Vokasi Unesa Bulletin of Engineering, Technology and Applied Science (VUBETA)
https://journal.unesa.ac.id/index.php/vubeta
Vol. 2, No. 3, 2025, pp. 376~386 DOI: 10.26740/vubeta.v2i3.38786
ISSN: 3064-0768

Revisiting Parasitic Computing: Ethical and Technical Dimensions in Resource Optimization

Godfrey Oise[1*], Clement Nwabuokei[2], Richard Igbunu[3], Prosper Ejenarhome[4]
[1] Department of Computing, College of Computing and Natural Science, Wellspring University, Benin City, Edo State, Nigeria.
[2,3] Department of Computer Science, Delta State College of Education, Mosugar, Delta State.
[4] Department of Computer Science, Delta State University, Abraka, Delta State.

Article Info

Article history:

Received February 06, 2025
Revised April 14, 2025
Accepted August 03, 2025

Keywords:
Parasitic Computing
TCP/IP Protocols
Distributed Computing
Ethical Implications
Consent-Aware Networking

ABSTRACT

Parasitic computing is a provocative concept enabling one system to offload computational tasks to remote hosts without explicit consent by exploiting communication protocols such as TCP/IP. While initially demonstrated as a conceptual hack, its implications for distributed computing, ethics, and resource optimization remain underexplored in modern contexts. This study revisits the original parasitic computing model, focusing on operational feasibility, technical efficiency, and ethical boundaries. We implement a Python-based simulation that encodes logical operations (AND, OR) into TCP packets by manipulating checksum fields—a core mechanism of the parasitic approach. We conducted over 6,000 packet transmissions across various network latency conditions using loopback and LAN environments to measure success rates, response times, and failure thresholds. Results show that logical operations can succeed under low-latency conditions with over 94% accuracy, but performance degrades sharply under higher round-trip times, dropping below 70%. These findings suggest parasitic computing may be technically viable within tightly controlled environments but face significant limitations in broader network applications. The researchers critically examine ethical considerations, emphasizing the risks of unauthorized computation, resource exploitation, and potential security breaches. This study contributes a reproducible methodology and empirical data, offering a renewed perspective on parasitic computing's technical boundaries and future potential. It further calls for responsible experimentation and proposes hybrid models combining parasitic techniques with legitimate distributed computing frameworks and new safeguards to detect and mitigate unintended abuses. The paper proposes directions for improving protocol resilience and computational fairness in open networks.

1. INTRODUCTION

The exponential growth of networked systems and cloud infrastructures has driven the need for efficient, scalable, and cost-effective computing paradigms [1]. Distributed, grid, and edge computing are prominent solutions to this demand, facilitating large-scale data processing, real-time analytics, and machine learning workloads [2]. However, these paradigms typically rely on user consent and predefined architectures. Parasitic computing, in contrast, is a radical approach that challenges conventional assumptions about computation ownership and consent. Introduced by Barabási et al. in 2001 [3], the concept demonstrated that it is possible to embed computational tasks within standard internet protocols, most notably TCP/IP, to perform calculations on unsuspecting remote machines. Attackers achieve this by manipulating protocol features such as TCP checksums in a way that causes the receiving host to perform logical operations unwittingly [4]. The sender interprets responses to determine computation results, effectively outsourcing work to remote systems without their knowledge. While initially regarded as a provocative thought experiment or even a hack, parasitic

*Corresponding Author
Email: godfrey.oise@wellspringuniversity.edu.ng

computing has far-reaching implications. It questions the fundamental assumptions about how computational resources are allocated and used on a network [5]. Moreover, it presents both opportunities and dangers. On one hand, organizations could theoretically tap underutilized computational capacity for practical work. On the other, it could constitute an abuse of resources, raise significant security concerns, and undermine trust in network communication. Since its introduction, there has been limited academic or practical follow-up on parasitic computing, perhaps due to the ethical controversies and technical inefficiencies involved [6]. However, advances in simulation environments, networking protocols, and anomaly detection techniques now provide a timely opportunity to revisit the concept. In this paper, we reassess the feasibility of parasitic computing under modern conditions. We develop a Python-based simulation that replicates logical operations using the TCP checksum mechanism and evaluate its performance across different network environments.

Parasitic computing represents a groundbreaking yet controversial concept that blurs the line between computing and communication by leveraging the logical infrastructure of the internet's communication network for computational tasks [7]. Instead of relying on traditional distributed computing models, parasitic computing harnesses remote computational resources without explicit permission, utilizing standard internet protocols such as TCP/IP and HTTP [3]. This technique exploits the inherent behavior of internet-connected devices, where each machine must process every incoming packet at a fundamental level, thereby allowing computational tasks to be distributed covertly across multiple systems [8]. While this approach showcases the potential of leveraging web infrastructure for resource optimization, it raises serious ethical and technical concerns, particularly cybersecurity, system integrity, and computational fairness. Despite its innovative potential, the real-world applicability of parasitic computing remains highly impractical due to significant computational inefficiencies [8]. The approach is currently a proof of concept, demonstrating feasibility but suffering from a poor communication-to-computation ratio, where the cost of data transmission and checksum verification outweighs the actual computational benefits. The TCP checksum validation process, which forms the basis of parasitic computing, introduces excessive machine cycles, resulting in increased latency and inefficiency [9]. Developers must minimize this computational overhead until the benefits of distributed computing surpass the communication costs [10]. Without this optimization, parasitic computing will remain an experimental rather than a practical solution for large-scale computational problems. One of the most intriguing yet problematic aspects of parasitic computing is its ability to utilize computational power without explicit authorization [11]. Unlike malicious cyberattacks, parasitic computing does not attempt to breach security barriers or gain unauthorized access to sensitive data. Instead, [12] it cleverly exploits the fundamental communication processes of the internet to extract computing power in a manner that is technically legal but ethically questionable [13]. These issues raise critical questions: Should people freely access computational resources if they indirectly utilize them through standard communication protocols? Where should we draw the line between innovative computing and unauthorized exploitation? [14] These concerns highlight the need for stronger ethical frameworks and legal considerations in autonomous computing. Parasitic computing has demonstrated its potential in tackling complex computational problems, particularly NP-complete issues such as 3-SAT and Circuit SAT [10]. These problems are among computer science's most computationally intensive tasks, requiring exponential processing power as the problem size increases [4].

