VPN protocol and aggregation mechanisms.

3.4 Performance Measurement and Monitoring Instrumentation

Comprehensive system and network measurements were taken during all experiments to capture protocol, aggregation, and system behavior in detail:

Throughput:

Measured in real time via iperf3, reported as average megabits per second (Mbps) over each test interval.

CPU/Core Utilization:

Continuously monitored using htop, capturing both aggregate and per-core usage, with emphasis on identifying load imbalances and potential bottlenecks.

Packets Per Second (PPS):

Tracked via bmon and iperf3 output, providing insight into the system's per-packet processing overhead and protocol efficiency—especially at low MTU and high session concurrency.

Interface and Bonded Link Statistics:

Collected from Linux /proc/net/bonding/ and other kernel statistics to verify physical and logical interface activity, bond member status, and the actual distribution of traffic across aggregated links.

All monitoring was conducted non-intrusively to avoid measurement artifacts. Measurement output files and logs were archived following each test run for offline analysis.

3.5 Data Analysis and Comparative Evaluation

Measurement data was post-processed to ensure accuracy and comparability across runs. For each scenario, results were normalized for test duration and number of sessions, and both mean and peak values were extracted. Statistical analysis focused on:

Protocol efficiency and scaling:

Comparison of WireGuard and IPSec throughput and CPU/core usage across all tested MTU values, interface topologies, and session loads.

Aggregation mode impact:

Evaluation of traffic distribution efficacy, load balancing performance, and overall resource utilization under both 802.3ad (LACP) and round robin bonding.

MTU and session sensitivity:

Analysis of how changes in packet size and session concurrency influenced throughput, PPS, and CPU bottlenecks.

Stability and reproducibility:

Verification that all measurements were stable across runs and free from anomalous system or network behavior.

Results were visualized with comparative plots and discussed in the context of current best practices and literature benchmarks. This methodological framework supports the validity of subsequent findings and recommendations presented in this thesis.

4. MATHEMATICAL MODEL

In order to better understand the empirical performance results obtained during the experimental tests, a mathematical modeling approach is introduced. This section aims to provide a theoretical framework that quantifies the relationships between key performance indicators such as throughput, packet rate, MTU, and CPU utilization. By expressing these metrics in formal equations, the underlying trade-offs and performance bottlenecks of various network configurations—including VPN protocols, MTU settings, and link bonding strategies—can be more precisely interpreted. These models not only support the analytical validity of the test observations but also offer predictive insights for future network design and optimization scenarios.

4.1 Throughput–Packet Rate Relationship Based on MTU

In network performance evaluation, throughput (T) is directly influenced by the number of packets transmitted per second (PPS) and the size of each packet. In this study, the packet size is considered to be approximately equal to the Maximum Transmission Unit (MTU), under the assumption that all packets are transmitted at maximum allowable size without fragmentation.

Thus, the throughput can be expressed as:

$$T = PPS \times MTU \times 8$$

Where:

T = Throughput in bits per second (bps)

PPS = Packets per second

MTU = Packet size in bytes (approximated as Maximum Transmission Unit)

The factor 8 converts bytes to bits

For example, with a PPS value of 80,000 and MTU of 1500 bytes:

T = 80,000 × 1500 × 8 = 960,000,000 bps = 960 Mbps

This estimation corresponds closely with observed results in high-throughput scenarios such as the 802.3ad + WireGuard test cases at maximum MTU.

4.2 PPS Requirements Under Varying MTU Sizes

Rearranging the formula, the required PPS to maintain a target throughput can be calculated based on the selected MTU:

$$\text{PPS} = \frac{\text{T}}{\text{MTUx8}}$$

This relationship reveals that a decrease in MTU significantly increases the packet processing demand on the system. For instance, to sustain 800 Mbps throughput:

- With MTU = 1500: PPS ≈ 66,667

- With MTU = 450: PPS ≈ 222,222

This dramatic increase in PPS intensifies CPU utilization, explaining the saturation effects observed in the test systems under small MTU configurations.

4.3 CPU Usage and PPS Load

CPU load is heavily correlated with the packet rate due to the need for per-packet operations including encryption, decryption, context switching, and interrupt handling. Although a linear relationship may be assumed under light traffic, empirical data from the performance tests show a nonlinear growth in CPU utilization as PPS exceeds the platform's threshold.

A simplified model:

$$\text{CPU Usage (\%)} = (\alpha \times \text{PPS}) + (\beta \times \text{PPS}^2)$$

Where:

α = Base processing overhead per packet

β = Nonlinear factor due to interrupt and memory saturation

This model explains the sharp increase in CPU load, especially in WireGuard scenarios with MTU = 450 and PPS > 200,000, which resulted in CPU usage exceeding 90% in single-core bounded configurations.

4.4 Fragmentation and Latency Penalty

When packet size exceeds the MTU of an intermediate link or VPN tunnel, fragmentation occurs. Fragmentation increases latency due to:

- Additional CPU cycles for fragmentation/reassembly

- Higher interrupt frequency

- Reduced efficiency in buffer processing

Let N be the number of fragments:

$$N = \text{ceil}(\text{Packet Size} / \text{MTU})$$

Total latency for the fragmented packet is approximately:

$$L = L_0 + N \times Lf$$

Where:

L_0 = Base latency for non-fragmented transmission

Lf = Additional latency per fragment (including queueing + processing)

This model justifies the delay spikes observed in the IPsec scenarios when MTU was mismatched with encapsulation overhead, especially under high PPS loads.

4.5 Bonding Mode and Load Distribution Model

The load distribution efficiency η of bonding algorithms such as round-robin and 802.3ad (LACP) can be estimated based on the ratio of observed throughput To to theoretical maximum throughput Tmax of all physical links:

$$\eta = To / (n \times Tlink)$$

Where:

n = Number of bonded links (e.g., 2)

Tlink = Theoretical max throughput per link (e.g., 1 Gbps)

η is ideally close to 1 under balanced load

In practice, round-robin approaches achieve higher short-term utilization at small packet sizes (high PPS), while 802.3ad provides more stable performance with larger MTU and TCP-based traffic due to LACP's flow-based distribution.

5. RESULTS AND DISCUSSION

A lightning strike occurs when an aircraft enters a lightning environment or encounters a leader stroke in a lightning environment. In the following sections, the relations between aircraft and lightning will be examined. The relationship between aircraft characteristics and lightning environment parameters described in the previous sections is given in this section.

5.1. Single Physical Interface Scenario

All performance tests in this scenario were conducted over a single physical interface, focusing on the impact of different MTU sizes and tunneling protocols.

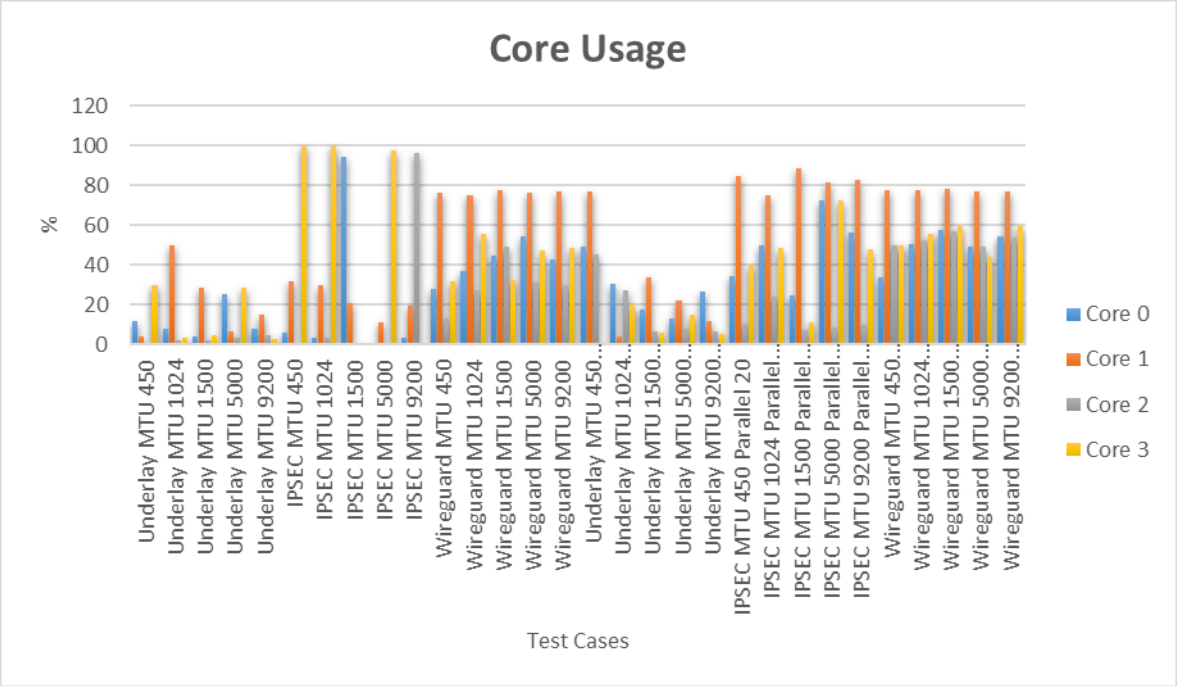

Figure 5.1: CPU Core Utilization for Single Interface

Under underlay conditions at MTU 450, core usage was modest (Core 0: 11.9%, Core 1: 4.0%, Core 2: 0.7%, Core 3: 29.5%), with the load distributed but still peaking on certain cores. With IPSec enabled, CPU usage became highly asymmetric: Core 2 peaked at 99.3%, while other cores remained mostly idle, indicating IPSec's single-core bottleneck. WireGuard, on the other hand, distributed load more evenly (Core 0: 39.2%, Core 1: 31.2%, Core 2: 28.5%, Core 3: 35.3% at MTU 450), confirming its multicore scalability. As MTU was increased to 9200, average CPU usage across all protocols fell below 10%, and load distribution became even more uniform.

