Function call traces are readily understandable

- **Function call traces** record the sequence of executed functions
- **Execution log from function call traces** is readily understandable
- Most programmers already design and think about systems in functional units
- Other tools already present data at a functional granularity, i.e. debugger back traces

Function call traces are very expressive

- **General logging systems** often record multiple types of events
- In most cases it is trivial to map events onto functions
- Interrupt events via the interrupt handler function
- Message send events via the message send function
- Context switches via a timer interrupt function and scheduler function

### Problem Description: Compacting call traces maximizes their utility

**Efficient function call trace logging mechanisms are not immediately obvious**

- Common implementation uses **globally unique function identifiers** to construct function call traces
- Each function is assigned a globally unique identifier
- A preamble is added before the body of each function to **record the identifier** of each executed function at runtime
- Identifier assignment and program instrumentation is easy to implement

### Proposed Solution: Compact call traces through alternate logging techniques

**CFG Decision Logging**

- Minimal scoping accomplished by providing one name space for each statement
- Non-control flow statements have at most one “next called” function
- Control flow statements with n branches have at most n “next called” functions
- Very compact token encoding
- Results in encoding the program’s runtime control flow decisions from which call traces may be inferred
- Uses the same caller side support required by local function identifier logging

```c
int handle_light_data(uint8_t light_intensity) {
    static uint16_t dark_count = 0;
    static uint16_t average_light = 0;
    delta = abs(average_light - light_intensity);
    if (delta > DETECTION_THRESHOLD) {
        broadcast_detection_event(light_intensity, NODEID);
        log(ID_TRUE);
        average_light = ewa(average_light, light_intensity);
    } else {
        log(ID_FALSE);
        ++dark_count;
        if (light_intensity < 10)
            return dark_count;
    }
    if (delta > DETECTION_THRESHOLD) {
        broadcast_detection_event(light_intensity, NODEID);
        log(ID_TRUE);
        average_light = ewa(average_light, light_intensity);
    } else {
        log(ID_FALSE);
        ++dark_count;
        if (light_intensity < 10)
            return dark_count;
    }
    return dark_count;
}
```

**Reducing the Logged CFG**

- **CFG decision logging** captures more data than is necessary
- Reduction over a program’s CFG returns nodes that effect function call traces