

# Shriram Raja

Boston, MA

✉ [shriramr@bu.edu](mailto:shriramr@bu.edu) | 🏠 [shriram-raja.github.io](https://shriram-raja.github.io) | [in](https://www.linkedin.com/in/shriram-raja) [shriram-raja](https://www.linkedin.com/in/shriram-raja)

## RESEARCH INTERESTS

---

Scheduling, Resource Management, Synchronization, Virtualization

## EDUCATION

---

**Boston University**, Boston, MA Sep 2023 - Present  
Ph.D. in Computer Science *3.88/4.00*

- Advisor: [Dr. Richard West](#)
- Coursework: Introduction to Operating Systems, Computing Systems for Robotics, Advanced Algorithms, Artificial Intelligence

**Virginia Tech**, Blacksburg, VA Aug 2021 - May 2023  
Master of Engineering, Computer Engineering *3.96/4.00*

- Advisor: [Dr. Haibo Zeng](#)
- Project Title: Hybrid Priority Assignment for Global Fixed Priority Scheduling

**PSG College of Technology**, Coimbatore, India Aug 2017 - May 2021  
Bachelor of Engineering, Electrical & Electronics Engineering *9.76/10.00*

## PUBLICATIONS

---

### Published/Accepted:

[J1] Xuanliang Deng\*, **Shriram Raja\***, Yecheng Zhao, and Haibo Zeng, “Priority Assignment for Global Fixed Priority Scheduling on Multiprocessors”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2024 [[Paper](#)] [[Code](#)] [[DOI](#)]

- Proposed a Priority Assignment algorithm for Global Fixed Priority (G-FP) scheduling that combines the advantages of heuristics and response time estimation (instead of actual response time) to outperform existing methods by 25 percentage points on average.
- \* - contributed equally

## EXPERIENCE

---

**Boston University** Sep 2023 - Present  
Research Fellow *Boston, MA*

- Worked on the kernel lock in the [Quest RTOS](#)
- Helped develop a software development kit of a [Quest-V](#) based separation kernel which supports Quest and Yocto Linux guests.

**Virginia Tech** Mar 2022 - May 2023  
Research Assistant *Blacksburg, VA*

- Developed, implemented and evaluated a novel algorithm for Global Fixed Priority scheduling on multiprocessors, advised by [Dr. Haibo Zeng](#)

**Security Solutions, Marvell Semiconductor** May - Aug 2022  
Firmware Engineer Intern *Santa Clara, CA*

- Implemented FRAM Logging feature in LiquidSecurity Cloud Hardware Security Module to enable communication between the processor and the FRAM during boot-up.

- Developed skills to understand and work with a large codebase spread across different repositories.

## SKILLS & TECHNOLOGIES

---

**Programming Languages** C, C++, Python, and Verilog  
**Microcontrollers & Boards** Arduino, 8085, 89C52, MSP430, and Raspberry Pi  
**Tools** MATLAB, Linux Shell, Git, and LLVM  
**Systems** Quest RTOS, Linux Kernel, FreeRTOS

## TEACHING EXPERIENCE

---

CS 552 Operating Systems, Teaching Fellow, Boston University Fall 2024  
 ECE 5480 Cybersecurity & IoT, Grad Teaching Assistant, Virginia Tech Spring 2022, 2023, Fall 2022

## SERVICE

---

Tutorial: [Getting Started with the Quest RTOS and Quest-V Partitioning Hypervisor](#) RTSS 2024  
 Secondary Reviewer RTSS 2024, RTAS 2025

## PROJECTS

---

**i386 Custom Bare-metal Operating System** Fall 2023  
 Developed a bare-metal operating system for 32-bit x86 systems that uses preemptive FIFO scheduling and supports a simple file system.

**Scheduler for Real-Time Operating Systems** Apr 2022  
 Programmed different periodic scheduling algorithms, resource management protocols, and a polling server for FreeRTOS. Determined the best priority that can be assigned to the polling server by analyzing the response time of aperiodic tasks by varying periodic load for different test cases when executed on an Arduino Mega 2560.

**LLVM Optimization Pass** Apr 2022  
 Implemented the Lazy Code Motion (LCM) algorithm to eliminate redundant statements and move arithmetic expressions to the latest point in the program without modifying the functionality of the code. Evaluated the performance of the optimization pass using open-source benchmarks.

**CPU Profiling Tool for Linux** Nov 2021  
 Used Kprobe to track the cumulative run time and the number of times each task is scheduled by the Linux scheduler. Created and maintained a red-black tree that stores the run time of the tasks. The 20 most scheduled tasks are identified, and their stack trace is displayed using the /proc file system.