

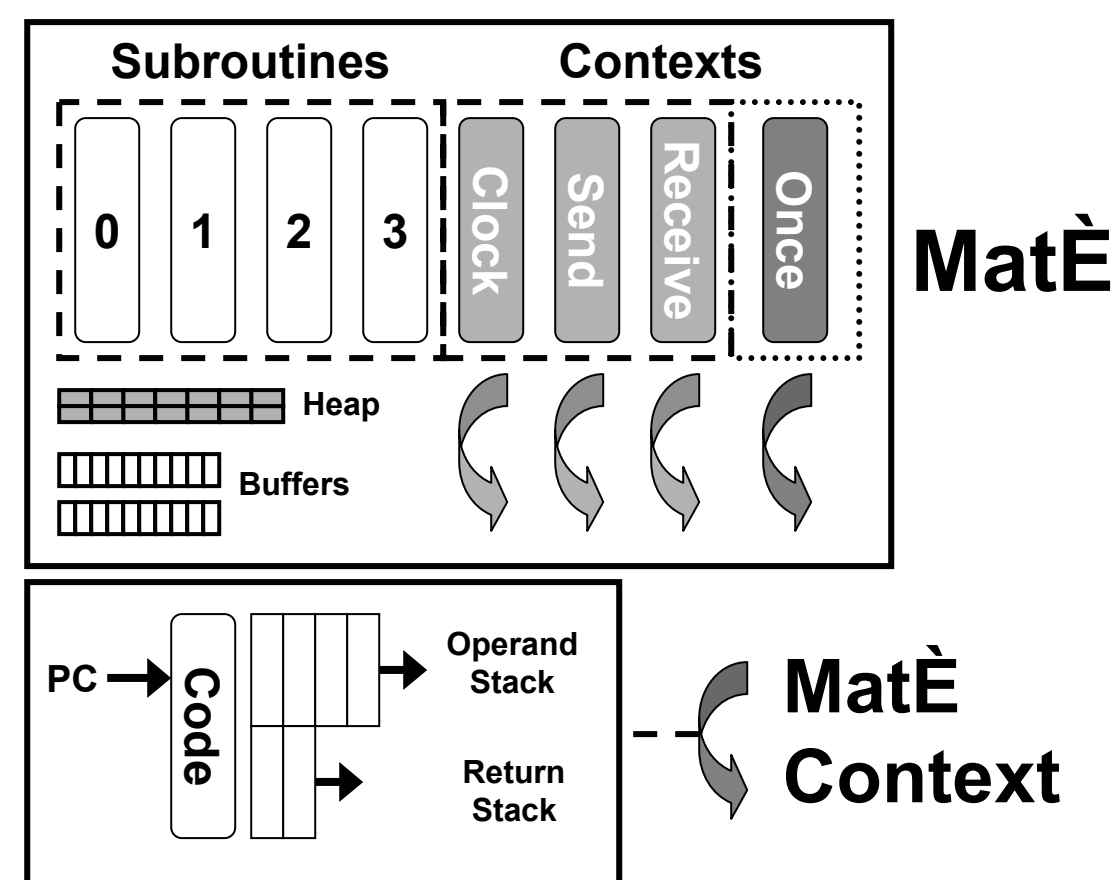
# MatÉ S<sup>3</sup>: Scalability, Security and Safety

Philip Levis and David Culler

UC Berkeley Computer Science Division, Intel Research: Berkeley

## MatÉ: Tiny Sensor Net VM

- Event-based bytecode interpreter
- Multiple concurrent contexts
- Incremental code loading (viral propagation)
- High code density



## Problems with MatÉ v1.0

- Need larger heap, making race conditions possible
- Explicit code forwarding scales badly
- Viral propagation is a two-edged sword
- Needs: Safety, Scalability and Security

## MatÉ S<sup>3</sup>

- New instruction set**
  - Buffer-centric
  - Retains all MatÉ 1.0 functionality
- Typed buffers**
- Triggered error state provides debugging info
- Eight variable heap**
- Scalability:** Implicit viral propagation
- Security:** End-to-end code integrity
- Safety:** Implicit context synchronization
  - Race-free
  - Deadlock-free
  - Safe for dynamic loading