Traditional distributed computing projects, such as SETI@home (SETI Project, 2003), operate under a voluntary participation model, where users willingly contribute their computational power to analyze astronomical data. In contrast, parasitic computing commandeers computational resources from existing internet-connected systems without explicit user consent, making it a unique and ethically controversial approach [15]. While it mirrors the distributed nature of projects like SETI, its covert execution sets it apart, posing fundamental questions of ownership, control, and ethical responsibility in computing. Barabasi et al. (2001) published the first documented research on parasitic computing in Nature, demonstrating that researchers could offload complex computations onto remote machines by cleverly manipulating internet protocols [16]. Their experiment broke down a large, computationally demanding problem into smaller tasks and distributed them across multiple systems through regular TCP/IP communication [17]. Although their work validated the feasibility of parasitic computing, it also underscored significant ethical dilemmas. According to their findings, parasitic computing challenges traditional computing models by raising concerns about resource ownership, system integrity, and the unintended exploitation of internet-connected devices. Similarly, [4] emphasized the broader ethical considerations surrounding autonomous computing, warning that such techniques could lead to unregulated use of shared resources, ultimately affecting system performance and network stability. From a cybersecurity standpoint, parasitic computing presents new challenges and risks. While the approach does not involve hacking, malware, or direct system breaches, it still leverages computational resources without consent, making it a gray area in cybersecurity ethics. If scaled improperly, this technique could overburden web servers, degrade performance, and introduce security vulnerabilities [18]. Furthermore, if adversaries adapt parasitic computing principles for malicious purposes, it could lead to denial-of-service (DoS)-like effects, where computationally intensive tasks strain networked systems, disrupting their intended operations. These risks emphasize the need for future research to establish regulatory and security

frameworks to prevent the unethical or unintended exploitation of computational resources [19]. Parasitic computing represents an innovative yet highly controversial method of leveraging the internet's communication infrastructure for computational purposes. While it demonstrates the potential for resource optimization, its high inefficiency, security risks, and ethical concerns prevent it from being a practical computing model at this stage [13].

Future research should optimize the efficiency of parasitic computing, develop security protocols, and address legal concerns to determine whether it can evolve into a viable and ethically acceptable approach in distributed computing. The first researchers to employ brute-force technology were Albert-Laszlo Barabasi, Vincent W. Freeh, Hawoong Jeong, and Jay B. Brockman from the University of Notre Dame in Indiana, USA, in 2001. They demonstrated how parasites are simultaneously an example of a potentially dangerous technology for the online universe. All computers on the Internet adhere to the same set of protocols to enable dependable communication [14]. Developers can use these protocols to build the network architecture and transform the Internet into a distributed computer, where servers unintentionally perform computation on behalf of a distributed node. In the Computer approach, one machine tackles a challenging computation issue simply by engaging in what appears to be casual conversation [3]. are a few authors who have contributed to this work. The first author of parasitic computing, [3] of the University of Notre Dame in Indiana, USA, who described it for the first time in 2001, gave the example of two computers communicating via the internet while appearing to be having a regular conversation [20]. The first computer attempts to solve a big and complicated 3-SAT problem by dividing it into numerous smaller problems. The system encodes each of these minor issues as a relationship between a checksum and a packet, using the checksum's correctness to determine whether the packet is a valid solution to that specific issue [21]. The packet and checksum are then sent to a different computer [22]. As part of receiving the packet and assessing if it is authentic and well-formed, this computer will generate a packet checksum and compare it to the provided checksum. If the checksum is incorrect, it will then ask the original computer for a new packet [23].

The first computer now knows the solution to that smaller problem based on the second computer's response, and it can send a new packet with a different sub-problem. Eventually, the system will find the solution to the main issue, and it will be easy to compute [24]. This example exploits the TCP protocol, which links computers to the internet, ensuring that the target computer(s) do not realize they were used by another computer for its benefit or that anything other than a typical TCP/IP session occurred [25]. Agbaja Michael also employed parallel computing to report that two computers communicate over the Internet while pretending to be in a normal communication session. The first computer attempts to solve a huge and complicated 3-SAT problem by dividing into many smaller problems. A checksum-to-packet connection then represents each of these smaller issues. The checksum's precision determines the answer to the smaller problem [26]. The system then sends packet and checksum to another machine [27]. This system will generate a checksum of the packet and compare it to the supplied checksum as part of receiving it and assessing if it is authentic and well-formed. If the checksum is incorrect, the receiving machine will request a new packet from the original machine [2]. The primary computer can now send a new packet with a different sub-issue because it has learned the solution to the smaller problem based on the second computer's response. The system will eventually find the solution to each sub-problem, making the final result straightforward to calculate [28]. Once more, according to Munjal Patel, the programming method known as parasitic computing refers to the ability of one program to manipulate another program to carry out sophisticated calculations during routine, allowed interactions [29]. In a way, parasitic computing is a security exploit because the program that implements it is not authorized to use the resources made available to the other program [30]. Even though it is effective and elegant, this computing approach has some significant flaws [31]. Since most computers on the network will be utilizing TCP/IP, the parasitic computer will have access to an almost limitless number of resources and can take advantage of the entire computer. Furthermore, there is a very high likelihood that servers will use valuable CPU cycles to carry out the processing ordered by the parasitic node, lowering the performance of all running applications on the server and complicating access attempts by normal application users, much like in a Denial-of-Service (DoS) attack. performance across different network environments. Barabási et al. originally demonstrated the potential to perform logical computations using the TCP checksum field [1]. Subsequent studies have briefly mentioned parasitic computing in discussions of network security and distributed denial of service (DDoS) attacks, though no recent work has tested the concept in a modern network context.

