# EdgeTrust-Offload: A Zero-Trust Framework for Secure, Dynamic Task Offloading in Congested Edge Clusters

Wanghley Soares Martins
ECE 654: Edge Computing

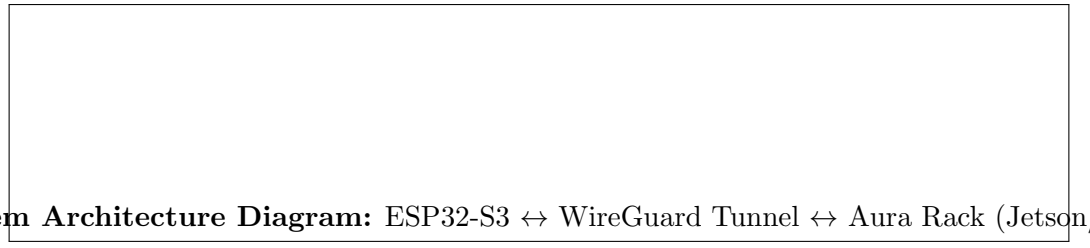
February 3, 2026

## THE CORE IDEA

Edge computing promises sub-millisecond latency by processing data near its source, yet this proximity creates a critical vulnerability: traditional offloading architectures assume implicit trust within the local network. This assumption is increasingly untenable as edge deployments handle sensitive data—from biometric signals to industrial telemetry—that demand cryptographic protection regardless of network topology.

This project proposes **EdgeTrust-Offload**, a security-aware scheduler for IoT-to-Edge task offloading that quantifies the performance trade-off between Zero-Trust Architecture (ZTA) overhead and computational offloading benefits. The system comprises three components:

1. **Security Layer:** A Zero-Trust network overlay using **Tailscale/WireGuard** to establish mutual TLS (mTLS) and identity-based access control between all nodes, eliminating the need for complex service mesh deployments.
2. **Workload Engine:** Signal processing tasks (1024-point FFT, FIR filtering) executing on an **ESP32-S3** sensor node with optional offloading to an **NVIDIA Jetson** accelerator within the Aura home-server rack.
3. **Decision Engine:** A Python-based scheduler implementing a cost function $C = \alpha \cdot T_{exec} + \beta \cdot T_{network} + \gamma \cdot E_{crypto}$ that dynamically selects between local processing and secure offloading based on real-time network jitter, encryption overhead, and task complexity.



[**System Architecture Diagram:** ESP32-S3 ↔ WireGuard Tunnel ↔ Aura Rack (Jetson/RPi)]

The core innovation lies in identifying the **security-performance crossover point**—the network latency threshold at which ZTA overhead renders offloading slower than local execution on resource-constrained MCUs.

## THE FOUR WHYS

**Why this?** Traditional computation offloading research assumes a trusted LAN environment, ignoring the "security tax" of encryption and authentication. This project benchmarks ZTA overhead in edge environments where raw sensor data (e.g., health metrics, industrial signals) requires protection under regulations like HIPAA and GDPR. By quantifying when security overhead exceeds offloading benefits, we establish design guidelines for secure edge systems [1], [2].

**Why now?** NIST SP 800-207 has established Zero-Trust as the federal standard for network security, yet edge computing deployments lag in adoption due to perceived performance penalties. Simultaneously, hardware-accelerated cryptography on chips like the ESP32-S3 (with dedicated AES and SHA accelerators) has reached maturity. We must determine whether modern crypto hardware has closed the performance gap sufficiently for ZTA adoption at the edge [3], [4].

**Why me?** My background combines computer engineering at Duke with hands-on experience in custom PCB design for IoT platforms (ESP32, Orange Pi) and research in non-contact capacitive sensing requiring real-time signal processing. I have deployed the Aura home-server rack infrastructure that serves as the testbed for this project, providing direct access to heterogeneous edge hardware.

**Why you?** ECE 654's emphasis on low-latency architectures and heterogeneous edge clusters provides the ideal framework for this systems-level performance analysis. The findings directly address the course's central question: how do we architect edge systems that balance competing constraints of latency, security, and resource efficiency?

## RELATED WORK

The intersection of Zero-Trust security and edge computing offloading represents an emerging research frontier spanning three domains:

**Zero-Trust Architecture Foundations.** NIST SP 800-207 [1] establishes the canonical ZTA model requiring continuous verification of all network actors. Rose et al. [5] extend this to distributed systems, while Kindervag's original work [6] at Forrester defined the "never trust, always verify" paradigm. However, these frameworks target enterprise networks with abundant computational resources, leaving edge-specific implementations unexplored.

**Computation Offloading in Edge Computing.** Mao et al. [2] provide a comprehensive survey of mobile edge computing offloading strategies, identifying latency minimization and energy efficiency as primary optimization targets. Kumar et al. [7] established foundational models for offloading decisions, while recent work by Wang et al. [8] addresses deep learning inference offloading. Lin et al. [9] specifically examine IoMT (Internet of Medical Things) offloading where data sensitivity intersects with latency requirements. Critically, none of these works incorporate cryptographic overhead into their decision models.

**Secure Edge Communication.** Tailscale's WireGuard implementation [10] offers cryptographically verified mesh networking with minimal configuration overhead—a significant advantage over traditional IPsec or service mesh approaches. Cloudflare's Magic WAN [11] demonstrates enterprise-

scale secure edge connectivity, while NVIDIA's work on TensorRT [12] enables high-throughput inference that could offset security overhead through accelerated processing.

**Gap Identification.** Existing literature treats security and offloading as orthogonal concerns. This project bridges this gap by developing an integrated cost model that treats cryptographic overhead as a first-class scheduling constraint alongside network latency and computational complexity.