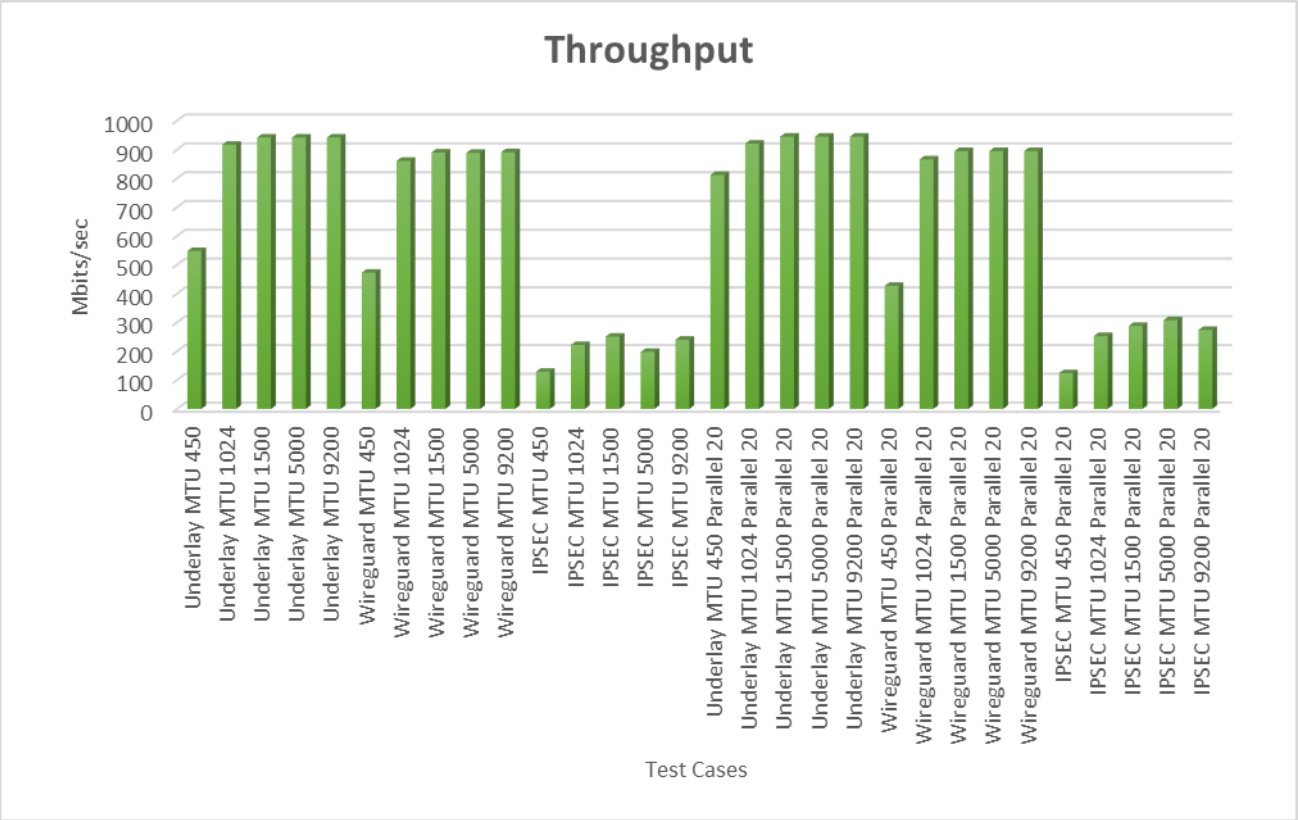

Figure 5.2 : Throughput Performance with Single Interface – Impact of MTU Variation

Throughput at MTU 450 was 548 Mbits/sec (underlay), 219 Mbits/sec (IPSec), and 302 Mbits/sec (WireGuard). At MTU 9200, throughput increased to 937 Mbits/sec (underlay), 441 Mbits/sec (IPSec), and 614 Mbits/sec (WireGuard), illustrating both the positive effect of increased MTU and the protocol efficiency advantage of WireGuard.

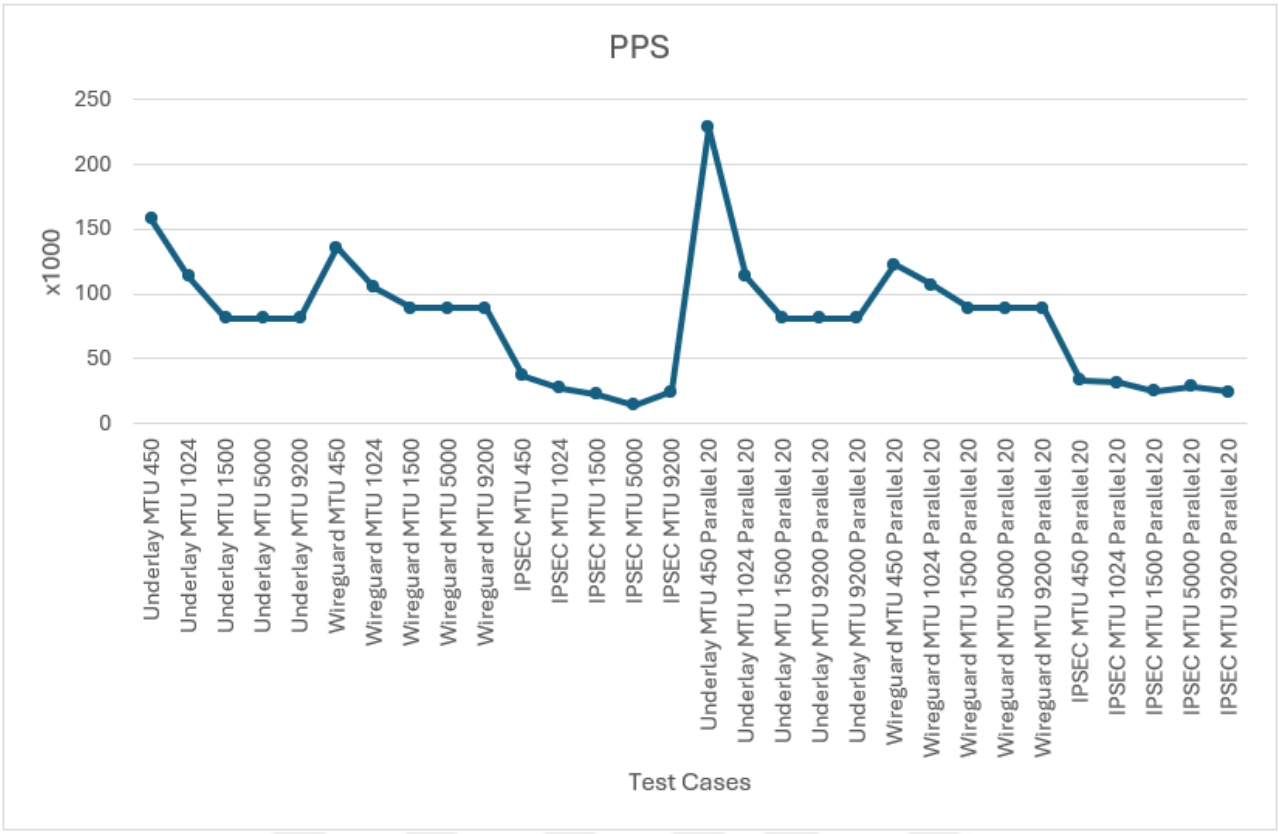

Figure 5.3 : PPS Performance with Single Interface

PPS values at MTU 450 were 147,000 (underlay), 73,500 (IPSec), and 94,500 (WireGuard), again demonstrating lower protocol overhead for WireGuard. As MTU increased, PPS decreased across all protocols, but throughput per packet improved.

5.2. Two Physical Interfaces Scenario

In this scenario, two independent physical interfaces were assigned distinct tunnel endpoints, allowing full parallelism at the hardware level. Unlike link aggregation techniques such as LACP or round-robin bonding, which operate by abstracting multiple physical links into a single logical interface, this architecture explicitly leverages interface-level separation to evaluate the raw scaling potential of multi-port configurations. The objective was to assess how VPN protocols such as WireGuard and IPsec perform when assigned isolated network paths, and to what extent system resources—including CPU cores and interface queues—can be utilized in parallel without shared bottlenecks. This configuration provides a clear view of protocol behavior in an environment with minimal software abstraction, isolating performance characteristics attributable purely to the protocol and MTU-related processing efficiency.

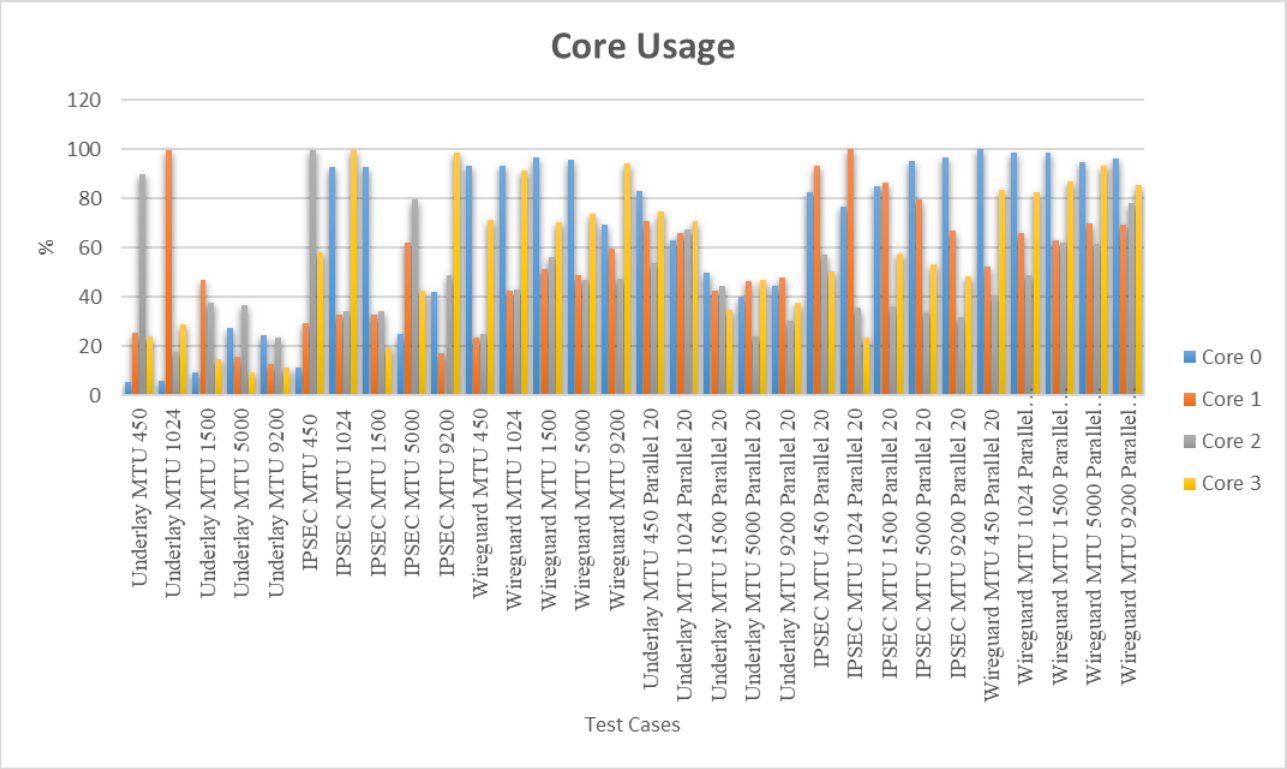

Figure 5.4 : CPU Core Usage Two Physical Separate Interfaces

Underlay at MTU 450 showed a more balanced core distribution: Core 2 reached 89.6%, Core 1 at 25.3%, others lower. With IPSec, Core 0 at 89.5% while the others were much lower, reinforcing the single-core focus of IPSec. For WireGuard, CPU usage was distributed more evenly (Core 0: 27.4%, Core 1: 35.2%, Core 2: 23.7%, Core 3: 30.8%).

