TARDIS: Software-Only System-Level Record and Replay in Wireless Sensor Networks

Matthew Tancreti, Vinaitheerthan Sundaram, Saurabh Bagchi, Patrick Eugster

Purdue University

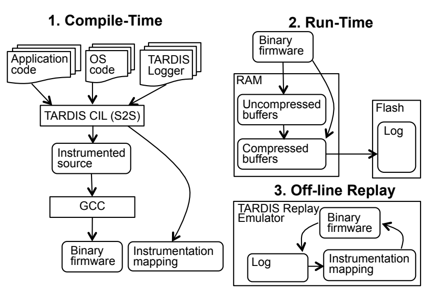
Problem Statement

- Despite best efforts, software defects are often encountered only after deployment of a Wireless Sensor Network (WSN)
- Prior work uses *record and replay* (the ability to reproduce an execution) to aid in debugging
- For example, TinyTracer can replay control flow and Envirolog can replay select variables and function calls
- TARDIS poses the question: is it feasible in software to record and replay every instruction and state of memory in a WSN?
- Challenges:
  - Small persistent storage for logging (1MB TelosB)
  - Small RAM for buffering (10KB TelosB)
  - Low CPU power (4MHz TelosB)

Solution Approach

- Record only the non-determinism present on μC
  - Values read from peripheral registers
  - Timing of interrupts
- Classify sources of non-determinism and compress in separate streams
  - 3 streams: state/timer register, generic register, interrupt timings
  - Low resource domain-specific compression techniques for each stream
- Buffer stream and write to flash during downtime
- Reconstruct events by feeding streams into emulator

TARDIS Architecture



- Source-to-source compiler identifies and instruments sources of non-determinism
- Buffer log in RAM before writing to Flash
- Emulator consults log and instrumentation mapping

Contributions

- Instruction and memory accurate record and replay for WSNs
- Classification of sources of non-determinism on μC and use of domain-specific compression
- Evaluation of resource costs for record and replay of WSN applications
- Demonstrate diagnosis of new defect in CTP

Recording All Non-determinism

1. Reads from peripheral registers
   - Peripheral registers contain values from external sources, e.g., ADC, I2C data, or timer
   - Reads are typically addressed statically
   - TARDIS-CIL identifies reads at compile time and adds code to store value
2. Timing of Interrupts
   - Hardware instruction counter not available, so we use software technique
   - Loop count and return address uniquely identifies when interrupt occurred
   - Every loop is instrumented with loop count increment instruction

Compression of Non-determinism

Baseline: Logging only non-deterministic registers
Log growth = 12.9 KB/s

Interrupts	12.8%
Timer registers:	11.2%
Data registers:	6.3%
State registers:	69.7%

TARDIS:
Log growth = 1.5 KB/s
(88.4% reduction)

Interrupts	51.3%
Timer registers:	23.4%
Data registers:	17.5%
State registers:	7.8%

- Non-determinism of registers
  - Not all peripheral registers are non-deterministic
  - In some registers only particular bits non-deterministic
  - For example, ADC12CTL1 is deterministic except for single busy flag bit
  - Design: only record non-deterministic bits
- Polling loops

```
while (IFG & TXFLG);
```

  - Example, interrupt register checked until transmitting flag is cleared
  - Assume polling loops are eventually exited
  - Therefore, no need to record read from IFG register
- Register masking pattern

```
not_done_transmitting = IFG & TXFLG;
```

  - Example, IFG masked except for single flag bit
  - Masked bits have no relevance to execution of code
  - Design: ignore masked bits
- Sleep-wake cycling and interrupts
  - WSNs depend on sleep-wake cycling for energy conservation
  - Interrupts normally require recording 16-bit return address and 16-bit loop counter
  - Sleep mode can only exit with an interrupt
  - Design: interrupts that exit sleep mode do not need timing logged
- Timer registers
  - Delta between timer reads or capture/compare register after interrupt is usually small
  - Design: record timer delta

Compression (cont.)

