Application Memory Isolation on Ultra-Low-Power MCUs

Taylor Hardin, Ryan Scott, Patrick Proctor, Josiah Hester, Jacob Sorber, David Kotz

DARTMOUTH

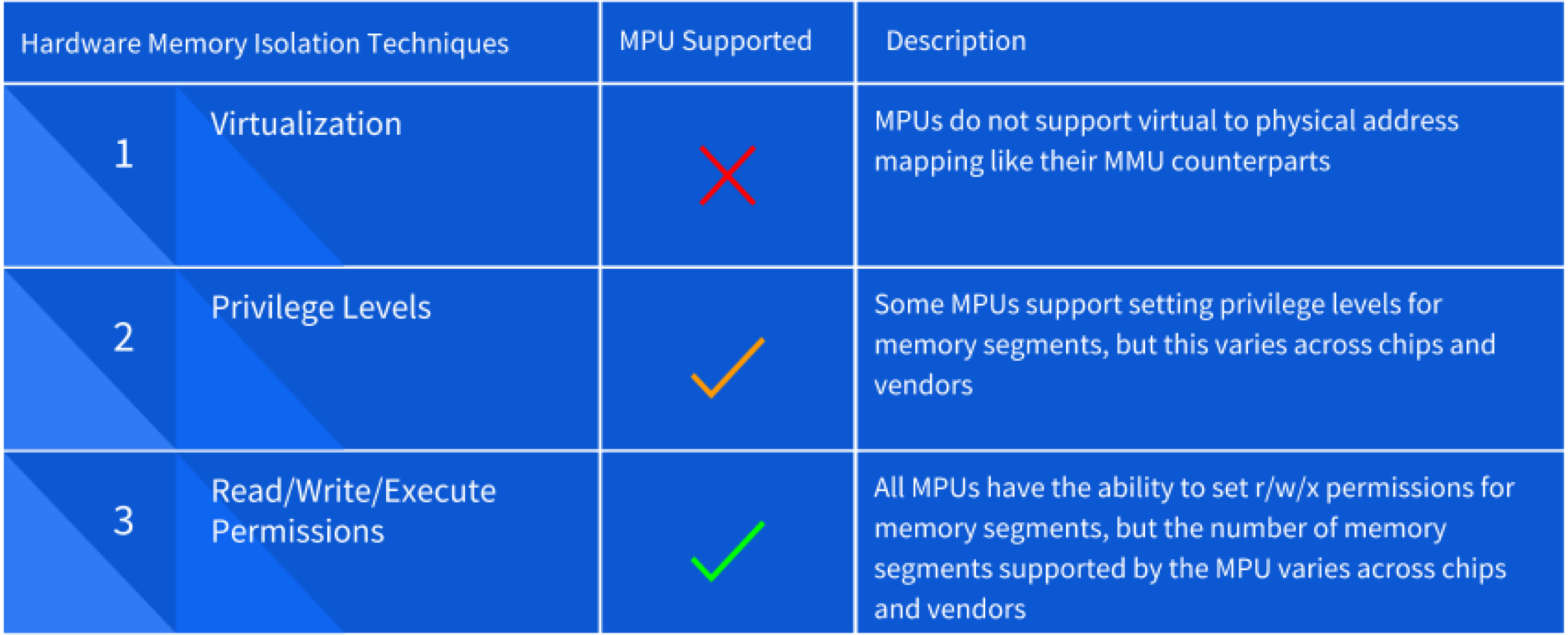
Northwestern University

CLEMSON UNIVERSITY

Motivation

- Many wearables and IoT devices utilize ultra-low-power MCUs to achieve long battery life

Motivation



Hardware Memory Isolation Techniques		MPU Supported	Description
1	Virtualization	✗	MPUs do not support virtual to physical address mapping like their MMU counterparts
2	Privilege Levels	✓	Some MPUs support setting privilege levels for memory segments, but this varies across chips and vendors
3	Read/Write/Execute Permissions	✓	All MPUs have the ability to set r/w/x permissions for memory segments, but the number of memory segments supported by the MPU varies across chips and vendors