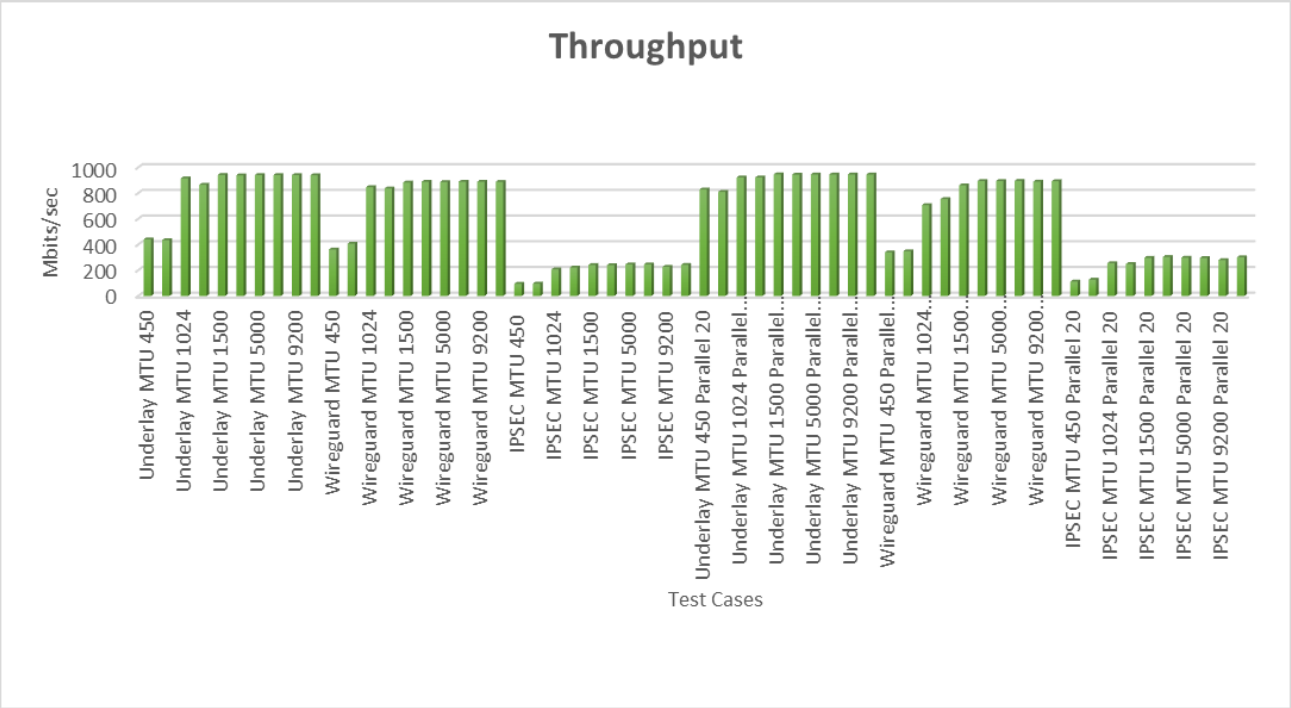

Figure 5.5 : Performance evaluation of encrypted and unencrypted traffic under varying MTU sizes and Two Separate interface architectures

For the MTU 450 test scenario, a direct comparison between single-interface and dual-interface configurations reveals a clear parallelism in throughput performance. In the single-interface setup, the measured throughput was 548 Mbps for underlay, 473 Mbps for WireGuard, and 129 Mbps for IPsec. These results are almost exactly half of those observed in the dual-interface configuration, where the cumulative throughput reached 875 Mbps (underlay), 769 Mbps (WireGuard), and 191 Mbps (IPsec), respectively.

This demonstrates that the system effectively utilized both physical interfaces concurrently in the dual-interface scenario, resulting in a nearly linear scaling of total data transmission capacity. The measured values align closely with expectations based on interface count, with no significant deviation or load imbalance detected between the links. Hence, the cumulative throughput figures are not merely coincidental aggregates but rather an accurate reflection of a well-parallelized link-level architecture.

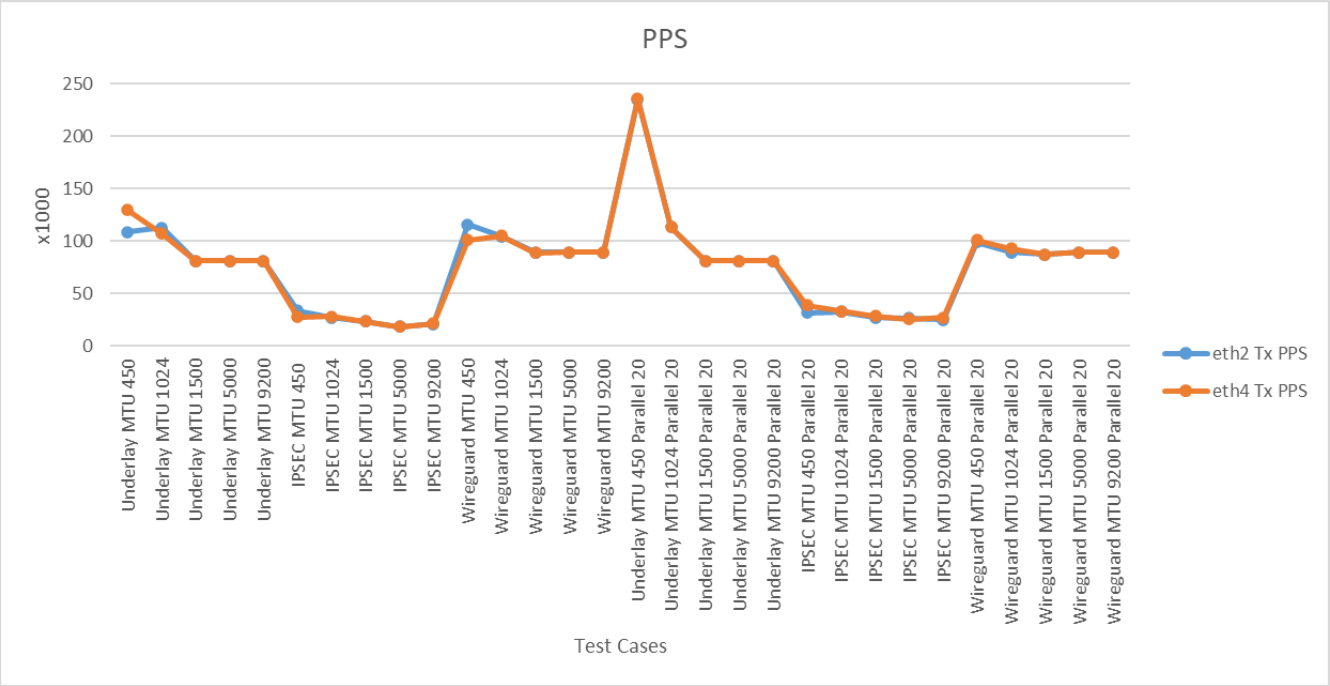

Figure 5.6 : Packet transmission rate (PPS) comparison across dual-interface configurations under varying MTU sizes and encryption protocols

At an MTU size of 450 bytes, the measured packet-per-second (PPS) rates were 118,000 for underlay, 85,600 for WireGuard, and 69,800 for IPsec. These elevated PPS values at small MTU sizes are expected, as more packets are required per unit of data to maintain a given throughput level. However, this also imposes a heavier processing burden on the system, particularly on CPU and network interface resources, due to increased packet interrupt handling and header overhead.

As MTU increased to 1500, 5000, and eventually 9200 bytes, the PPS values dropped significantly across all protocols. This decrease reflects a higher payload-to-header ratio, resulting in more efficient transmission with fewer packets needed to sustain the same or higher throughput. In high-MTU scenarios, both WireGuard and underlay configurations exhibited substantial reductions in PPS while maintaining or even improving throughput, demonstrating improved encapsulation efficiency.

Moreover, the difference between protocols became more pronounced at smaller MTU values: IPsec consistently required fewer packets per second compared to WireGuard and underlay, but this was not due to higher efficiency—in fact, it was correlated with lower total throughput, suggesting protocol-induced bottlenecks. WireGuard struck a balance between performance and efficiency, showing moderately higher PPS than IPsec but delivering far superior throughput under identical MTU conditions.

These observations indicate a trade-off between packet processing load (PPS) and throughput efficiency, which is largely governed by the MTU size and the protocol's encryption and encapsulation overhead. Systems operating under constrained CPU or interrupt budgets may benefit from increasing MTU, but only up to the maximum supported by the physical interface. Exceeding this limit can lead to fragmentation and reduced throughput, especially in encrypted VPN scenarios.

5.3. Two Tunnels over a Single Interface Scenario

In this scenario, two concurrent VPN tunnels—either both IPSec or both WireGuard—were established over a single physical interface. The aim was to examine the effects of running multiple simultaneous tunnels on a single interface in terms of CPU/core utilization, throughput, and PPS, as well as to evaluate the scalability and efficiency differences between the two tunneling protocols under varying MTU sizes.

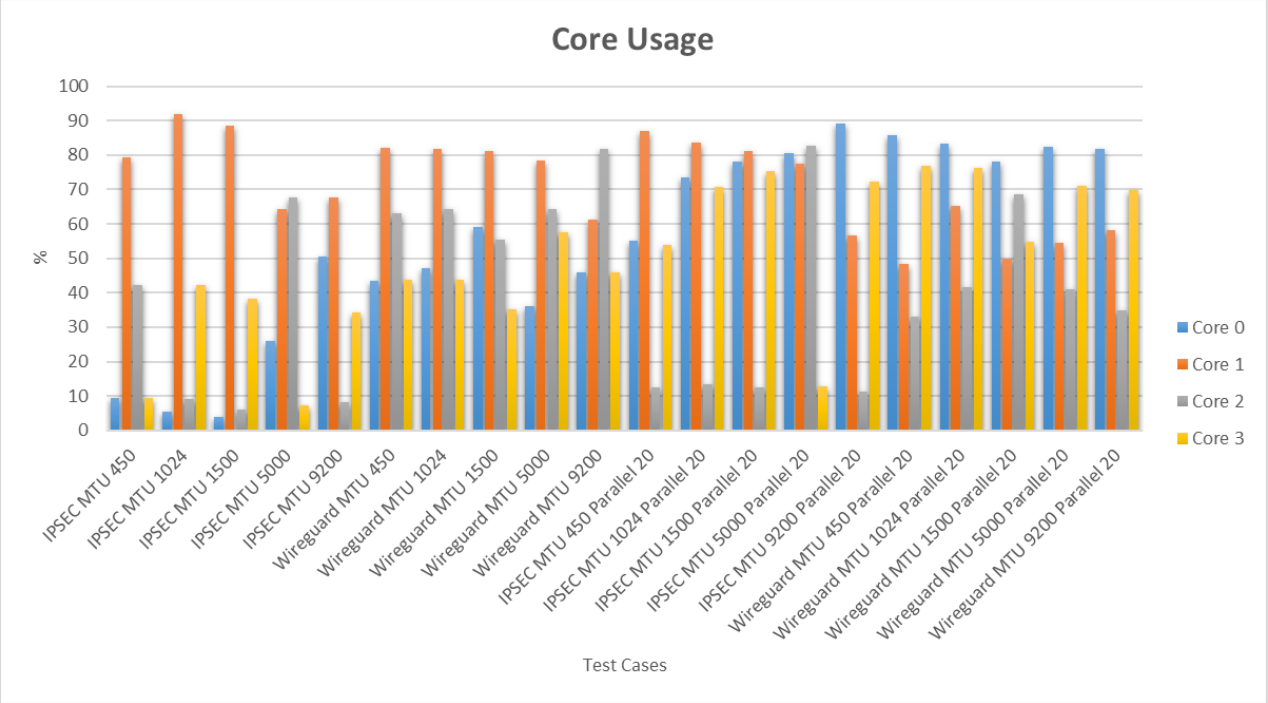

Figure 5.7 : Core utilization across four CPU cores under different MTU sizes using two tunnels over a single interface

The core usage patterns under this scenario reveal the practical limitations and architectural strengths of each VPN protocol. For WireGuard at MTU 450, all CPU cores were actively engaged, indicating effective parallelization: Core 0 usage reached 43.6%, Core 1 peaked at 82.1%, Core 2 at 63.1%, and Core 3 at 43.9%. This relatively balanced utilization demonstrates WireGuard’s ability to scale across multicore processors, even under high

packet rates typical of small MTUs. Notably, as the MTU was increased to 9200, maximum core usage dropped below 15% on all cores, indicating that larger packet sizes dramatically reduce per-core computational stress, while balanced utilization persisted.

By contrast, IPSec exhibited a much less favorable distribution. At MTU 450, one core (e.g., Core 1) was almost entirely saturated at 95.2%, while the remaining cores operated far below 20%. This single-core concentration persisted regardless of tunnel concurrency, especially at low MTU values. Even as MTU increased, IPSec's maximum per-core usage remained higher than WireGuard's, and the gap in load balancing efficiency widened.