## METHODOLOGY & VALIDATION

### Hardware Testbed: The Aura Rack

The experimental platform comprises a 10-inch 9U rack ("Aura") containing heterogeneous compute nodes:

- **Worker Nodes:** NVIDIA Jetson Nano (128 CUDA cores), Raspberry Pi 4 (4GB), Orange Pi 4A
- **Client Node:** ESP32-S3 (T-Display) with hardware AES-256 and SHA-256 accelerators
- **Network Infrastructure:** TP-Link TL-WR1502X router, TL-SG108 gigabit switch
- **Security Overlay:** Tailscale mesh network with WireGuard tunnels

### Experimental Scenarios

We evaluate four configurations to isolate security overhead effects:

1. **Baseline (Local):** 1024-point FFT executed entirely on ESP32-S3 using ESP-DSP library. Measures $T_{local}$ and $E_{local}$.
2. **Insecure Offload:** Task offloaded over raw TCP/IP to Jetson. Establishes $T_{network}^{insecure}$ baseline.
3. **Secure Offload (ZTA):** Identical offloading through Tailscale/WireGuard tunnel. Measures $T_{network}^{secure}$ including encryption/decryption overhead.
4. **Secure + Congested:** ZTA offloading with synthetic network impairment using Linux `tc` (traffic control) to inject 10-100ms jitter and 1-10% packet loss.

### Metrics

- **End-to-End Latency:** $T_{e2e} = T_{serialize} + T_{encrypt} + T_{transmit} + T_{decrypt} + T_{compute} + T_{return}$
- **Security Overhead Ratio:** $R_{sec} = T_{e2e}^{secure} / T_{e2e}^{insecure}$
- **Crossover Point:** Network latency $L^*$ where $T_{e2e}^{secure}(L^*) = T_{local}$
- **Throughput:** Tasks completed per second under sustained load

| Week | Deliverable | Status |
|---|---|---|
| Week 1-2 (Feb 7-21) | Literature review; Tailscale deployment on Aura rack | Proposal |
| Week 3-4 (Feb 22-Mar 7) | ESP32-S3 FFT implementation; baseline measurements | Development |
| Week 5-6 (Mar 8-21) | Offloading protocol; insecure/secure comparison | Progress Report |
| Week 7-8 (Mar 22-Apr 4) | Scheduler decision engine; congestion experiments | Testing |
| Week 9-10 (Apr 5-18) | Analysis; visualization; final documentation | Final Report |

## PROJECT PLAN & TIMELINE

## RISK MANAGEMENT

- **Risk:** ESP32-S3 memory limitations (512KB SRAM) may constrain encryption buffer sizes for large payloads.

  **Mitigation:** Segment FFT data into 256-sample chunks; utilize hardware crypto accelerators to minimize software memory footprint; fall back to FIR filtering (lower memory) if FFT proves infeasible.

- **Risk:** High network jitter (¿50ms) causing TCP retransmissions and measurement variance.

  **Mitigation:** Implement UDP-based transport with application-layer reliability; use median latency over 100 trials; characterize jitter distribution rather than assuming Gaussian.

- **Risk:** Tailscale coordination server dependency introduces external failure mode.

  **Mitigation:** Deploy Headscale (self-hosted coordination server) on Aura rack for fully local operation; maintain fallback to direct WireGuard configuration.

- **Risk:** Thermal throttling on Jetson Nano under sustained crypto+compute load.

  **Mitigation:** Monitor junction temperature; implement active cooling; characterize performance degradation curves.

## DUKE COMMUNITY IMPACT

This research directly supports Duke's strategic initiatives in secure IoT and edge computing. The validated security-performance models can inform the Duke Smart Home Program's sensor deployments, where student health and occupancy data require protection. The Aura rack testbed will remain available for future ECE 654 projects requiring heterogeneous edge infrastructure. Furthermore, the open-source scheduler implementation will be contributed to the Duke GitHub organization, enabling reproducibility and extension by subsequent research teams. The findings may also support Duke Health's exploration of edge-based medical device data processing, where HIPAA compliance necessitates Zero-Trust architectures but latency constraints demand careful optimization.

## References

[1] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," National Institute of Standards and Technology, Tech. Rep. NIST SP 800-207, 2020. DOI: 10.6028/NIST.SP.800-207.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017. DOI: 10.1109/COMST.2017.2745201.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," in *IEEE Internet of Things Journal*, vol. 3, 2016, pp. 637–646. DOI: 10.1109/JIOT.2016.2579198.

[4] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017. DOI: 10.1109/MC.2017.9.

[5] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," *NIST Special Publication*, vol. 800, p. 207, 2020.

[6] J. Kindervag, "No more chewy centers: Introducing the zero trust model of information security," *Forrester Research*, 2010.

[7] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," in *Mobile Networks and Applications*, vol. 18, 2013, pp. 129–140. DOI: 10.1007/s11036-012-0368-0.

[8] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020. DOI: 10.1109/COMST.2020.2970550.

[9] K. Lin, Y. Li, Q. Sun, S. Zhou, and L. T. Yang, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019. DOI: 10.1109/JPROC.2019.2922285.

[10] J. A. Donenfeld, "Wireguard: Next generation kernel network tunnel," in *Network and Distributed System Security Symposium (NDSS)*, 2017. DOI: 10.14722/ndss.2017.23160.

[11] Cloudflare, Inc., *Magic wan: Secure, performant connectivity for branch offices and data centers*, https://www.cloudflare.com/magic-wan/, 2021.

[12] NVIDIA Corporation, *Tensorrt: Programmable inference accelerator*, https://developer.nvidia.com/tensorrt, 2019.