Our Proposal

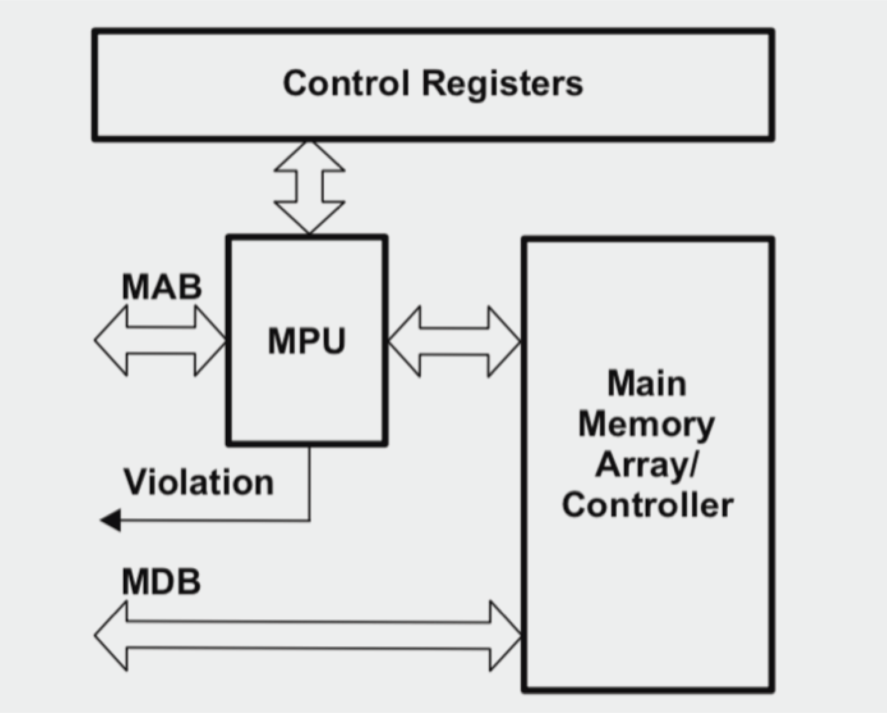
Utilize MPU to **relax** language restrictions and achieve **better** runtime performance

System Design: Platform

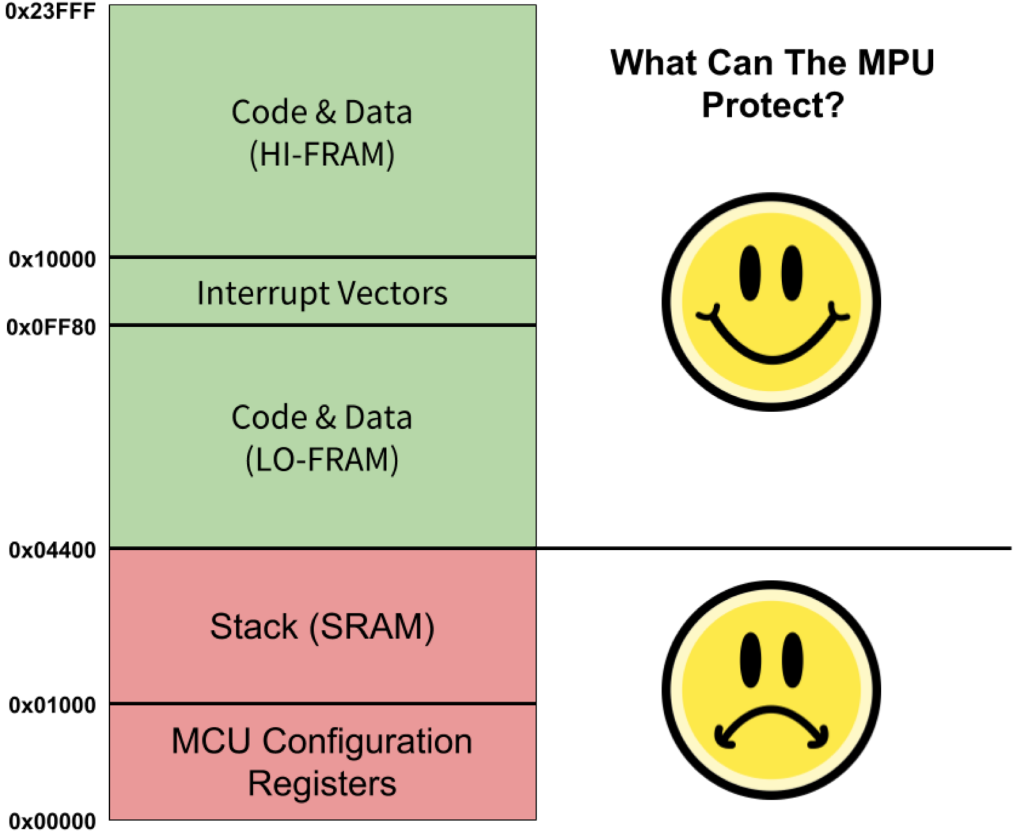
- Amulet Platform
  - Open-source software & hardware
  - Multi-application
  - Low-power MSP430 MCU
  - Memory isolation via language restrictions and runtime bounds checks