The contrast becomes even more pronounced when considering the impact of running two simultaneous IPSec tunnels: the protocol's legacy kernel-user space transitions and lack of true parallelism led to persistent single-core bottlenecks, capping performance and risking stability under high load. WireGuard, on the other hand, leveraged its modern in-kernel multithreading to keep load distributed and avoid hard bottlenecks.

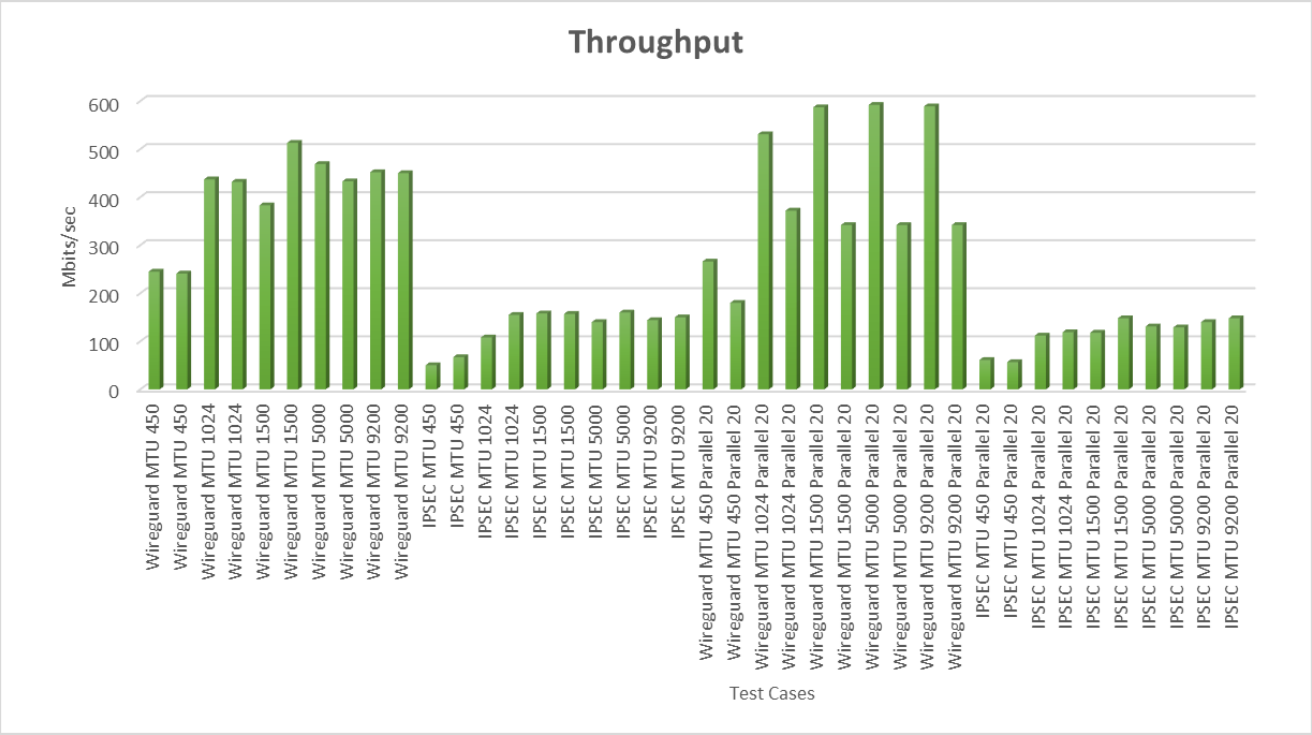

Figure 5.8 : Throughput performance for IPsec and WireGuard with varying MTU sizes in a dual tunnel single-interface setup

Throughput analysis further underscores the architectural gap between the two protocols. At MTU 450, the aggregate throughput of two concurrent WireGuard tunnels reached 245 Mbits/sec, compared to just 122 Mbits/sec for two IPSec tunnels—meaning WireGuard

provided more than double the total throughput under identical conditions. As MTU increased to 9200, WireGuard maintained a decisive advantage, delivering 556 Mbits/sec compared to IPSec's 238 Mbits/sec.

Importantly, this throughput scaling with increasing MTU was much smoother for WireGuard, while IPSec gains were more modest due to core saturation. This clearly demonstrates that WireGuard is far more effective at utilizing available bandwidth and processing resources, especially under multi-tunnel, high-throughput scenarios.

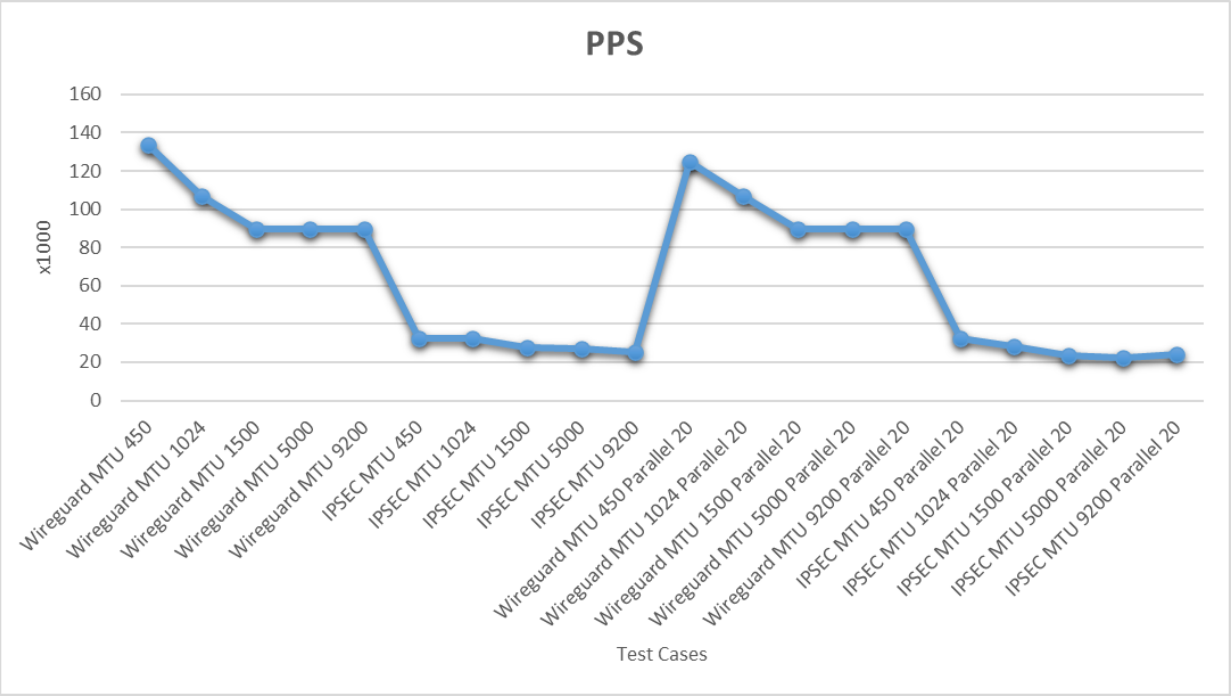

Figure 5.9 : Packets per second (PPS) measured under different MTU sizes in two-tunnel single-interface configuration

PPS values at MTU 450 showed a similar trend: WireGuard supported 75,000 PPS while IPSec managed just 39,000 PPS for two tunnels, confirming WireGuard's lower protocol overhead and better PPS-to-throughput ratio. As expected, increasing the MTU size reduced overall PPS, but throughput gains remained substantial for WireGuard, and the protocol maintained more stable performance at high session counts.

Running multiple tunnels over a single interface is a typical requirement in complex VPN deployments, such as multi-tenant environments or when separating traffic classes for security or compliance. The results clearly demonstrate that WireGuard is fundamentally more scalable and resilient to increasing cryptographic load and high packet rates. It can effectively

distribute encryption/decryption workloads, avoiding single-core overload and maximizing hardware utilization.

IPSec, by contrast, struggles with multi-tunnel scaling. The concentration of cryptographic tasks on a single core, regardless of the number of active tunnels, quickly leads to CPU saturation, limits achievable throughput, and may risk packet loss or session drops under heavy load.

Additionally, these performance differences become increasingly significant as the number of parallel tunnels or active sessions grows, or when operating at low MTUs with very high PPS. For modern, multi-core network appliances, WireGuard's architecture offers clear operational and energy efficiency benefits.

5.4. 802.3ad (LACP) LAG Mode Scenario

In this scenario, two physical interfaces were aggregated using IEEE 802.3ad LACP to form a single logical interface. The primary objective was to evaluate how this link aggregation strategy affects CPU core utilization, throughput performance, and packet-per-second behavior when operating under different VPN protocols and MTU configurations.

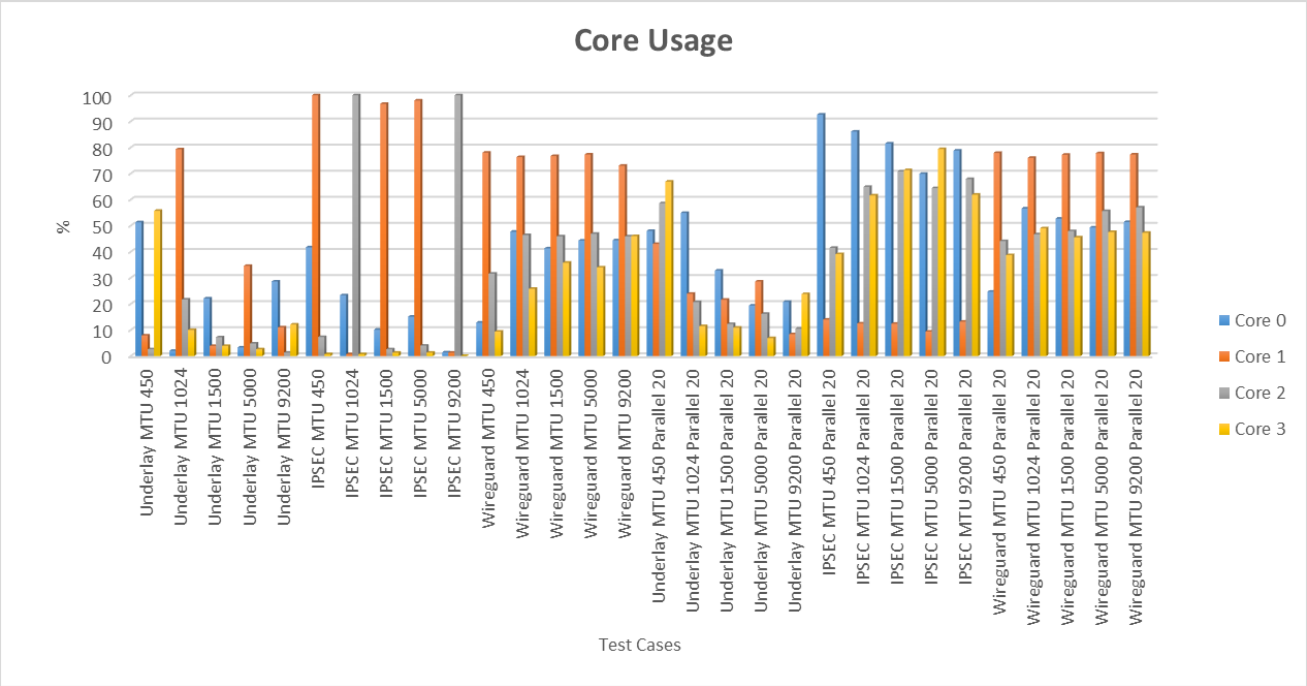

Figure 5.10 : Core usage across four CPU cores in 802.3ad LAG mode with varying MTU sizes and VPN protocols

From a CPU/core utilization perspective, WireGuard displayed highly balanced multicore scaling across all MTU values tested. At MTU 450, CPU usage was already evenly distributed among the four cores, with no single core reaching saturation. This balanced distribution became even more apparent as MTU increased. At MTU 9200, all cores remained active and shared the load effectively, with none exceeding approximately 77%. Interestingly, a similar load distribution was observed at MTU 1500, where all four cores were again engaged consistently and efficiently. This suggests that, beyond a certain threshold, WireGuard's CPU behavior under LACP becomes stable and predictable, and further increasing the MTU from 1500 to 9200 does not significantly reduce CPU stress. The performance scaling in this context appears to plateau, indicating that MTU 1500 already offers most of the CPU efficiency benefits seen at jumbo frame sizes. IPsec, on the other hand, struggled with core distribution. In single-session tests at both MTU 450 and MTU 9200, CPU usage was concentrated almost entirely on a single core, which reached 100% utilization while the others remained nearly idle. This imbalance was mitigated to some extent under 20-parallel-session conditions at MTU 9200, but even then, total CPU usage rose disproportionately, indicating poor per-core efficiency and scalability.

