Bombilla

By
NIRAV PRAJAPATI

Instructor: Dr. Ajay Gupta
CS 603: Sensor Networks
Spring 2004
Western Michigan University

What is Bombilla?

- Virtual Machine of Tiny OS
- Bytecode Interpreter that runs on top of Tiny OS and also uses some of the Tiny OS components.
- Base Architecture of Developing Tiny OS
- Virtual machines is called “Mate”.
- Bombilla is an instance of Mate.
- Allowsuser to write programs according to the application as in tiny os.

Why do We need VM like Bombilla?

- Large Networks with 100’s or 1000’s of nodes.
  - Example: Great Duck Island
- Some will fail during operation.
- Change of Function might be required on nodes.
- Almost impossible to manually recollect and program.
- Need for Virtual Programming.
- Bombilla provides required viral programming
Some Features of Bombilla

- Programs are written in high-level assembly language.
- Programs are very small and simple programming interface comparing with Tiny OS
  - Sense, Send: only 6 instructions
- Catches errors at runtime and provides debugging information.
- Resource management
- Programs can’t crash motes.
- Viral Programming

Architecture:

- Stack-based Architecture
- Context:
  - operand, return stacks (used at execution time)
  - capsules (Program)
- Heap - shared variables (capsules use them to store values)
- Buffers

Programming Environment:

- Programs are broken into capsules.
- Capsule Injector Interface for installing programs onto the motes.
Capsules:

- Programs are broken into capsules.
- Each capsule is of Max. 24 instructions
- Two types of Capsules

- event handler
  (Example: sense)
- subroutine
  (called from another capsule)

Sample Capsule:

CatToLed Application:

```
operand   operand
Stack
```

Sequence of Execution of Capsule

- Timer going off – event
- Event handler capsule is invoked
  example: send, receive
- Control jumps to the first instruction of the capsule
- All instructions are executed up to halt
- If subroutine is called from capsule return address is pushed onto the return address stack and control jumps to the first instruction of the subroutine.
- Some of the Underlying Tiny OS components are used in instruction processing.
Classes Of Instructions

Basic: Arithmetic operations - eq, halt, activating LEDs - push
M-class: Access message headers, used with only send and receive capsules - getms, getmb
V-class: To access 16 word heap (shared variable) - getvar, setvar
J-class: for loops and conditions - jumps, jumpc
X-class: only one instruction - push

Bombilla Errors

- If error encountered, program goes into the error state (i.e. trying to pop off an empty operand stack, stack overflow, buffer overflow, stack underflow)
- Capsules stops running
- Now on each clock event all LEDs toggles
- Packet of BombillaErrorMsg is send over UART

```c
typedef struct BombillaErrorMsg {
    uint8_t context;
    uint8_t reason;
    uint8_t capsule;
    uint8_t instruction;
} BombillaErrorMsg;
```

Sense with Bombilla

- Support two sensors: light sensor (1), temperature sensor (2)
- SenseToLeds:
  ```c
  pushc 1  // 1 is pushed onto operand stack
  sense   // light sensor reading of 10 bit is pushed
  cast    // pushes value equal to reading's magnitude
  pushc 7  // 7 is pushed onto the operand stack
  shiftr  // bitshift value to right
  putled  // top 3 bits are displayed onto the mote
  ```

Typedef struct BombillaErrorMsg {
Capsule Forwarding or Viral Programming

- **Density Adjusting Algorithm:**
  - Each mote maintains a time interval of length $T$.
  - Pick a random time $t$ from $T$ in which to transmit the summary of capsule version.
  - If an identical version of summary than discarded; if mote hears version summary with newer capsules than it has, it broadcasts the needed capsules.

---

Interface: Capsule Injector

- **Command:** `net.tinyos.vm.asm.CapsuleInjector`

- **References:**
  - [webs.cs.berkeley.edu/tos/api/tinyos-1.x/doc/tutorial/bombilla.html](webs.cs.berkeley.edu/tos/api/tinyos-1.x/doc/tutorial/bombilla.html)
  - [www.cs.virginia.edu/~qc9b/fall03cs851/mate_damon_jo.ppt](www.cs.virginia.edu/~qc9b/fall03cs851/mate_damon_jo.ppt)
Thank you