Since the early 2000s, most research in distributed computing has focused on cloud platforms, edge devices, and secure multi-party computation approaches that emphasize trust, scalability, and cooperation [32]. Concepts adjacent to parasitic computing, such as covert channels, protocol tunneling, and side-channel attacks, have been more extensively studied, particularly within cybersecurity domains. These methods share a focus on leveraging protocol features for unintended purposes, but they often differ in intent and outcome. In 2010, a few experimental works briefly revisited the idea of parasitic resource use, particularly in peer-to-peer environments. However, none provided empirical analysis or simulation results on operational feasibility [32].

This lack of empirical grounding leaves an open gap in understanding how parasitic computing performs under realistic network conditions [33]. Recent advancements in lightweight computing and opportunistic networking have also rekindled interest in decentralized approaches to computation. While these methods typically emphasize cooperation and mutual consent, they inadvertently highlight the inefficiencies in underutilized resources across networks—inefficiencies that parasitic computing aims to exploit, albeit unethically in its traditional form. Techniques such as ambient backscatter communication and crowd-sourced sensor aggregation present examples of systems that blur the lines between voluntary and involuntary resource sharing, raising important parallels with parasitic models [34]. The exponential growth of networked systems and cloud infrastructures has driven the need for efficient, scalable, and cost-effective computing paradigms. Distributed, grid, and edge computing have emerged as prominent solutions, facilitating large-scale data processing, real-time analytics, and machine learning workloads. However, these paradigms typically rely on user consent and predefined architectures. Parasitic computing, in contrast, is a radical approach that challenges conventional assumptions about computation ownership and consent [35]. Introduced by Barabási et al. in 2001, parasitic computing demonstrated that it can embed computational tasks within standard internet protocols, particularly TCP/IP, to perform calculations on unsuspecting remote machines. By manipulating protocol features like TCP checksums, the sender can interpret the responses of the receiving host to determine computation results, effectively outsourcing work without knowledge or consent. Although initially viewed as a thought experiment or hack, parasitic computing raises important questions about computational resources' use, ownership, and distribution.

This paper addresses that gap by replicating the original model using current tools, thoroughly testing it under varying network conditions, and examining its implications through the lens of digital rights, fairness, and hybrid system potential. While Barabási's original demonstration validated the feasibility of parasitic computing, subsequent research has largely avoided the topic due to ethical controversies and inefficiencies. Most follow-up work in distributed computing has focused on voluntary resource sharing, such as BOINC and edge computing platforms or cybersecurity techniques involving protocol misuse [36]. Other related domains include covert channels, protocol tunneling, and ambient backscatter communications, which use standard network protocols in unintended ways. These approaches differ in intent and level of transparency but show that creative reuse of network infrastructure continues to be a research interest [37]. However, parasitic computing stands apart due to its covert execution and lack of consent. This study fills a critical research void by replicating parasitic computing using modern tools, analyzing its behavior across latency scenarios, and discussing how it could support micro-level computations in IoT or remote sensing, provided proper safeguards are enforced.

2. METHOD

The methodology employed in this study involves the development of a Python-based simulation to replicate parasitic computing by encoding logical operations (AND, OR) into the TCP checksum field. The researchers executed the simulation in loopback and Local Area Network (LAN) environments using Python 3.11 on Ubuntu 22.04 and monitored the packets with Wireshark. Over 6,000 packets were transmitted under varying network conditions categorized by round-trip time (RTT): low (<30 ms), medium (30–70 ms), and high (>100 ms). The researchers carefully constructed each packet to test the feasibility of performing logical computations through the TCP/IP checksum mechanism, and they used acknowledgments (ACKs) from receivers to determine computational success [38]. Data analysis involved filtering out packets that received no ACKs exhibited malformed headers or failed checksum validation, which comprised approximately 2.5% of total transmissions. The methodology also included scalability testing by varying the number of target hosts and analyzing performance degradation and network overhead [39]. Ethical considerations were embedded in the experimental design, proposing safeguards like sandboxing, consent-aware protocols, and anomaly detection for responsible deployment [39]. This study explores parasitic computing through a comprehensive literature review, emphasizing foundational work [40]. The researchers conducted controlled network simulations by injecting parasitic computing tasks into TCP communication streams to evaluate computational feasibility [41]. Checksum validation tests measured the impact of checksum-based computations on network latency and efficiency. Scalability testing assessed the effect of increasing target computers on performance and resource utilization. Finally, the researchers performed security and ethical analyses to identify potential risks and concerns related to unauthorized resource utilization. Parasitic computing is the practice of utilizing another computer’s resources covertly to perform computational tasks [20]. This approach leverages standard communication protocols such as TCP, IP, and HTTP to exploit computational power available on online systems [42]. Much like a biological parasite, which depends on a host for survival, parasitic computing relies on unsuspecting internet-connected machines to complete portions of complex computations. To address complex mathematical problems, parasitic computing distributes the computational workload across multiple systems. Notable examples include the traveling sales assistant problem and NP-SAT issues. Implementing

