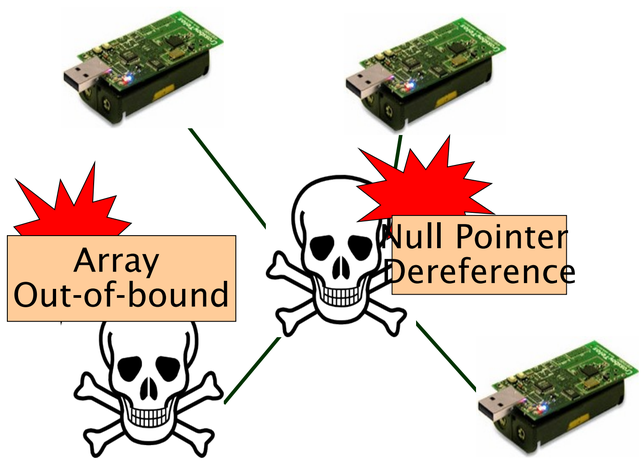
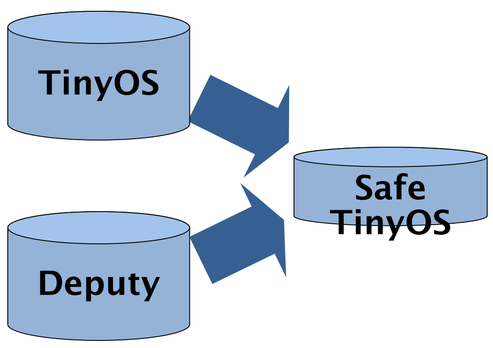
Surviving Sensor Network Software Faults

Yang Chen†, Omprakash Gnawali‡, Maria Kazandjieva*, Philip Levis*, John Regehr†
†University of Utah ‡University of Southern California *Stanford University

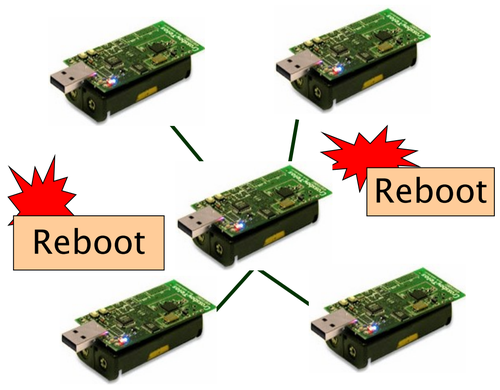


- large numbers of tiny, low-power wireless devices in inconvenient or remote locations
- in reality, WSN have lower availability and require significant human attention
- hardware-based memory protection is unavailable and unforeseen bugs can halt a deployment or result in unusable data

Safe TinyOS: Compiler-enforced Safety



- TinyOS is the dominant operating system for programming wireless sensor network devices
- Deputy is a source-to-source compiler for ensuring type and memory safety for C code

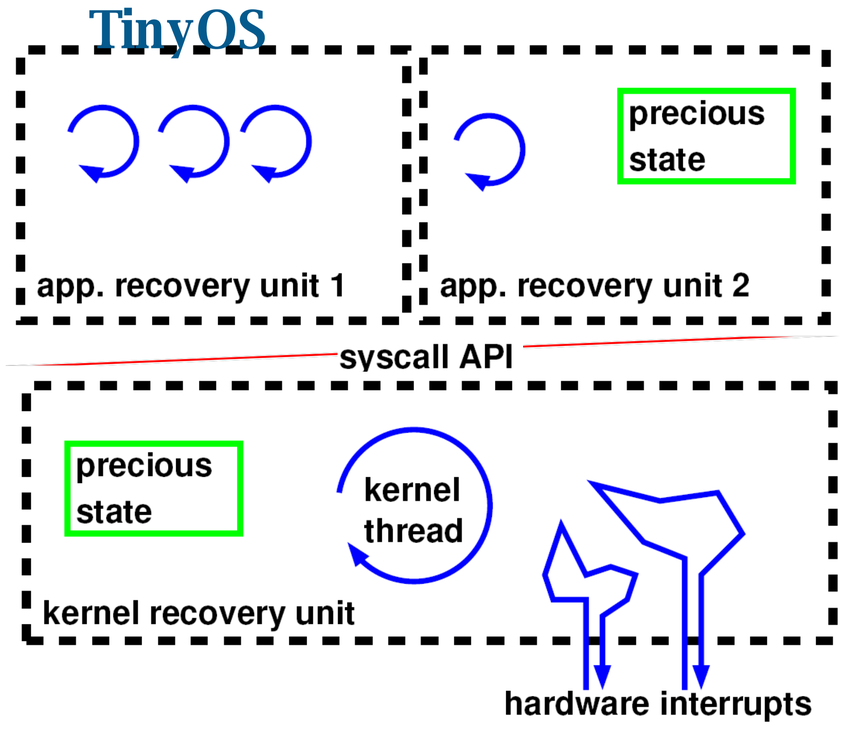


Naïve rebooting wastes energy and loses data.