System Design: MPU Capabilities

- **No** privilege levels
- **3** variable size memory segments
- Only protects memory addresses **above** 0x4400



System Design: Memory Layout



What Can The MPU Protect?

0x23FFF

Code & Data (HI-FRAM)

0x10000

Interrupt Vectors

0x0FF80

Code & Data (LO-FRAM)

0x04400

Stack (SRAM)

0x01000

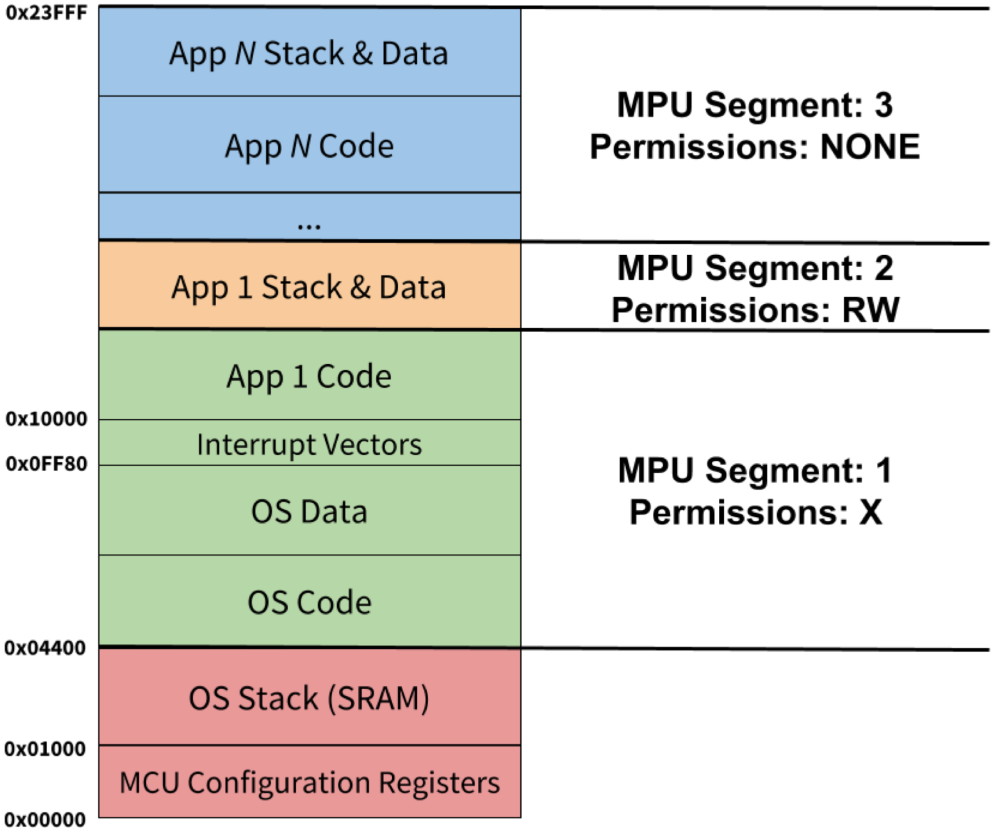
MCU Configuration Registers

0x00000

System Design: Memory Violations

- Memory Accesses
  - Application data
  - Indirect function calls
- Context Switches
  - Passing a pointer to the OS
  - Changing return address