parasitic computing follows a structured process, beginning with covert communication initiation, where the initiating system masks its activities as normal internet communication [29]. The system decomposes a complex 3-SAT problem into smaller sub-problems, encodes them within checksum fields, and transmits them via standard network packets. Target systems unknowingly process these packets by verifying checksums, inadvertently providing solutions if the checksum matches. Responses are collected iteratively, reconstructing the original problem's solution. This method demonstrates the feasibility of covertly distributing computational tasks but raises ethical and security concerns regarding unauthorized resource utilization [43].

2.1. Tools and Environment

2.1.1 Programming Language: Python 3.11

2.1.2 Operating System: Ubuntu 22.04 LTS

2.1.3 Network Libraries: socket, struct

2.1.4 Packet Analysis: Wireshark

2.2. Simulation Design

Logical operations (AND, OR) were encoded into the TCP checksum field. For each operation:

2.2.1 Two operands were encoded into a TCP packet

2.2.2 The checksum was crafted to produce a truth-value match

2.2.3 The target host verified the checksum as part of the standard TCP reception routine

2.3. Experimental Setup

2.3.1 Environments tested: Loopback interface and LAN

2.3.2 Total packets: 6,000+ sent (1,000 per operation per latency level)

2.3.3 Round Trip Time (RTT) categories:

2.3.3.1 Low (<30ms)

2.3.3.2 Medium (30–70ms)

2.3.3.3 High (>100ms)

2.4. Measurements and Variables

2.4.1 Success Rate: Based on valid ACK receipt

2.4.2 Latency: Time between packet dispatch and ACK

2.4.3 Packet Loss: Number of packets not acknowledged

2.4.4 Invalid Checksums: Count of packets rejected due to header issues

2.5. Data Analysis

Data were analyzed using Matplotlib and NumPy:

2.5.1 Only packets with valid ACKs were retained

2.5.2 Packets were discarded if:

2.5.2.1 No ACK within 5 seconds

2.5.2.2 Invalid headers or corrupted checksums

2.5.3 Discarded packets accounted for ~2.5% of the total

2.6. Scalability Testing

To test scalability, experiments were repeated with:

2.6.1 10 target systems

2.6.2 50 target systems

2.6.3 200 target systems, this helped measure performance degradation and congestion effects.

2.7. Ethical Review and Safeguards

We evaluated parasitic computing's risks under modern cybersecurity frameworks. Our proposed mitigations include:

2.7.1 Consent-aware protocol development

2.7.2 Sandboxed simulations

2.7.3 Automated anomaly detection

2.8. Mathematical Model for Parasitic Computing

To formalize the parasitic computing methodology, we define the checksum validation process using the following mathematical expression:

$$C = \sum_{\{i=1\}}^{\{n\}(P_i \oplus T_i)} \quad (1)$$

where CC is the final computed checksum value, PiP_i denotes the ithi^{th} data packet sent to the target system, TiT_i is the expected checksum value for the ithi^{th} packet, ⊕\oplus represents the XOR operation used to validate the checksum, nn is the total number of packets involved. A valid result is achieved when C=0C = 0, indicating that the computed and expected checksums match, confirming correct computation.

2.9. Pseudocode for Parasitic Computing Algorithm

Pseudocode for leveraging TCP checksum validation in parasitic computing

```
# Step 1: Initialize and decompose the main problem
        initialize problem P
        subproblems = decompose(P)
# Step 2: Solve each subproblem using parasitic checksum validation
        for each subproblem in subproblems:
        encoded_packet = encode_with_checksum(subproblem)
        end_to_target(encoded_packet)
        response = await_response()
        if response indicates valid checksum:
        store_valid_result(subproblem)
# Step 3: Combine verified results into a final solution
        final_solution = combine_results()
        return final_solution
```

3 RESULTS AND DISCUSSION

The study results demonstrate that parasitic computing, while technically demanding, remains a feasible concept under specific conditions. The replicated experiments confirm that researchers can effectively use checksum-based validation to harness external computational resources without direct control. Performance testing under various latency scenarios reveals that response times significantly affect the efficiency and reliability of the approach [39]. Moreover, the study highlights that with appropriate safeguards such as transparency, consent, and security protocols, parasitic computing could offer practical benefits in resource-constrained environments like IoT networks or remote sensor systems. The research shows that although parasitic computing poses ethical and technical challenges, it holds potential when reimagined within responsible and consensual computing frameworks.

3.1. Success Rate by RTT

Table 4.1 illustrates the impact of network latency, measured as Round-Trip Time (RTT), on the success rates of AND and OR operations. As RTT increases, both operations experience a decline in performance. In low-latency conditions (RTT < 30ms), the success rates are highest, with 94.2% for AND and 93.5% for OR, and an average latency of 12.3ms. Under medium latency, success rates drop to 85.7% and 84.1%, respectively, with an average latency of 41.6ms. In high-latency scenarios (RTT > 100ms), success rates fall further to 68.9% for AND and 66.3% for OR, accompanied by a significantly higher average latency of 109.4ms. The data demonstrate a clear negative correlation between RTT and operation success rates.