How should a node respond to a safety violation?

Neutron: Surviving Sensor Network Software Faults

Neutron's Structure in TinyOS



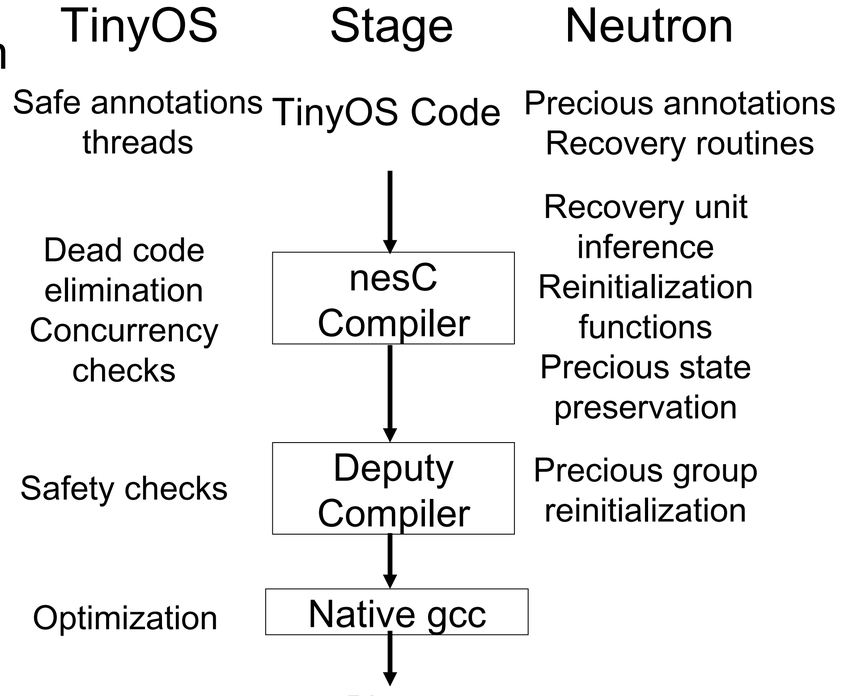
Neutron

a version of the TinyOS operating system that efficiently recovers from memory safety violations

Features

- dividing programs into recovery units; only faulting unit is rebooted; the TinyOS kernel itself is a recovery unit
- minimizing safety violation cost by maintaining "precious" states across reboots

Neutron's Toolchain



Recovery Unit

- Neutron derives application recovery unit boundaries at compile-time

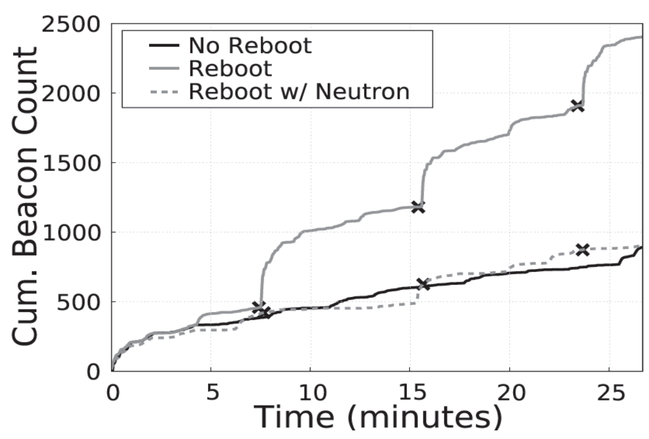
```
/* definition */
Configuration BlockingActiveMessageC
    @syncall_base() { }
/* instantiation */
Components BlockingActiveMessageC;
```

Precious State

- Neutron preserves precious data by introducing "precious" annotation
- The Neutron nesC compiler divides a recovery unit's precious variables into precious groups

```
TableItem @precious() table[MAX_ENTRIES];
uint8_t @precious() tableEntries;
```

Collection Tree Protocol Experiment



THE UNIVERSITY OF UTAH

USC UNIVERSITY OF SOUTHERN CALIFORNIA

THE LELAND STANFORD JUNIOR UNIVERSITY 1891