



Remote Incremental Adaptation of Sensor Network Applications

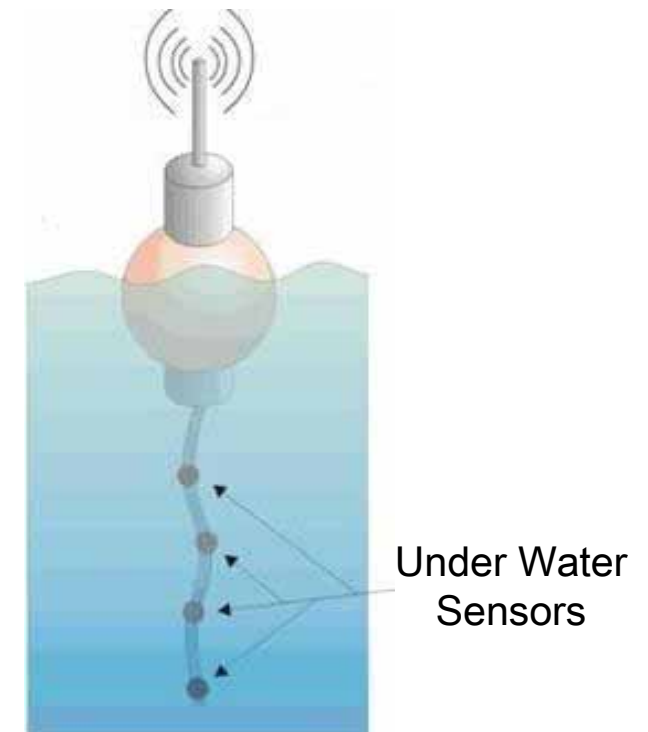
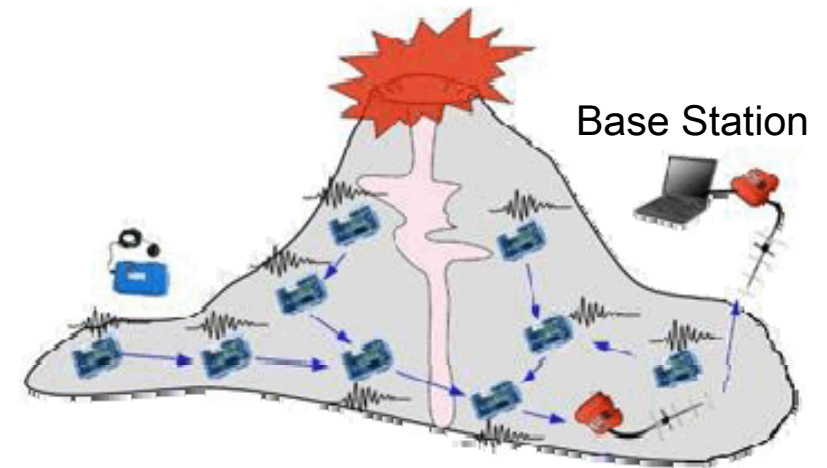
Waqas Munawar, Olaf Landsiedel, Muhammad Hamad
Alizai, Klaus Wehrle

<http://ds.cs.rwth-aachen.de>

waqaas.munawar@rwth-aachen.de

Motivation

- **WSN require uninterrupted operation indefinitely**
- **Customizing the system to the environment**
 - ▶ Feature upgrades
 - ▶ Retasking of the System
- **Post-deployment software updates are common**
 - ▶ Bug Removal



- **State of the art**

- ▶ Image replacement
- ▶ Virtual machines
- ▶ Dynamic OS's

Disadvantages:

Energy wastage, Restricted reconfigurability, Clean slate approaches

- ▶ **TinyOS** – A framework to generate application specific OS
 - Component based architecture
 - Select app components, statically analyze and optimize
 - **Supports only full binary upgrades**

- **Goals**

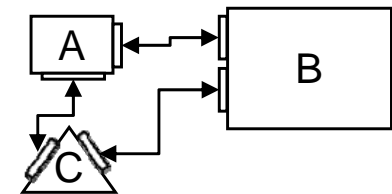
Energy Efficient, Fully configurable, Integrated and Transparent,
Fulfilling usual Embedded System Constraints

- Motivation
- **Strategy**
- **Architecture**
 - ▶ Overview
 - ▶ Components
 - ▶ Update Procedure
- **Evaluation**
 - ▶ Transfer cost
 - ▶ Update cost
- **Conclusion & Future Work**