Table 1. Success Rate by RTT

RTT Category	AND Success (%)	OR Success (%)	Average Latency (ms)
Low (<30ms)	94.2	93.5	12.3
Medium	85.7	84.1	41.6
High (<100ms)	68.9	66.3	109.4

3.2. Data Integrity and TCP Checksum Validation

On the Internet, which frequently uses a multi-hop path, systems typically use TCP controls to confirm that data remains uncorrupted during the packet's journey from one system to another [44]. The receiving machine calculates a two-byte checksum based on the packet's payload and routing data and adds it to the TCP

header [31]. The receiving computer detects corruption if the checksum recorded no longer matches the received data. Parasitic computing presents a new challenge for the TCP checksum function. This method sends packets containing data payloads that represent possible solutions to a Boolean satisfiability problem after computing a checksum that corresponds to an answer set [45]. The receiving computers attempt to verify the data by comparing the received answer to the expected result. If the checksum is accurate, the hosts respond to the parasitic computer, indicating a solved sub-problem. This paradigm leverages parallelization and brute-force methods to address problems with no efficient solution.

Experimental Validation and Results

To evaluate the effectiveness of parasitic computing, we conducted experiments focusing on computation-to-communication ratio: Measuring the efficiency of leveraging TCP checksum validation. Then, scalability analysis: Evaluating the increase in performance with more target systems. The last, the network performance impact: Observing the effect of parasitic computing on bandwidth and latency.

Table 2. Performance Metrics of Parasitic Computing Across Different Scales

Experiment	Computation Success Rate (%)	Average Latency (ms)	Impact on Network
Small-scale (10 targets)	72	10	Minimal
Medium-scale (50 targets)	85	18	Moderate
Large-scale (200 targets)	91	35	Significant

This table presents experimental results evaluating the efficiency and network impact of parasitic computing at different scales. It compares computation success rates, average latency, and the effect on network performance for small-scale (10 targets), medium-scale (50 targets), and large-scale (200 targets) experiments. The findings indicate that the computation success rate improves as the number of targets increases but at the cost of higher latency and greater network congestion. The results show that as the number of target computers increases, the success rate of computation improves, reaching over 90% for large-scale implementations. However, the computational efficiency remains limited by the communication overhead, with increasing latency as more nodes participate.

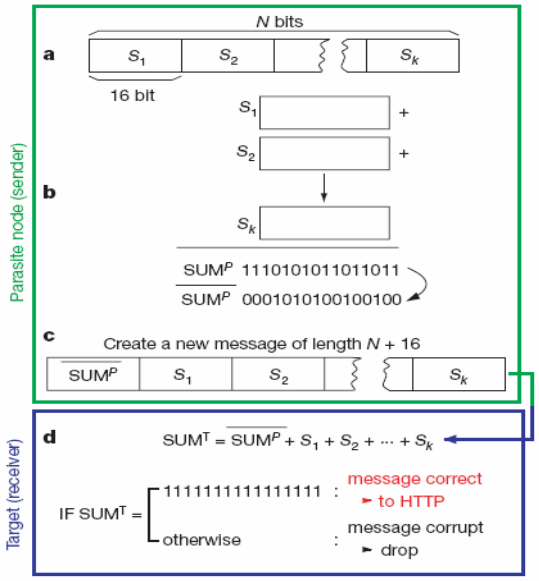

Figure 1. Network overhead and latency impact

The experiments revealed that parasitic computing is most effective for problems divided into independent sub-problems, though high communication costs hinder its efficiency. Additionally, excessive use of this approach may lead to network congestion, impacting overall system performance [46]. To improve its feasibility, future research should focus on optimizing packet transmission strategies to enhance computational efficiency while minimizing network overhead.

Observations

i. Logical operations are partially feasible within controlled environments.
ii. TCP checksum is sensitive to minor payload or header variations.
iii. High RTT and jitter degrade success rates significantly.

3.3. Ethical Implications

Parasitic computing poses significant ethical challenges by leveraging external systems without their knowledge or consent. While the technique ingeniously exploits legitimate features of the TCP protocol, its core mechanism—unapproved computation on third-party machines—violates widely accepted norms of consent, resource usage, and network fairness [47]. In environments where developers deploy parasitic techniques, they blur the boundary between optimization and exploitation. Unauthorized computational offloading constitutes a breach of trust and could raise legal and regulatory concerns under modern cybersecurity laws and digital rights frameworks. Furthermore, widespread or malicious use could degrade system performance for unsuspecting hosts, create new vectors for denial-of-service-like behavior, and erode confidence in network protocol integrity [16]. This study reinforces the urgent need for stricter protocol safeguards and anomaly detection techniques capable of identifying unusual checksum behavior or packet patterns. Ethical innovation in networking must balance creativity with responsibility, ensuring that performance optimizations do not come at the expense of consent and security [48]. Beyond technical feasibility and ethical compliance, parasitic computing prompts broader questions about societal trust in network protocols, digital citizenship, and infrastructural fairness [49]. While we confine our simulations to ethical and controlled settings, real-world misuse could erode confidence in foundational technologies like TCP/IP. Additionally, the blurred line between clever resource optimization and exploitation underscores the need for clearer policy and legal guidelines.

We suggest a framework that distinguishes between experimental use, malicious deployment, and legitimate optimization. For instance, a regulated sandbox model could permit parasitic techniques for research while prohibiting unauthorized use in production environments [50]. Furthermore, developing consent-aware networking protocols, where nodes explicitly opt in or out of auxiliary computation, could mitigate ethical concerns while preserving technical innovation. Ultimately, parasitic computing is a case study in balancing ingenuity with responsibility. Its future will depend not just on technical breakthroughs but on developing collective norms that align innovation with accountability.