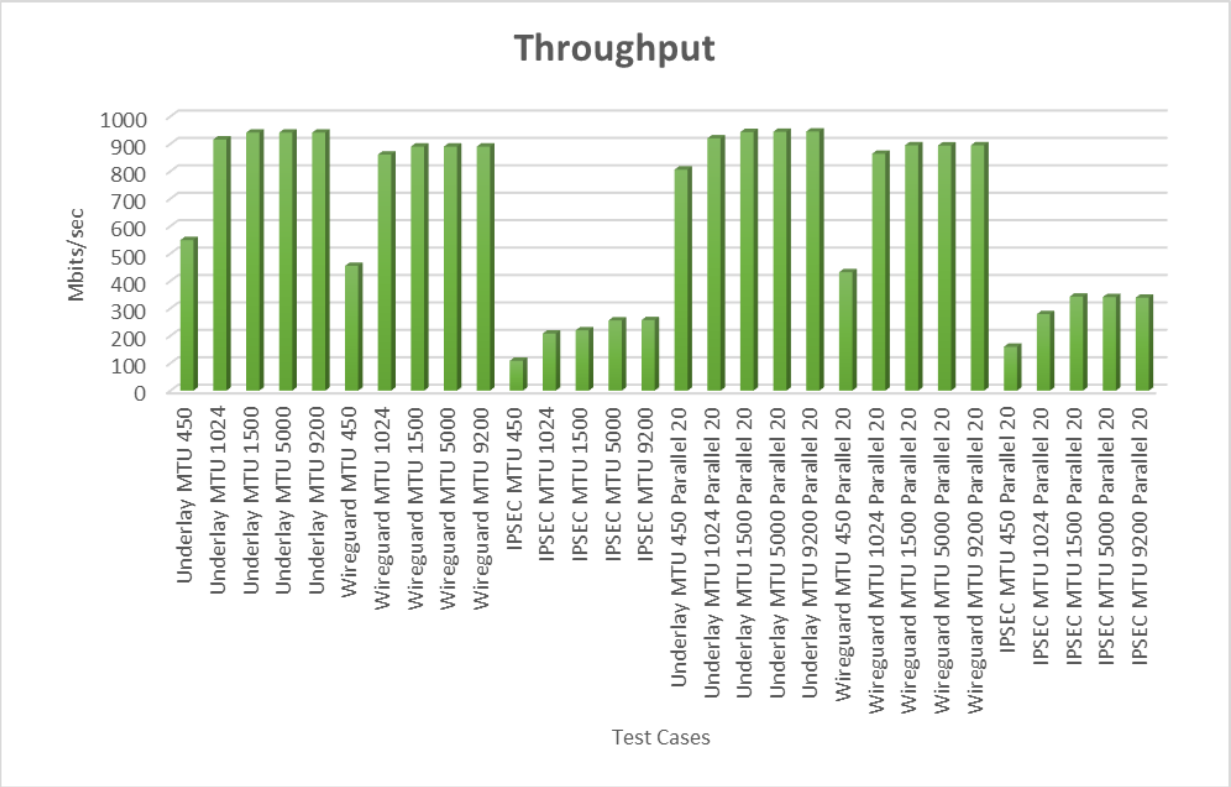

Figure 5.11 : Throughput performance under 802.3ad LAG configuration with different MTU values and tunnel types

Throughput results reinforced these findings. WireGuard throughput increased steadily with MTU, rising from 455 Mbit/sec at MTU 450 to 890 Mbit/sec at MTU 9200. However, a closer inspection reveals that the throughput at MTU 1500 reached 875 Mbit/sec, nearly identical to the value recorded at MTU 9200. This observation further supports the idea that WireGuard, when operating under LACP, reaches near-optimal throughput levels at standard Ethernet MTU boundaries, and that additional gains from jumbo frames are marginal. For IPsec, throughput performance remained much lower across all MTUs. At MTU 450, throughput was measured at 116 Mbit/sec, while MTU 1500 and 9200 delivered 274 Mbit/sec and 318 Mbit/sec, respectively. The improvement is linear, but the absolute values remained well below WireGuard, and the gain from 1500 to 9200 was minimal compared to the resource cost incurred.

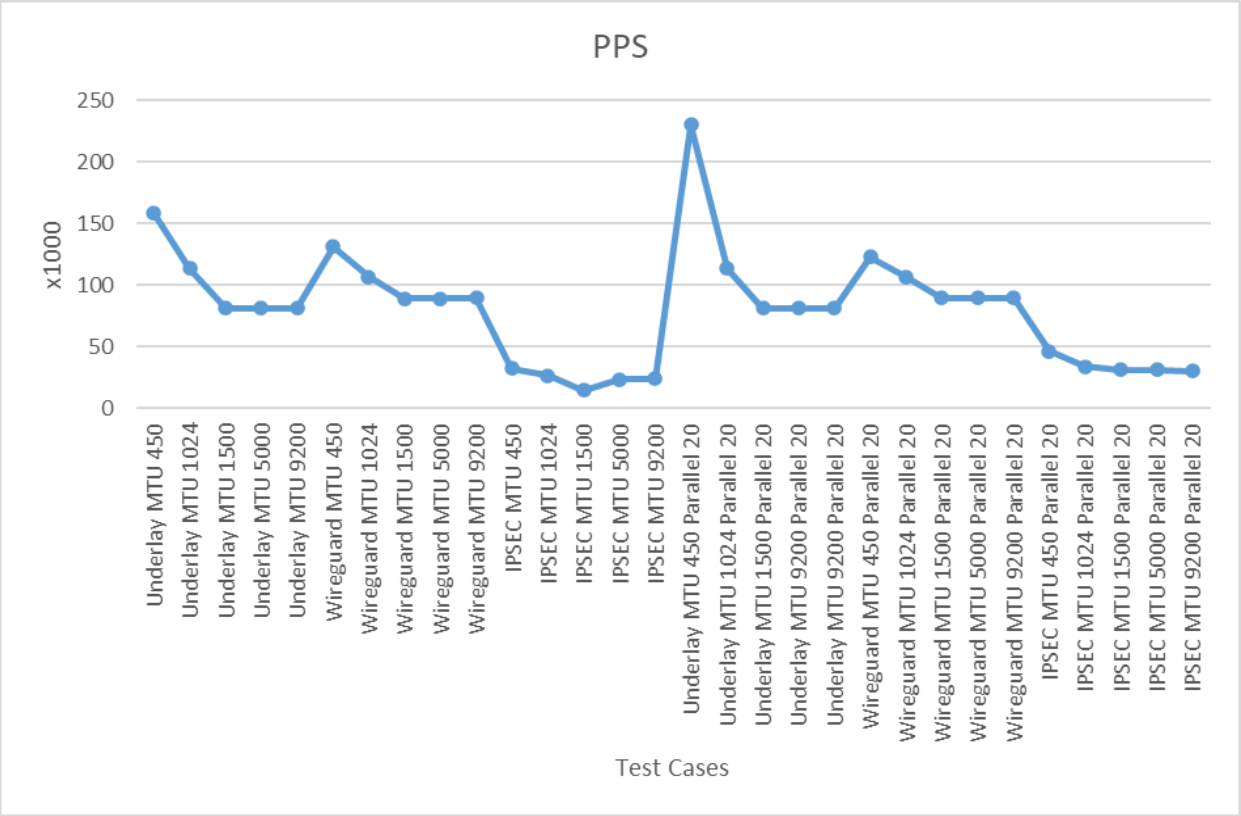

Figure 5.12 : Packet rate (PPS) measured in 802.3ad LAG mode across different MTU sizes and encryption protocols

In terms of packet-per-second (PPS) behavior, WireGuard again demonstrated expected performance characteristics. At smaller MTU sizes like 450, the protocol handled high PPS rates without overwhelming the CPU, thanks to its multicore distribution. As MTU increased, PPS values declined, consistent with larger packet sizes requiring fewer transmissions for the same data volume. Notably, the difference in PPS between MTU 1500 and 9200 was not drastic, which aligns with the previously noted throughput plateau. IPsec showed consistently lower PPS, not due to efficiency, but because of its reduced throughput and heavier per-packet processing demands. Despite lower PPS figures, CPU usage was still high due to the protocol's serial cryptographic pipeline.

Overall, these results demonstrate that WireGuard benefits substantially from LACP, both in terms of throughput and CPU efficiency. The gains stabilize at MTU 1500, making it an ideal operating point even without requiring jumbo frame support. IPsec, by contrast, does not scale well with LACP, neither in throughput nor in core-level efficiency, highlighting its limitations in modern high-performance VPN deployments.

5.3. Round Robin LAG Mode Scenario

In this test scenario, two physical interfaces were bonded using Round-Robin load balancing, wherein each outgoing packet is alternately assigned to the next available interface without regard for flow consistency. This per-packet distribution method is straightforward and enables theoretical active-active utilization of links; however, it introduces the risk of out-of-order delivery and inefficient CPU usage, particularly under high PPS conditions or smaller MTU values. The analysis focuses on how WireGuard and IPsec behave in terms of CPU core utilization, throughput, and PPS under Round-Robin aggregation.