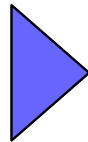
TinyOS as a base

- ▶ Seasoned code repository
- ▶ Wide user base
- ▶ **Runtime wiring**

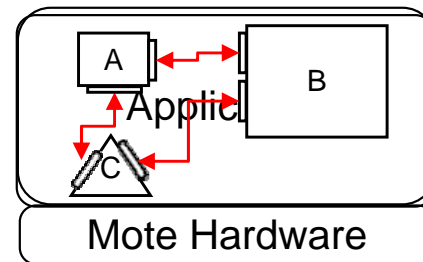
Mote Reconfiguration



Appropriate
Components wired
together



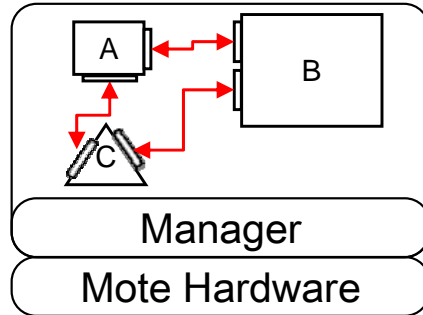
Compiled +
Linked +
Loaded



Mote reconfigured
with a new
Monolithic Binary

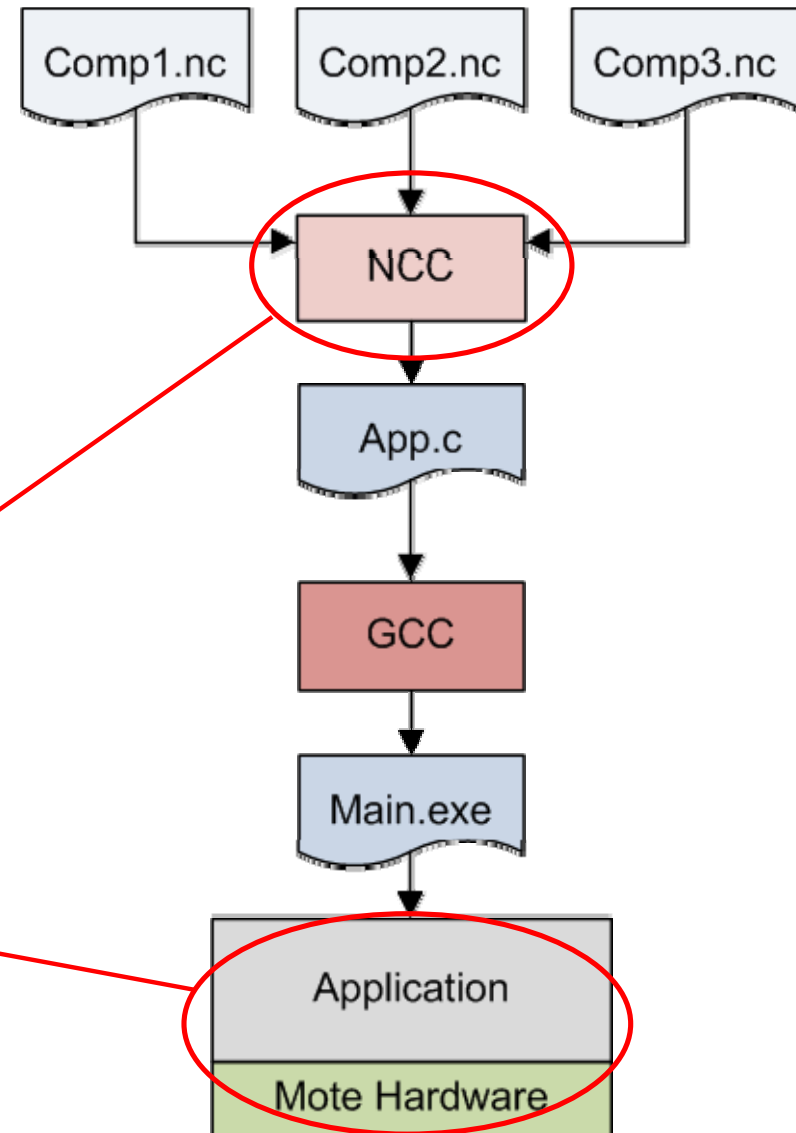
- Component structure Preserved

Runtime Wiring



Required