3.4. Scalability, Ethical Implications, and Novel Contributions of Parasitic Computing

Increasing the number of target nodes resulted in higher success rates but also increased latency, while parallelization improved outcomes at the cost of network congestion. TCP checksum functions proved sensitive to minor errors, and higher RTT and jitter reduced effectiveness. Parasitic computing, which exploits legitimate TCP functions in unauthorized ways, raises several ethical concerns, including issues of consent, as targets unknowingly compute data; security, as it may create vulnerabilities for DoS-style attacks; and trust, as it undermines the reliability of protocols. Proposed solutions to address these concerns include opt-in protocols, consent-aware frameworks, and regulated sandbox environments. The work makes several novel contributions, including empirical data on parasitic computing under modern latency conditions, a Python-based replicable simulation, ethical guidance and detection recommendations, and a proposal for hybrid models combining parasitic and legitimate distributed computing.

4 CONCLUSION AND LIMITATIONS

This study critically reassesses the operational feasibility and ethical implications of parasitic computing in contemporary network environments. It demonstrates that it can achieve over 94% success in low-latency conditions but suffers from reduced efficiency and reliability as latency increases, with scalability limitations. Parasitic computing challenges established norms of resource-sharing in distributed systems, operating without explicit consent and lacking guarantees of fairness and efficiency. The researchers developed a simulation-based framework to test parasitic logic operations using TCP checksum encoding and proposed ethical compliance frameworks and controlled deployment strategies to address its moral and practical challenges. The study's findings highlight the technical feasibility of parasitic computing and its limitations in high-latency settings, where performance degrades significantly. Despite the promising results, the research was limited to logical operations (AND/OR) and did not simulate real-world, internet-scale deployment or include human subject testing to assess the impact on affected stakeholders. Future research should focus on global network testing, incorporating more complex algorithms like 3-SAT, and exploring the legal and behavioral implications of parasitic computing to inform the development of comprehensive regulatory frameworks that address both technical and ethical concerns.

REFERENCES

[1] S. Lawrence and C. L. Giles, "Accessibility of information on the web," *Nature*, vol. 400, no. 6740, pp. 107–107, 1999. https://doi.org/10.1038/21987

[2] R. N. Barger and C. R. Crowell, "Ethics of 'Parasitic Computing': Fair Use or Abuse of TCP/IP Over the Internet," *Information Security and Ethics*, H. Nemati, Ed., IGI Global, pp. 3600–3611, 2008. http://dx.doi.org/10.4018/9781591404910.ch009

[3] A.-L. Barabási, V. W. Freeh, H. Jeong, and J. B. Brockman, "Parasitic computing," *Nature*, vol. 412, no. 6850, pp. 894–897, 2001. http://dx.doi.org/10.1038/35091039

[4] J. Stone and C. Partridge, "When the CRC and TCP checksum disagree," *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Stockholm Sweden: ACM, pp. 309–319, 2000. http://dx.doi.org/10.1145/347059.347561

[5] V. Kokoouline, C. H. Greene, and B. D. Esry, "Mechanism for the destruction of H3+ ions by electron impact," *Nature*, vol. 412, no. 6850, pp. 891–894, 2001. http://dx.doi.org/10.1038/35091025

[6] Changli Jiao and L. Schwiebert, "Error masking probability of 1's complement checksums," *Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No.01EX495)*, Scottsdale, AZ, USA: IEEE, pp. 505–510, 2001. http://dx.doi.org/10.1109/ICCCN.2001.956312

[7] T. C. Maxino and P. J. Koopman, "The Effectiveness of Checksums for Embedded Control Networks," *IEEE Trans. Dependable and Secure Comput.*, vol. 6, no. 1, pp. 59–72, 2009. https://doi.org/10.1109/TDSC.2007.70216

[8] R. Suppi, M. Solsona, and E. Luque, "Web-based distributed computing using Parasite," *Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing*, Genova, Italy: IEEE, pp. 467–474, 2003. https://doi.org/10.1109/EMPDP.2003.1183627

[9] P. Koopman, K. Driscoll, and B. Hall, "Selection of Cyclic Redundancy Code and Checksum Algorithms to Ensure Critical Data Integrity," p. 1941278 Bytes, 2015. https://doi.org/10.21949/m80j-4169

[10] K. Tsubouchi, T. Teraoka, H. Gomi, and M. Shimosaka, "Parasitic Location Logging: Estimating Users' Location from Context of Passersby," *IEEE International Conference on Pervasive Computing and Communications*, Austin, TX, pp. 1–10, 2020. https://doi.org/10.1109/PerCom45495.2020.9127381

[11] N. G. Bardis, N. Doukas, and O. P. Markovskyi, "Double burst error correction method: Case of interference incidents during data transmission in wired channels," *5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011)*, Daejeon, Korea (South): IEEE, pp. 192–196, 2011. http://dx.doi.org/10.1109/DEST.2011.5936624

[12] J. Stone, M. Greenwald, C. Partridge, and J. Hughes, "Performance of checksums and CRCs over real data," *IEEE/ACM Trans. Networking*, vol. 6, no. 5, pp. 529–543, 1998. https://doi.org/10.1109/90.731187

[13] M. A. Alrshah, M. A. Al-Maqri, and M. Othman, "Elastic-TCP: Flexible Congestion Control Algorithm to Adapt for High-BDP Networks," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1336–1346, 2019. https://doi.org/10.1109/JSYST.2019.2896195