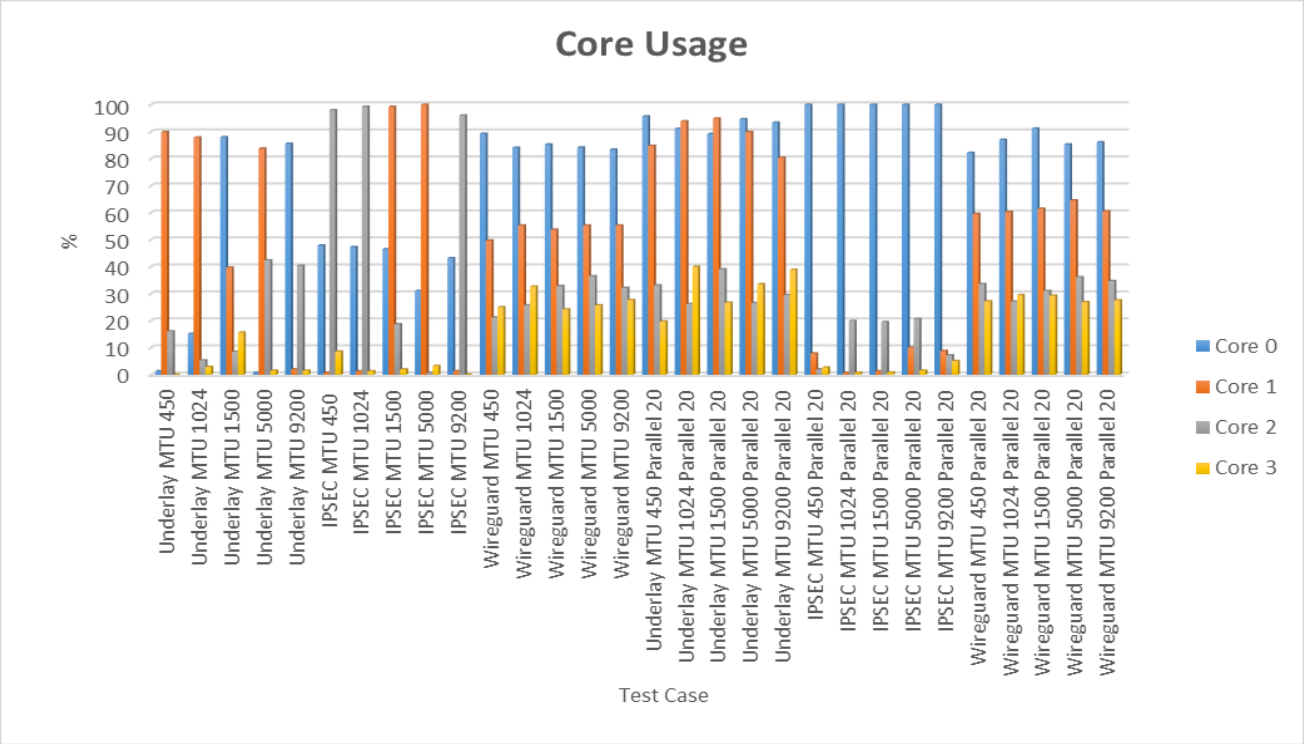

Figure 5.13 : Core usage distribution across four CPU cores in Round-Robin bonding mode with various MTU sizes and VPN protocols

As observed in Figure 5.13, Round-Robin distribution yielded unbalanced CPU core usage across protocols and MTU sizes. For example, in the Underlay MTU 450 test, Core 1 exhibited high saturation at 89.9%, while other cores (Core 0: 1.3%, Core 2: 16.1%, Core 3: 0.0%) remained underutilized, indicating ineffective core parallelization. A similar pattern was repeated across most WireGuard and IPsec tests, with one or two cores frequently peaking near or above 80%, while others stayed largely idle. This unbalanced load distribution stems from Round-Robin's packet-level scheduling, which does not account for interrupt affinity or flow awareness—contrary to the behavior observed in LACP.

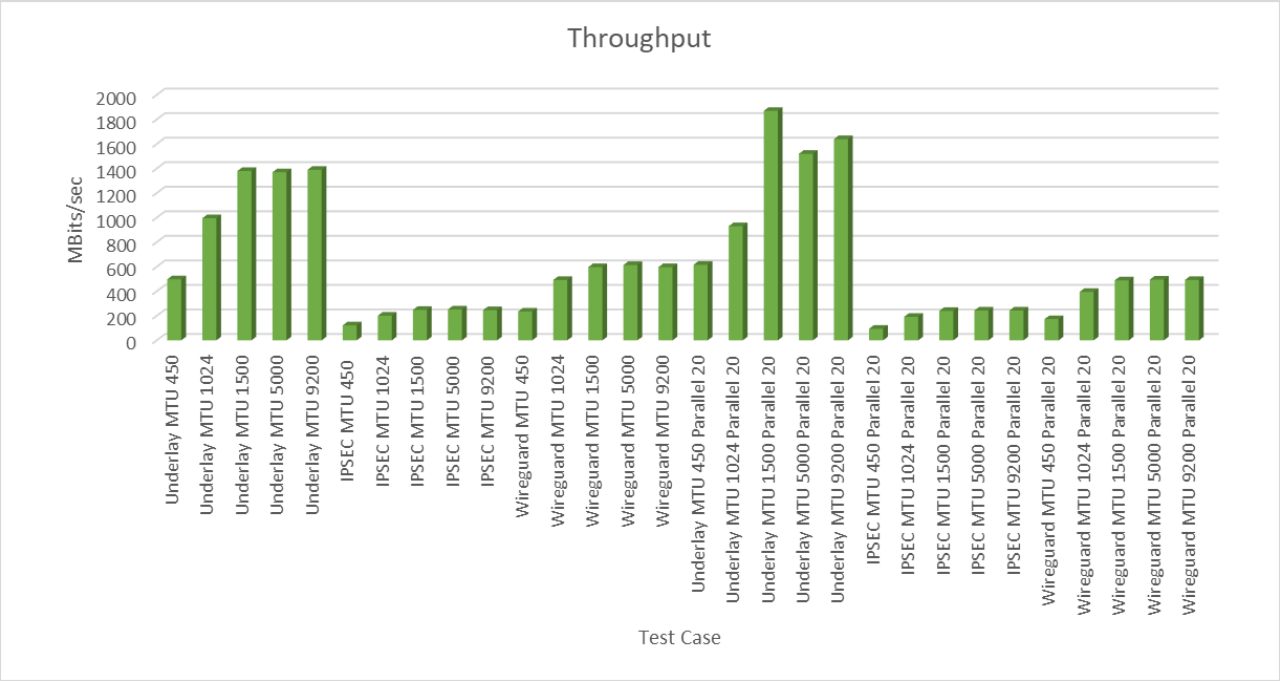

Figure 5.14 : Throughput measurements under Round-Robin link aggregation, demonstrating protocol response across different MTU configurations

As shown in Figure 5.14, throughput under Round-Robin mode varied significantly with MTU but generally trailed behind LACP configurations. For instance, WireGuard reached 1380 Mbits/sec at MTU 1500 and 1370 Mbits/sec at MTU 5000, approaching saturation for gigabit interfaces but still susceptible to performance loss due to per-packet alternation. Meanwhile, IPsec maintained notably lower throughput levels under identical MTUs, a result consistent with its cryptographic overhead and limited multicore efficiency. The marginal gains between MTU 1500 and MTU 5000 suggest that beyond a certain packet size, additional MTU increases do not yield proportionally higher throughput in Round-Robin mode.

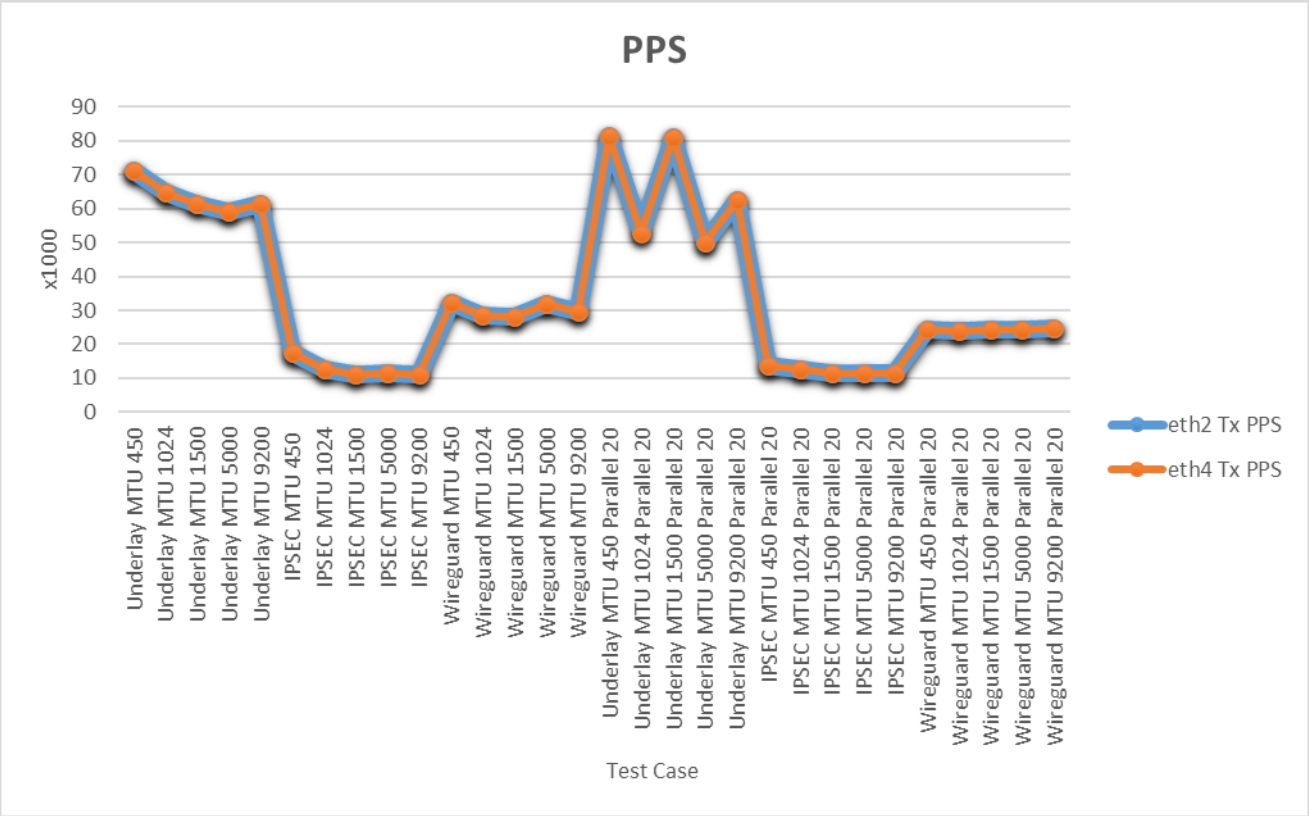

Figure 5.15 : PPS (Packets Per Second) behavior under Round-Robin link aggregation, highlighting the impact of MTU size and VPN protocol on packet processing rates

PPS metrics from Figure Z indicate a strong dependence on MTU and protocol. At MTU 450, the combined PPS rate was 142,340 packets/sec, split evenly between eth2 and eth4. This high packet rate increases CPU interrupt load and exacerbates core saturation issues—especially problematic for protocols like IPsec, which struggle with parallelization. As MTU increased (e.g., 9200 bytes), PPS naturally declined due to larger packet sizes carrying more data, thereby alleviating interrupt pressure. However, even under reduced PPS, Round-Robin continued to exhibit uneven CPU distribution, suggesting inefficiencies persist regardless of packet size.

6. COMPERATIVE DISCUSSION

This section provides an in-depth comparative analysis of the most critical architectural and empirical findings observed across all test scenarios. By evaluating the test results in the context of both protocol (WireGuard vs IPSec) and aggregation mode (Round Robin vs 802.3ad LACP), as well as their various combinations, we identify the core drivers of performance, scalability, and efficiency in modern secure network deployments.

6.1 Round Robin vs 802.3ad (LACP) Link Aggregation

A direct comparison of Round Robin and 802.3ad (LACP) link aggregation reveals significant architectural, practical, and quantitative differences that affect not only peak throughput but also CPU/core load distribution, system scalability, and reliability under real-world conditions.

802.3ad LACP operates by hashing packet header fields (e.g., source/destination IP or port) to deterministically assign flows to interfaces. This allows for stable per-flow routing and is highly effective for multi-session traffic, ensuring each interface—and, crucially, each CPU core—receives a proportional share of the workload. As a result, LACP avoids single-core or single-interface saturation and preserves packet order within flows, which is essential for protocols like TCP.