System Design: Memory Layout

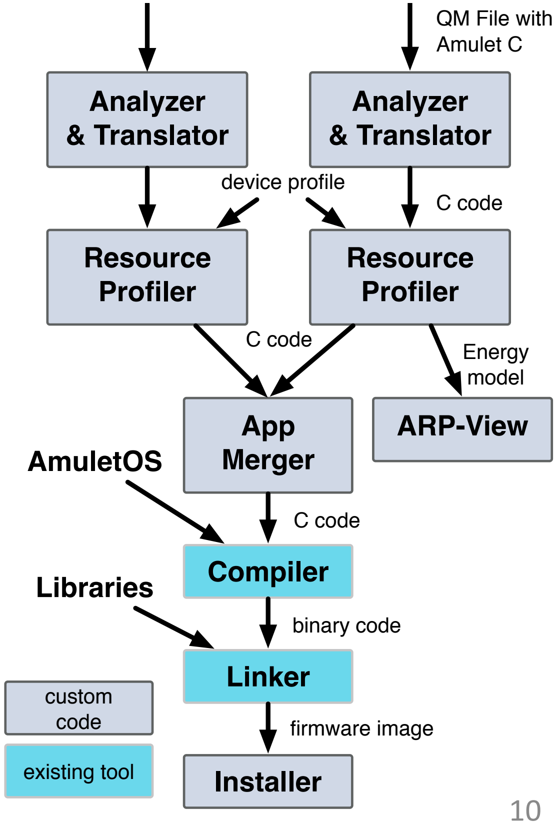


System Design: MPU Model

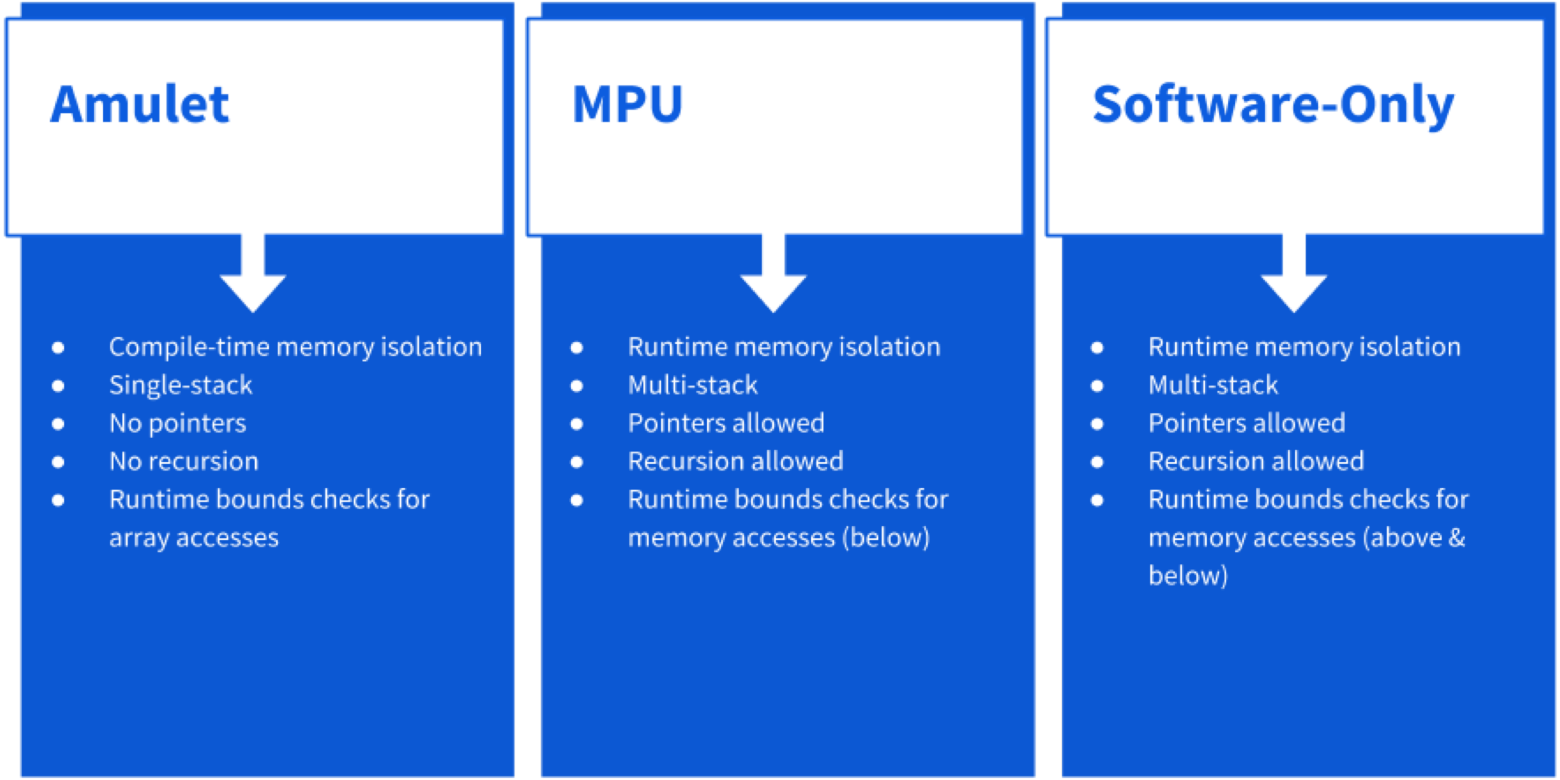
- MPU prevents memory accesses and indirect calls **above** the current app’s memory space
- Runtime software checks handle accesses and indirect calls **below** the current app’s memory space
- Each application has its **own** stack
- Runtime software checks verify return addresses