- State registers
  - State registers report a state, for example, interrupt flags indicating pending interrupt
  - Consecutive reads often repeat value
  - Design: encode state registers with RLE
- Data registers
  - For example, I2C data
  - Design: compression using light-weight generic compression, LZRW-T

Log Format

State/Timer Stream:

```
if type == state then write
   0b111<6-bit index><8-bit run_length><X-bit value>
if type == timer and delta < 4 then write 0b0<2-bit delta>
if type == timer and delta < 64 then write 0b10<6-bit delta>
if type == timer and delta >= 64 then write 0b110<16-bit delta>
```

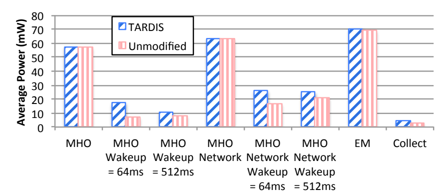
Generic Stream (LZRW-T):

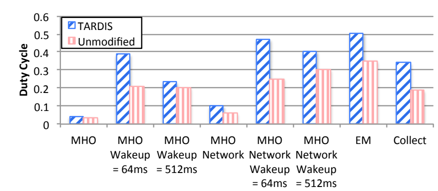
```
if no matching sequence found then 0b0<8-bit value>
if matching sequence found then 0b1<8-bit offset><8-bit length>
```

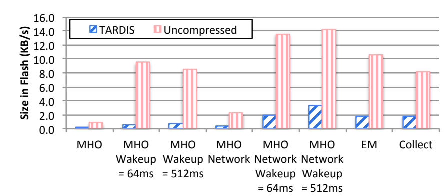
Interrupt Stream:

```
if loop_count == 0 then write 0b0<4-bit vector>
if loop_count < 256 then write
   0b10<4-bit vector><16-bit return_address><8-bit loop_count>
if loop_count >= 256 then write
   0b11<4-bit vector><16-bit return_address><16-bit loop_count>
```

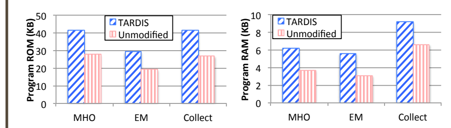
Resource Consumption

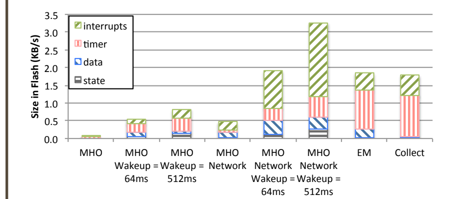




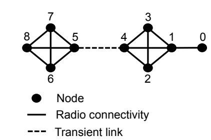


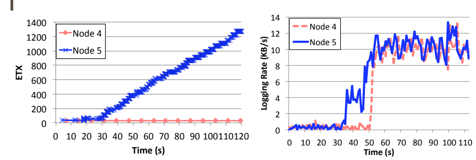
Resource Consumption (cont.)





Case Study: CTP Bug





- When partition forms for longer than 30 seconds, network takes 30 minutes to heal
- Goes against CTP principal of quick recovery from broken links
- ETX value continues to rise because of routing loops in partitioned nodes
- Caused by nodes on non-partitioned side sending beacons at lowest rate (they think network is healthy)
- Logging rate is low in healthy network but raises due to high traffic cased by routing loops

Acknowledgements

This work was supported in part by NSF grants ECCS-0925851 and CNS-0834529. The views expressed represent those of the authors and do not necessarily reflect the views of the sponsoring agency.

Reference

M. Tancreti, V. Sundaram, S. Bagchi, P. Eugster, "TARDIS: Software-Only System-Level Record and Replay in Wireless Sensor Networks," in Proceedings of the 14th ACM/IEEE Conference on Information Processing in Sensor Networks, IPSN '15, ACM, 2015.