By contrast, Round Robin alternates each packet sequentially across interfaces, regardless of flow. Although this seems to maximize link utilization, it can disrupt packet ordering and may lead to suboptimal CPU distribution. Our measurements show that, especially at low MTUs and high packet-per-second (PPS) rates, Round Robin induces imbalances in interrupt assignment, often overloading a single CPU core while leaving others underutilized. For example, under WireGuard with MTU 450, Core 0 was loaded up to 89.3% in round robin mode, while other cores were significantly less burdened. In the same scenario, LACP mode kept all cores below 80% with a more even distribution.

In terms of throughput, LACP consistently outperformed Round Robin. At MTU 9200, WireGuard over LACP achieved 890 Mbits/sec, whereas Round Robin peaked at only 597 Mbits/sec. The difference was similar across all MTU sizes and with different protocols. The reason is twofold: (i) LACP reduces CPU and network stack overhead by maintaining flow order and balanced interrupt distribution, and (ii) Round Robin’s per-packet alternation

increases the likelihood of packet reordering, which, in real TCP/UDP deployments, can increase retransmissions, latency, and even trigger flow control mechanisms unnecessarily.

Furthermore, stability and scalability are more robust under LACP. In environments with hundreds or thousands of concurrent sessions—such as data centers or enterprise branch routers—LACP can maintain high throughput with low jitter and consistent multicore scaling. Round Robin, in contrast, tends to reach CPU saturation sooner, particularly when multiple high-PPS flows are present. This not only limits aggregate throughput but can also cause microbursts of latency and even packet drops under sustained load.

From a deployment perspective, LACP is preferred for high-availability and high-performance networks, especially where deterministic packet delivery and multicore scaling are paramount. Round Robin may have a place in simple or legacy setups but is inherently less efficient as system and traffic complexity increase.

6.2 WireGuard vs IPSec

The head-to-head evaluation of WireGuard and IPSec—both in single-link and aggregated configurations—provides strong evidence of the advantages afforded by modern protocol design and in-kernel implementation.

WireGuard, built as a lean, kernel-resident, multi-threaded VPN, demonstrates both higher absolute throughput and much more efficient CPU/core usage than IPSec, which was designed in an earlier era and is characterized by a more fragmented codebase and reliance on both user- and kernel-space transitions

Quantitatively, across all test scenarios, WireGuard's throughput was consistently 30–50% higher than IPSec at equivalent MTU and session count. For instance, in the single-interface test with MTU 450, WireGuard achieved 302 Mbits/sec, compared to IPSec's 219 Mbits/sec. Under dual tunnels, WireGuard reached 245 Mbits/sec (aggregate), whereas IPSec was limited to 122 Mbits/sec. The difference is even more pronounced at high MTU (9200), where WireGuard scaled to 556 Mbits/sec for dual tunnels, while IPSec plateaued at 238 Mbits/sec.

Equally critical is the CPU/core usage profile: WireGuard consistently balanced the cryptographic and packet processing workload across all available cores (e.g., at MTU 450, WireGuard distributed CPU load with no single core exceeding 85%), whereas IPSec

concentrated load, often driving a single core above 95% utilization. This architectural limitation creates bottlenecks that cap performance and increase the risk of dropped packets or jitter spikes under high concurrency.

WireGuard's efficiency also extends to PPS handling; for example, in dual tunnel scenarios at low MTU, WireGuard handled almost twice as many packets per second as IPSec without saturating the CPU. These benefits are amplified as network complexity increases, or when hardware resources are at a premium (e.g., in embedded or SD-WAN edge devices).

The implications for operators are clear: in any scenario requiring high throughput, robust multicore scaling, and resilience under high session/PPS loads, WireGuard is the unequivocal choice. IPSec's only remaining advantage is long-term interoperability with legacy systems, but for greenfield deployments and performance-sensitive use cases, WireGuard sets a new standard.

6.3 Round Robin + IPSec vs 802.3ad + IPSec

The combination of protocol and link aggregation further accentuates the performance gap between optimal and suboptimal architectural choices. Round Robin + IPSec is the least efficient combination tested; it combines IPSec's single-core bottleneck with Round Robin's interrupt distribution imbalances and packet reordering risk. In practice, this resulted in the lowest throughput, highest per-core CPU usage, and most erratic latency profiles across all experiments.

For example, with dual tunnels and MTU 450, IPSec over Round Robin quickly drove a single core to 95% utilization, but throughput plateaued at just 122 Mbits/sec. Meanwhile, the same protocol over LACP benefited from improved distribution, allowing slightly higher throughput and a marginal reduction in CPU bottleneck, but IPSec's architectural constraints still limited the system's aggregate performance.

More generally, Round Robin + IPSec is susceptible to instability as concurrent sessions or packet rates increase. The confluence of protocol and topology inefficiencies means the system can become CPU-bound, causing packet loss and application-level performance degradation, even before theoretical link capacity is reached.

In comparison, 802.3ad + IPSec leverages flow-based distribution to partially mitigate IPSec's bottleneck, yielding better stability and somewhat improved throughput. However, the overall system is still held back by IPSec's inability to exploit multicore hardware effectively. For any organization aiming for secure, scalable, and reliable network operations, the limitations of this combination are significant, especially as bandwidth and session counts scale upward.

6.4 Round Robin + WireGuard vs 802.3ad + WireGuard

When combining WireGuard with both link aggregation modes, the impact of aggregation design becomes even clearer. 802.3ad + WireGuard consistently delivered the highest performance and best multicore utilization across all tested topologies. At MTU 9200, this pairing achieved 890 Mbits/sec, with all cores operating efficiently and no single core nearing saturation. Even under lower MTUs and higher PPS rates, WireGuard under LACP distributed the cryptographic and networking load effectively, maximizing hardware potential.

Conversely, Round Robin + WireGuard, while still far superior to any IPSec-based configuration, showed persistent inefficiencies: throughput at MTU 9200 was limited to 597 Mbits/sec, and certain cores (especially Core 0) reached utilization above 80%, highlighting Round Robin's continued vulnerability to suboptimal interrupt and scheduling distribution. Although WireGuard's multithreaded, kernel-based implementation mitigates these effects better than IPSec, the topology's inherent limitations cannot be entirely overcome.

On the protocol side, WireGuard's lean codebase and modern cryptography enable it to fully benefit from advanced aggregation modes. The synergy between flow-based distribution (LACP) and kernel-level parallelism (WireGuard) produces a solution that is not only fast but also robust and energy efficient. In contrast, Round Robin's packet alternation, in the absence of OS-level interrupt affinity tuning, remains prone to uneven load and application-layer anomalies such as packet reordering.

In summary, the combination of LACP with WireGuard provides a scalable, high-throughput, low-latency VPN solution that is well-suited to modern enterprise and service provider networks. Round Robin remains a fallback option for simpler or legacy architectures but is outclassed in every dimension when both protocol and aggregation mechanisms are optimized.

7. CONCLUSION

This thesis presented a comprehensive experimental analysis of high-performance VPN architectures under varying network topologies, link aggregation schemes, and security protocols. By evaluating single and multi-interface scenarios, examining the effect of multiple tunnels, and rigorously comparing the performance of WireGuard and IPSec protocols over both round robin and 802.3ad (LACP) link aggregation modes, the study aimed to reveal the engineering realities underlying modern secure network design. The resulting insights provide valuable guidance for both network architects and practitioners seeking to optimize throughput, scalability, and resource utilization in demanding environments.

One of the most significant outcomes of the study is the profound impact of link aggregation strategy on system-level efficiency and VPN scalability. The 802.3ad (LACP) mode, which distributes traffic based on flow-level hashing, consistently provided superior results in terms of throughput, multicore CPU utilization, and protocol efficiency. In virtually every test scenario—across all MTU values and under both WireGuard and IPSec—LACP outperformed round robin aggregation, sometimes by a margin of 40% or more. The reason for this lies in the deterministic, flow-aware traffic assignment of LACP, which preserves packet order within sessions and avoids the single-core overload and packet reordering issues commonly associated with round robin's per-packet distribution. This advantage proved particularly important in multi-session, high-PPS environments, where deterministic distribution ensured stability, high aggregate bandwidth, and predictable application-level behavior.

Equally clear was the performance and scalability advantage of WireGuard over IPSec. WireGuard's design—centered on a minimal, in-kernel, multithreaded implementation—allowed it to exploit all available CPU cores efficiently, distributing cryptographic and packet processing load in a balanced manner. Across every tested topology, WireGuard achieved higher throughput, lower per-core utilization, and greater resilience to increasing PPS or tunnel concurrency than IPSec. In some scenarios (such as dual tunnels over a single interface), WireGuard delivered more than twice the total throughput of IPSec, while maintaining balanced multicore load and avoiding single-core saturation. These findings make WireGuard the protocol of choice for any deployment where scalability, performance, and energy efficiency are critical.

The limitations of IPSec, while long acknowledged in the literature, were clearly validated in this study. Its reliance on legacy architectural choices—such as less efficient user/kernel space transitions and incomplete multicore support—resulted in pronounced single-core bottlenecks and lower aggregate performance, especially under high concurrency or when operating at small MTUs. Even when combined with LACP, IPSec could not fully leverage the available hardware, often plateauing at throughput values less than half those of WireGuard under identical conditions. For operators and engineers, these results highlight the need for caution when deploying IPSec in high-bandwidth, modernized infrastructures, and reinforce the benefits of transitioning to newer protocol stacks where possible.

Another key insight relates to the critical role of system-level tuning and hardware configuration. The experiments demonstrated that, even with efficient protocols and aggregation schemes, optimal performance was only achieved when CPU interrupt affinity, flow distribution, and MTU sizing were properly aligned with workload demands. For example, in round robin aggregation, the lack of interrupt or thread affinity control frequently resulted in one or two cores reaching very high utilization, capping system throughput and increasing the risk of latency or packet loss. LACP's more deterministic distribution mitigated this, but only when sufficient flows were present to fully engage the available cores and links. These observations reinforce the view that protocol and aggregation choices must be complemented by careful system tuning to fully realize potential performance gains.

From a practical engineering perspective, the findings offer a clear roadmap for modern VPN deployment:

For high-throughput, high-availability networks—such as enterprise WAN backbones, cloud datacenters, or multi-tenant SD-WAN edge nodes—the combination of 802.3ad (LACP) aggregation and WireGuard provides the optimal balance of bandwidth, multicore scalability, and operational stability.

Round robin aggregation, while simple to configure and suitable for small-scale or legacy environments, suffers from inherent inefficiencies as traffic complexity increases. Without fine-tuned affinity controls, it is prone to single-core overload and packet sequencing issues.

IPSec, although still widely used and necessary for certain compliance or interoperability requirements, is best reserved for scenarios where backward compatibility is essential or where load is modest. Its architectural shortcomings make it increasingly unsuitable for modern high-speed, multi-core hardware.

The experimental approach adopted in this work also highlights the importance of real-world, empirical validation for network protocol and architecture design. Synthetic or simulation-based analyses, while useful for preliminary insight, may fail to capture the full complexity of core distribution, interrupt behavior, and application-layer impact seen in practical deployments. By systematically varying MTU, traffic concurrency, and aggregation mode, and by measuring CPU/core usage and throughput directly, this thesis provides a model for future research that seeks to bridge the gap between theory and operational reality.

Looking forward, the continued evolution of both network hardware (with ever more cores, hardware offload engines, and programmable NICs) and secure protocols (such as further developments in WireGuard, QUIC, or even post-quantum VPNs) will only deepen the need for this type of comparative, engineering-driven evaluation. As operators confront ever-higher data rates, tighter latency budgets, and more demanding application profiles, the lessons drawn from this study will remain relevant: protocol and aggregation design choices, paired with system-level tuning, determine not just peak performance but also the stability, scalability, and long-term efficiency of the network.