System Design: AFT

- Amulet Firmware Toolchain (AFT)
  - Analyze,
  - Transform
  - Merge
  - Compile



Eval: Isolation Models



Amulet

- Compile-time memory isolation
- Single-stack
- No pointers
- No recursion
- Runtime bounds checks for array accesses

MPU

- Runtime memory isolation
- Multi-stack
- Pointers allowed
- Recursion allowed
- Runtime bounds checks for memory accesses (below)

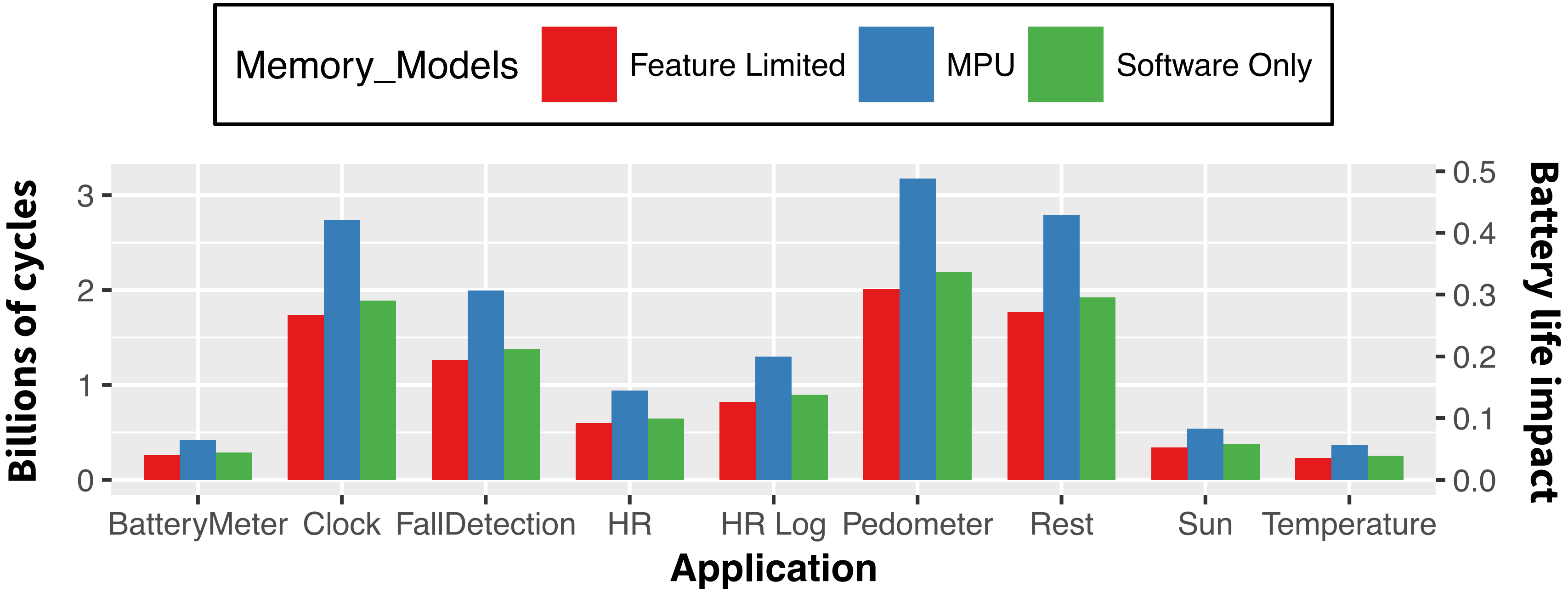
Software-Only

- Runtime memory isolation
- Multi-stack
- Pointers allowed
- Recursion allowed
- Runtime bounds checks for memory accesses (above & below)