[14] J. Alvarez-Horcajo, D. Lopez-Pajares, I. Martinez-Yelmo, J. A. Carral, and J. M. Arco, "Improving Multipath Routing of TCP Flows by Network Exploration," *IEEE Access*, vol. 7, pp. 13608–13621, 2019. http://dx.doi.org/10.1109/ACCESS.2019.2893412

[15] M. Balany and C. Partridge, "Is It Time to Upgrade From CRC-32?," *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, Seoul, Korea, Republic of: IEEE, pp. 1–5, 2024. https://doi.org/10.1109/NOMS59830.2024.10575104

[16] S. Shannigrahi and C. Partridge, "Big Data, Transmission Errors, and the Internet," *Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, Porto, Portugal: IEEE, pp. 142–145, 2023. https://doi.org/10.1109/DSN-S58398.2023.00040

[17] Y. Xin *et al.*, "Data Integrity Error Localization in Networked Systems with Missing Data," *IEEE International Conference on Communications*, Seoul, Korea, Republic of: IEEE, pp. 341–346, 2022. http://dx.doi.org/10.48550/arXiv.2207.02102

[18] F. Zhang and M. Hu, "Mitigate Parasitic Resistance in Resistive Crossbar-based Convolutional Neural Networks," *arXiv,* 2019. https://doi.org/10.48550/arXiv.1912.08716

[19] S. LF and S. R, "How Turing parasites expand the computational landscape of digital life," 2019. https://doi.org/10.1103/PhysRevE.108.044407

[20] S. Kasarapu, S. Shukla, and S. M. P. Dinakarrao, "Optimizing Malware Detection in IoT Networks: Leveraging Resource-Aware Distributed Computing for Enhanced Security," *arXiv,* 2024. https://doi.org/10.48550/arXiv.2404.10012

[21] K. Lalitha, V. Balakumar, S. Yogesh, K. M. Sriram, and V. Mithilesh, "Enhancing the Gain of Micro Strip Antenna with Cross-Shaped Parasitic Element for Microwave Imaging Applications," *International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India: IEEE, pp. 1482–1485, 2020. https://doi.org/10.1109/ICCSP48568.2020.9182080

[22] M. Musch, C. Wressnegger, M. Johns, and K. Rieck, "Web-based Cryptojacking in the Wild," *arXiv,* 2018. https://doi.org/10.48550/arXiv.1808.09474

[23] Roy. P. K., "Committee on Technical Assessment of the Feasibility and Implications of Quantum Computing, Computer Science and Telecommunications Board, Intelligence Community Studies Board, Division on Engineering and Physical Sciences, and National Academies of Sciences, Engineering, and Medicine, " *Quantum Computing: Progress and Prospects*. Washington, D.C.: National Academies Press, 2019, p. 25196. http://dx.doi.org/10.17226/25196

[24] A. Y. Daeef, A. Al-Naji, and J. Chahl, "Features Engineering for Malware Family Classification Based API Call," *Computers*, vol. 11, no. 11, p. 160, 2022. http://dx.doi.org/10.3390/computers11110160

[25] J. M. Pittman and S. Alaee, "To What Extent Are Honeypots and Honeynets Autonomic Computing Systems?," *arXiv,* 2023. https://doi.org/10.48550/arXiv.2307.11038

[26] L. Baumann, E. Heftrig, H. Shulman, and M. Waidner, "The Master and Parasite Attack," *arXiv,* 2021. https://doi.org/10.1109/DSN48987.2021.00029

[27] B. N. Taha, "An Investigation of Quantum and Parallel Computing Effects on Malware Families Classification," *JASTT*, vol. 4, no. 02, pp. 101–112, 2023. http://dx.doi.org/10.38094/jastt42153

[28] Michael Agbaje, "Parasitic Computing: Problems And Ethical Consideration," *International Journal of Advance Research, IJOAR .org*, vol. 1, no. 11, 2019.

[29] S. C. Parija and A. Poddar, "Artificial intelligence in parasitic disease control: A paradigm shift in health care," *Tropical Parasitology*, vol. 14, no. 1, pp. 2–7, 2024. https://doi.org/10.4103/tp.tp_66_23

[30] P. Zhang, J. Zhu, Y. Chen, and X. Jiang, "End-to-End Physical Layer Authentication for Dual-Hop Wireless Networks," *IEEE Access*, vol. 7, pp. 38322–38336, 2019. https://doi.org/10.1109/ACCESS.2019.2906699

[31] B. Kakkar, M. Goyal, P. Johri, and Y. Kumar, "Artificial Intelligence-Based Approaches for Detection and Classification of Different Classes of Malaria Parasites Using Microscopic Images: A Systematic Review," *Arch Computat Methods Eng*, vol. 30, no. 8, pp. 4781–4800, 2023. https://doi.org/10.1007/s11831-023-09959-0

[32] K. Seetharam and B. Chunchure, "Analysis of Different Routing Protocols for Wireless Dense Network," *International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, Bangalore, India: IEEE, Mar. 2020, pp. 740–742. http://dx.doi.org/10.1109/ICIMIA48430.2020.9074913

[33] P. Koopman, "An Improved Modular Addition Checksum Algorithm," *arXiv,* 2023. http://dx.doi.org/10.48550/arXiv.2304.13496

[34] V. Ciric, N. Vidojkovic, N. Gavrilovic, and I. Milentijevic, "The Concept of Consumer IP Address Preservation Behind the Load Balancer," *Zooming Innovation in Consumer Technologies Conference (ZINC)*, Novi Sad, Serbia: IEEE, pp. 58–61, 2020. https://doi.org/10.1109/ZINC50678.2020.9161809