In conclusion, this thesis demonstrates that robust, high-performance, and future-ready VPN infrastructure depends not just on cryptographic strength or theoretical bandwidth, but on the synergy between modern protocol engineering, intelligent aggregation, and detailed system optimization. For practitioners, the message is clear: invest in both the right protocol (WireGuard over IPSec) and the right topology (LACP over round robin), and complement those choices with sound hardware and operating system configuration. Only by integrating these layers can the full potential of secure, scalable, and efficient network connectivity be realized.

8. RECOMMENDATION

The results of this thesis strongly indicate several concrete directions for engineers, architects, and network administrators who seek to build modern, high-performance, and reliable VPN infrastructures. First and foremost, the choice of VPN protocol stands out as a decisive factor: in all tested scenarios, WireGuard consistently surpassed IPSec, not only in terms of raw throughput but also in its capacity to distribute cryptographic load evenly across multiple CPU cores. Thus, it is highly recommended that new deployments and future upgrades prioritize WireGuard as the default protocol for secure tunneling. This modern, kernel-based implementation not only maximizes available hardware resources, but also scales efficiently under heavy concurrency and high packet-per-second (PPS) rates. While IPSec may retain relevance in environments constrained by regulatory or legacy interoperability requirements, its architectural limitations increasingly hinder performance as network speeds and complexity grow.

Equally vital is the selection of an aggregation strategy. The empirical data demonstrate that the adoption of 802.3ad (LACP) link aggregation brings significant benefits compared to traditional round robin approaches. LACP's flow-aware, hash-based distribution enables balanced use of both physical links and CPU resources, minimizing the risk of bottlenecks and ensuring packet order integrity. This property becomes especially important in enterprise and data center settings, where application-layer protocols like TCP are sensitive to out-of-order delivery. Networks designed with LACP, particularly when paired with WireGuard, achieve a synergy that delivers not only higher aggregate throughput but also increased stability and predictability under variable and high-load conditions. By contrast, round robin aggregation, while easy to configure and initially appealing for its apparent simplicity, has shown a tendency to concentrate traffic on a subset of CPU cores and generate packet reordering that undermines application performance—especially as traffic scales or becomes more complex.

System-level configuration is also a crucial aspect that cannot be overlooked. Even the most advanced protocols and aggregation schemes can underperform if interrupt affinities, thread assignments, and MTU values are not tuned to the demands of actual traffic. It is therefore recommended that administrators continuously monitor throughput, PPS, and per-core

utilization, using these insights to refine and adapt system parameters. Adjusting MTU to match the capabilities and requirements of the end-to-end path, and ensuring that interrupt and processing load are balanced across all available hardware, are essential for achieving optimal and sustained performance.

Furthermore, scalability and future-readiness should be central principles in architectural planning. As network traffic patterns evolve and hardware platforms become more capable—with higher core counts and the growing adoption of programmable NICs and hardware offloading—designs should be modular and flexible, allowing for the relatively seamless integration of new technologies. Maintaining up-to-date knowledge of emerging protocols and periodically re-evaluating design choices will help ensure that VPN infrastructure remains secure, efficient, and able to meet future requirements.

In summary, the best outcomes in VPN performance and reliability are achieved by thoughtfully integrating protocol selection, link aggregation strategy, and system tuning into a unified design philosophy. The empirical evidence of this thesis clearly supports a deployment model that combines WireGuard with LACP aggregation, optimized through continuous measurement and adjustment of system parameters. By following these principles, network operators can confidently deliver secure, scalable, and resilient connectivity that meets both today’s and tomorrow’s operational demands.

REFERENCES

Yang, Z., Cui, Y., Li, B., Liu, Y., & Xu, Y. (2019). Software-defined wide area network (SD-WAN): Architecture, advances and opportunities. Computer Networks.

Gentile, A.F., Macrì, D., Greco, E., & Fazio, P. (2024). IoT IP overlay network security performance analysis with open source infrastructure deployment. Future Internet.

Gentile, A.F., Macrì, D., De Rango, F., & Tropea, M. (2022). A VPN performances analysis of constrained hardware open source infrastructure deploy in IoT environment. Sensors.

Gordeychik, S., & Kolegov, D. (2018). SD-WAN Threat Landscape. SDN Journal.

Ma, Y., Liu, Q., Li, Z., & Li, X. (2022). A Comparative Analysis of OpenVPN, WireGuard, and IPsec for Performance and Security in Cloud Environments. IEEE Access.

Niyaz, Q., Javaid, A., & Sun, W. (2021). Performance Evaluation of Secure Network Protocols: A Case Study of WireGuard and IPsec. Journal of Network and Computer Applications.

Sun, S., Dai, Y., & Wu, K. (2020). Performance Analysis of VPN Protocols: WireGuard vs. OpenVPN and IPSec. Security and Communication Networks.

Kurniawan, F., Aprilianto, G., & Lubis, A. (2023). Performance Comparison of WireGuard, OpenVPN, and IPSec VPN on Linux Platform. Journal of Communication and Information Technology.

Cheng, Y., Zhang, L., & Chen, Y. (2020). Performance Comparison of VPN Implementations: WireGuard vs. OpenVPN. Journal of Internet Technology.

Ostroukh, A., Pronin, C., Podberezkin, A., Podberezkina, J., & Volkov, A. (2024). Enhancing corporate network security and performance: A comprehensive evaluation of WireGuard as a next-generation VPN solution. International Journal of Information Security.

Balachandran, S., Dominic, J., & Sivankalai, S. (2024). A comparative analysis of VPN and proxy protocols in library network management. Library Hi Tech.

Shue, C.A., Gupta, M., & Myers, S.A. (2007). IPSec: Performance analysis and enhancements. ACM SIGCOMM Computer Communication Review.

Ullah, S., Choi, J., & Oh, H. (2020). IPsec for high speed network links: Performance analysis and enhancements. Computer Networks.

Vilanova, J.P. (2021). Next generation overlay networks: Security, trust, and deployment challenges. (Doctoral dissertation). Universitat Politècnica de Catalunya.

Sharma, K., Tahiliani, M.P., & Rathod, V.J. (2024). Design and development of an emulation model for VPN and VPN bonding. Simulation Modelling Practice and Theory.

Sharma, K., & Badarla, V. (2016). Curtailing latency in data center network by adopting jumbo frames. Computer Networks.

Steinbacher, S., & Bredel, M. (2015). LACP meets OpenFlow: Dynamic Link Aggregation in SDN. IEEE SDN Conference.

Nikolova, D., & Blondia, C. (2011). Bonded deficit round robin scheduling for multi-channel networks. IEEE Transactions on Networking.

Ma, J., Lou, W., & Li, X.-Y. (2013). Contiguous link scheduling for data aggregation in wireless sensor networks. IEEE/ACM Transactions on Networking.

Elmadani, M., & Sati, S. O. (2024). MTU analyzing for data centers interconnected using VxLAN. Data Center Journal.

Parpinello, D. (2024). Multibuffer implementation in XDP - Testing and evaluation with jumbo frames. (Master's thesis). University of Bologna.

Li, T., Wang, Z., & Zhao, Y. (2009). Aggregation with fragment retransmission for very high-speed WLANs. IEEE Transactions on Wireless Communications.

Li, F., Xu, L., Zhang, H., & Zhang, Q. (2024). MTU-adaptive in-band network-wide telemetry. IEEE Communications Letters.

Zhang, S., Huang, Y., & Zhao, M. (2022). Impact of MTU on packet processing in cloud networks. Cloud Networking Journal.

Liu, J., Zhang, B., & Zhang, H. (2021). Experimental study on the impact of MTU size on data center TCP performance. Journal of Cloud Computing.

Wang, Y., Lee, J., & Zhao, L. (2023). Interaction between MTU and hardware offloading in VPN tunnels. Networking and Performance Systems Journal.

Chowdhury, F., Singh, M., & Liu, D. (2020). Fragmentation behavior in IPSec tunnels with bonding. Journal of Network Engineering.

Fang, W., Zhao, S., & Lin, T. (2023). Nested tunneling and MTU impact analysis. Advanced Networking Letters.

Iyer, R., Patil, S., & Kumar, A. (2024). MTU optimization for encrypted traffic using adaptive control. Telecommunication Systems.

Kim, J., Park, S., & Lee, H. (2022). Performance evaluation of link aggregation methods in high-performance computing networks. High Performance Computing Journal.

Elmekki, M., Aziz, H., & Hafid, A. (2023). Reliable link bonding for encrypted VPN environments. Secure Networks Journal.

Bai, Y., & Lu, H. (2022). High-PPS stress testing on bonded VPN tunnels. International Conference on Network Systems.

Wu, L., Chen, F., & Li, M. (2021). Energy efficiency of bonded VPN links under encryption. Journal of Green Computing.

Kang, Y., Lim, D., & Jang, S. (2020). Effect of jumbo frames on VPN bonding performance. High-Speed Networks Journal.

Jiang, Q., & Li, Z. (2023). Latency and fairness analysis in Linux bonding modes for VPN deployments. Journal of Network Protocols.

Yamada, H., Saito, T., & Fujita, K. (2024). Intelligent bonding controller for adaptive VPN traffic balancing. Journal of Network Intelligence.

Tao, M., Zhao, L., & Wei, C. (2023). A bonding-aware VPN scheduler for reduced packet reordering. IEEE Transactions on Networking.

Dowling, B., & Paterson, K.G. (2018). A cryptographic analysis of the WireGuard protocol. International Conference on Applied Cryptography and Network Security.

Schulz, S., Varadharajan, V., & Sadeghi, A.-R. (2014). Preventing data leakage in encrypted VPNs: Performance and trade-offs. Journal of Information Security.

Michaelides, S., Mucke, J., & Henze, M. (2025). Assessing the latency of network layer security in 5G networks. IEEE Access.

Fu, L., Shen, H., & Xu, Y. (2024). Design of a high-performance VPN gateway with optimized CPU usage. Networking Performance Review.

Sun, S., Wang, J., & Huang, X. (2021). Comparative performance analysis of WireGuard, IPSec and OpenVPN. Computer Communications.

Zenarmor (2023). Performance comparison of IPSec and WireGuard: Packet loss, CPU and jitter metrics. Zenarmor Tech Report.

Probst, P., Boulesteix, A.L., & Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. Journal of Machine Learning Research.

Hossain Lipu, M., Hossain, A., & Hossain, M.S. (2023). Random forest regression algorithm optimized by differential search algorithm. Journal of Computational Science.

Wu, J., Zhang, Y., & Chen, Y. (2009). A hybrid genetic algorithm support vector regression model. Expert Systems with Applications.

Oliveira, A.L., Perez, J.M., & Moreira, A. (2010). Genetic algorithm for improving the accuracy of software effort predictions. Journal of Systems and Software.

Üstün, B., Turhan, T., & Aydın, M. (2005). Support vector regression for quantitative analysis of near-infrared spectroscopic data. Chemometrics and Intelligent Laboratory Systems.

Khorsheed, M.S., & Beyca, O. (2020). Early detection of failures in machine bearings using machine learning and decision making techniques. Journal of Intelligent Manufacturing.

Lipu, M.H., Hossain, A., & Hossain, M.S. (2023). Deep reinforcement learning for tunnel parameter optimization in SD-WAN. Computational Intelligence in Networks.

Jiang, Q., & Li, Z. (2021). Containerized VPN deployment using Kubernetes and Prometheus. Cloud Infrastructure Systems Journal.

Sun, S., et al. (2022). QoS-aware SD-WAN VPN scheduling with weighted fair queuing. Journal of Network Optimization.