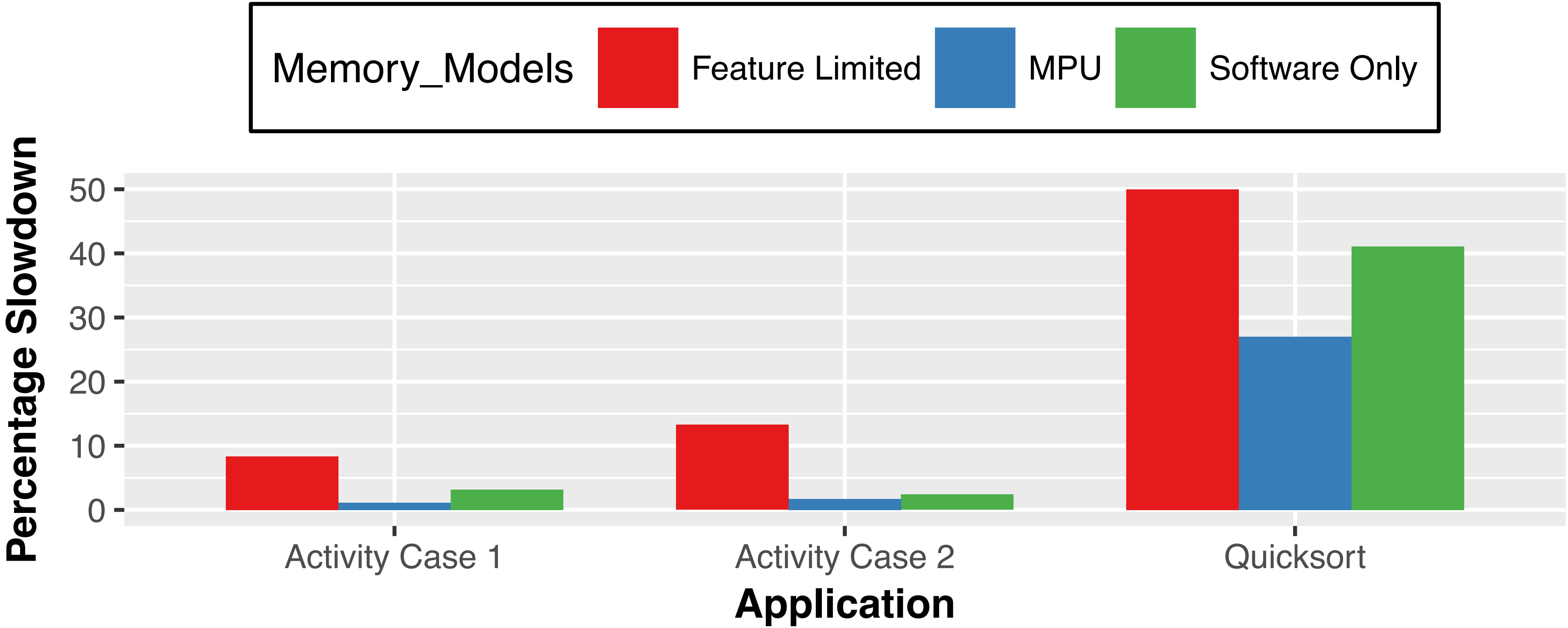
Eval: Simulation

- Simulated 9 applications from the Amulet suite using the Amulet Resource Profiler (ARP)
- Each application was simulated using
  - Amulet isolation
  - MPU isolation
  - Software-only isolation

Eval: Simulation Results



Eval: Amulet Deployment Results



Summary

- MPU can provide performance benefits for applications with high frequency of memory accesses
- While our approach was not effective for apps with frequent context switches, our MPU approach had, for all applications, less than 0.5% battery impact

Application Memory Isolation on Ultra-Low-Power MCUs

Contact: Taylor.A.Hardin.GR@dartmouth.edu
Amulet Platform: amulet-project.org

This research results from a research program at the Institute for Security, Technology, and Society, supported by the NSF under award numbers CNS-1314281, CNS-1314342, CNS-1619970, and CNS-1619950. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

DARTMOUTH

Northwestern University

CLEMSON UNIVERSITY