## MatÉ Scalability: Version Vectors

- Capsule version summaries periodically transmitted
  - Decaying timer
  - Ranges from 1Hz to 1/300 Hz
- Timer reset when a newer vector is heard
- Capsules sent when an older vector is heard
- Timer reset when code is installed

## MatÉ Security Analysis

- TinySec provides symmetric link-level mechanism
  - Depends on physical mote security
  - Current motes require extensive resources to open

### Requirements

- Program integrity
- Routed data integrity

### Threats

- Adversarial capsules
- Routed data replay
- MatÉ data replay
- Traffic control (e.g., cut and paste attacks)

## MatÉ Security Mechanisms

### Signed capsules

- Modification of BiBa signing algorithm
- 24 byte signature, quickly verifiable (3 hashes)
- End-to-end integrity capsule check
- Finite number of signatures possible (~10<sup>5</sup>)

### Routed data sequence numbers

- Unique sequence number, end-to-end suppression

### Composition Vectors

- Specify valid combinations of capsules
- Specify which receive capsule is used

### MatÉ data capsule IDs

- Refuse data from older versions

## MatÉ Safety

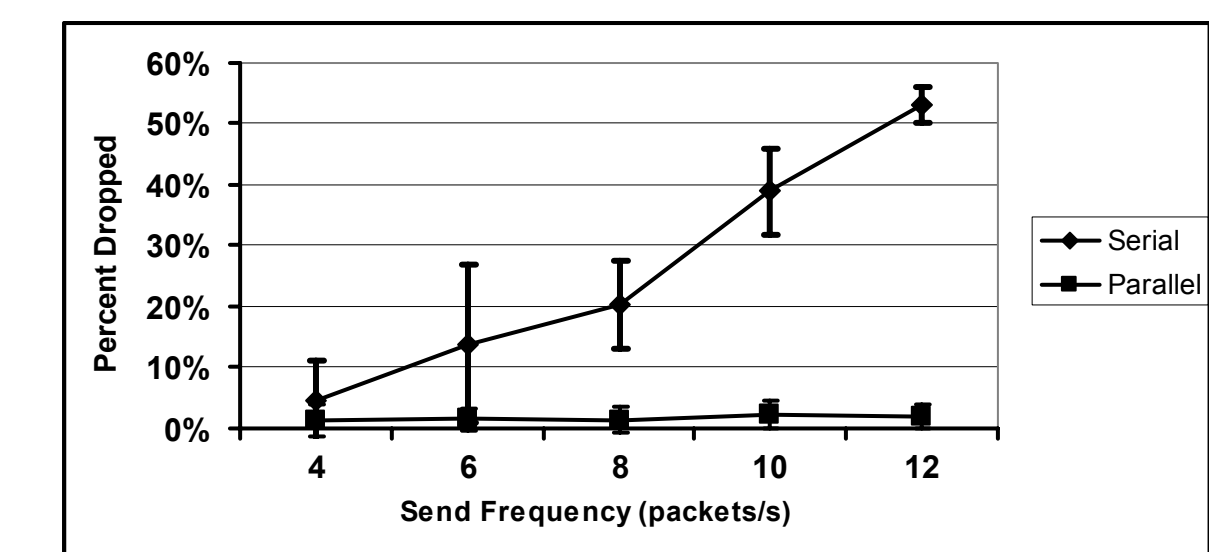
- Full program analysis on capsule install
  - Inexpensive ñ static naming
- Implicit locking, explicit yielding
  - Release locks across split-phase ops
  - Yield ñ in acquire set, Relinquish ñ not in acquire set
  - Reduced code size
- Fair-share ready to run scheduler
  - Instruction granularity interleaving
- Greedy FIFO resource waiting scheduler
  - Livelock starvation possible
  - Simple, fast, high parallelism

## Safety Overhead

- < 70 byte overhead
- Suspend/resume context == 3 VM instructions
- Full program analysis: < 4ms

## Improved Parallelism

- Compared forced atomicity with safe atomicity
  - Serial vs. parallel execution of handlers



## Future Work

- Merging with mottle to gain a programming model
- Capsule data segments
- Incorporation with TinyDB