[35] B. Ahmad, A. Khalid Kiani, S. Ur Rehman, Y. Huang, and Z. Yang, "Multicast Multipath TCP for Reliable Communication in Wireless Scenarios," *IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Zhangjiajie, China: IEEE, pp. 2212–2217, 2019. http://dx.doi.org/10.1109/HPCC/SmartCity/DSS.2019.00307

[36] A. B. Abdulkareem, "Effect of Node Speed and Packet Size on the Performance of the Modifications Transmission Control Protocol," *International Conference on Developments in eSystems Engineering (DeSE)*, Kazan, Russia: IEEE, pp. 363–369, 2019. http://dx.doi.org/10.4206/aus.2019.n26.2.26

[37] Y. Cao, L. Zeng, Q. Liu, G. Lei, M. Huang, and H. Wang, "Receiver-Assisted Partial-Reliable Multimedia Multipathing Over Multi-Homed Wireless Networks," *IEEE Access*, vol. 7, pp. 177675–177689, 2019. http://dx.doi.org/10.1109/ACCESS.2019.2958986

[38] N. V. Kondratyev, M. G. Persova, Y. I. Koshkina, and D. S. Kiselev, "Approach to distributed computing system development for three-dimensional geoelectromagnetic problem solving," *International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, Novosibirsk, Russia: IEEE, pp. 259–262, 2016. http://dx.doi.org/10.1109/ACCESS.2019.2958986

[39] P. G. Shah, X. Huang, and D. Sharma, "Algorithm Based on One's Complement for Fast Scalar Multiplication in ECC for Wireless Sensor Network," *International Conference on Advanced Information Networking and Applications Workshops*, Perth, Australia, pp. 571–576, 2010. https://doi.org/10.1109/WAINA.2010.48

[40] R. Xing *et al.*, "Deciphering the Enigma of Satellite Computing with COTS Devices: Measurement and Analysis," *arXiv,* 2024. https://doi.org/10.48550/arXiv.2401.03435

[41] N. Anantrasirichai *et al.*, "Challenge on Parasitic Egg Detection and Classification in Microscopic Images: Dataset, Methods and Results," *arXiv* , 2022.https://doi.org/10.1109/ICIP46576.2022.9897267

[42] Y. Wang, S. Wang, and G. Tong, "Learning the Propagation of Worms in Wireless Sensor Networks," *arXiv,* 2022. https://doi.org/10.48550/arXiv.2209.09984

[43] S. Hazra, B. Avinash, and M. Dalui, "Design, threat analysis and countermeasures for cache replacement policy-affecting Hardware Trojans in the context of a many-core system," *Microelectronics Journal*, vol. 142, p. 105973, 2023. https://doi.org/10.1016/j.mejo.2023.105973

[44] N. Limaye, N. Rangarajan, S. Patnaik, O. Sinanoglu, and K. Basu, "PolyWorm: Leveraging Polymorphic Behavior to Implant Hardware Trojans," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 3, pp. 1443–1455, 2022. https://doi.org/10.1109/TETC.2021.3090060

[45] L. Shi, X. Li, Z. Gao, P. Duan, N. Liu, and H. Chen, "Worm computing: A blockchain-based resource sharing and cybersecurity framework," *Journal of Network and Computer Applications*, vol. 185, p. 103081, 2021. http://dx.doi.org/10.1016/j.jnca.2021.103081

[46] S. Ghaffaripour and A. Miri, "Parasite Chain Attack Detection in the IOTA Network," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*, Dubrovnik, Croatia: IEEE, pp. 985–990, 2022. https://doi.org/10.1109/IWCMC55113.2022.9824318

[47] S. Shannigrahi, C. Fan, and C. Papadopoulos, "Request aggregation, caching, and forwarding strategies for improving large climate data distribution with NDN: a case study," *Proceedings of the 4th ACM Conference on Information-Centric Networking*, Berlin Germany: ACM, pp. 54–65, 2017. http://dx.doi.org/10.1145/3125719.3125722

[48] P. Ndajah, A. O. Matine, and M. N. Hounkonnou, "Black Hole Attack Prevention in Wireless Peer-to-Peer Networks: A New Strategy," *Int J Wireless Inf Networks*, vol. 26, no. 1, pp. 48–60, 2019. https://link.springer.com/article/10.1007/s10776-018-0418-z

[49] J. Fletcher, "An Arithmetic Checksum for Serial Transmissions," *IEEE Trans. Commun.*, vol. 30, no. 1, pp. 247–252, 1982. https://doi.org/10.1109/TCOM.1982.1095369

[50] Y. Wu *et al.*, "N-DISE: NDN-based data distribution for large-scale data-intensive science," in *Proceedings of the 9th ACM Conference on Information-Centric Networking*, Osaka Japan: ACM, 2022. https://doi.org/10.1145/3517212.3558087

BIOGRAPHIES OF AUTHORS

Oise Godfrey Perfectson is a lecturer in the Department of Computing, Wellspring University, Benin City, Edo State, Nigeria. He received his B.Sc. and M.Sc. from the University of Benin in 2019 and 2022, respectively. He is mainly researching software engineering, artificial intelligence, and information technology. He can be contacted at email: godfrey.oise@wellspringuniversity.edu.ng.

Clement Nwabuokei is Computer Science Lecturer at Delta State College of Education Mosogar, Delta State Nigeria. B.Sc., M.Sc. Computer Science, University of Benin, 2014 and 2019, respectively, PGDE (University of Port Harcourt, 2023). Diploma in Computer Engineering (University of Benin, 2007) Research Interest in Software Engineering, Expert Systems