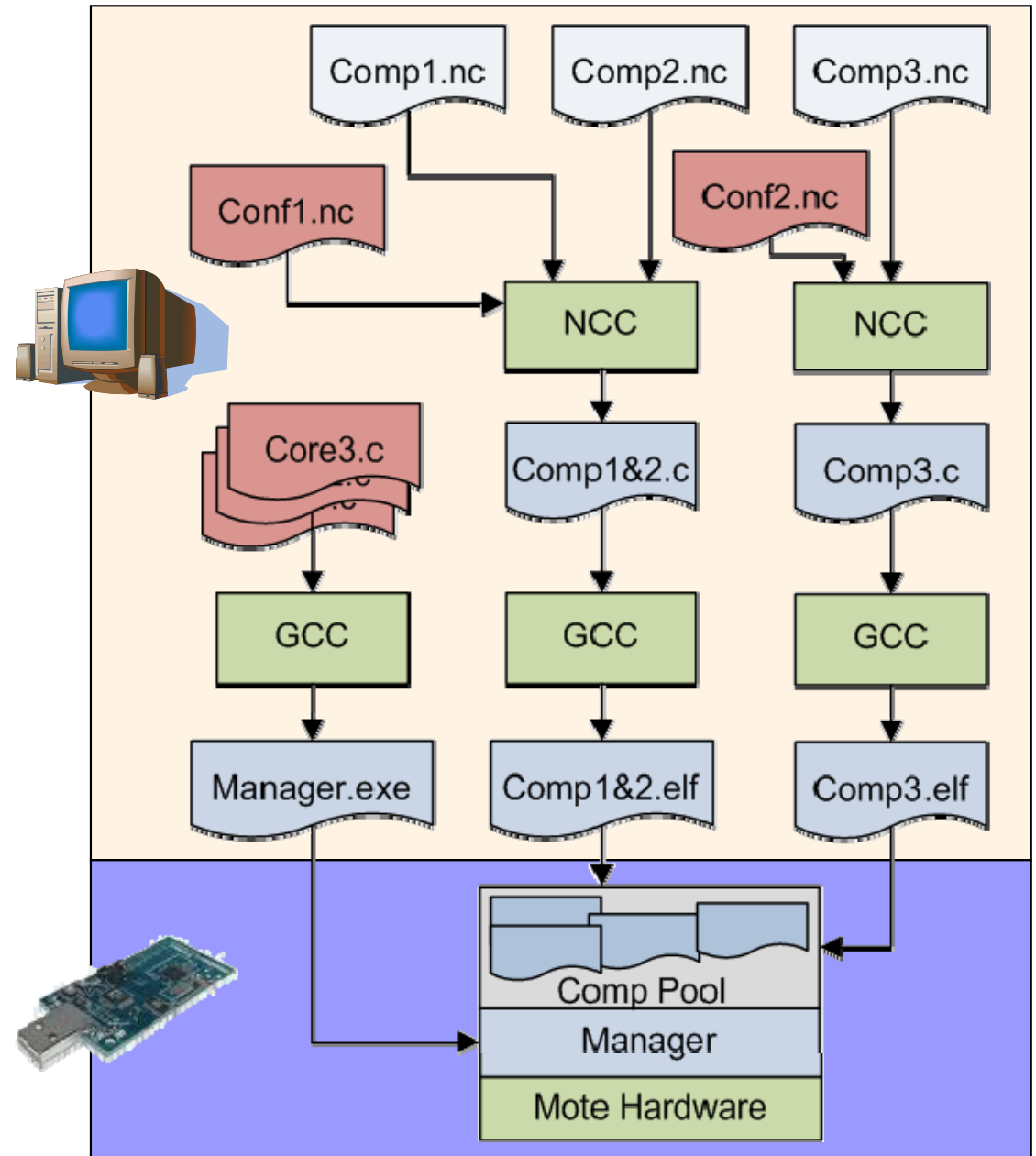
- ▶ Isolate individual nesC components
- ▶ A runtime Manager/Linker



Traditional TinyOS Mote Reconfiguration Process needs modifications

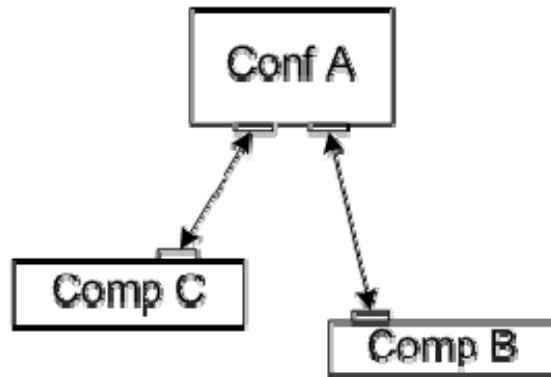
Main Components

- ▶ Host
 - Configuration Generator
- ▶ Mote
 - Manager (**TinyMan**)
 - Linker
 - Memory Manager
 - Interrupt Router
 - File System
 - Controller
 - Global Symbol Table

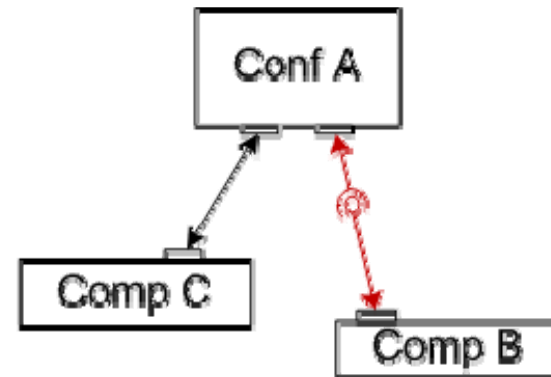


Configuration Generator

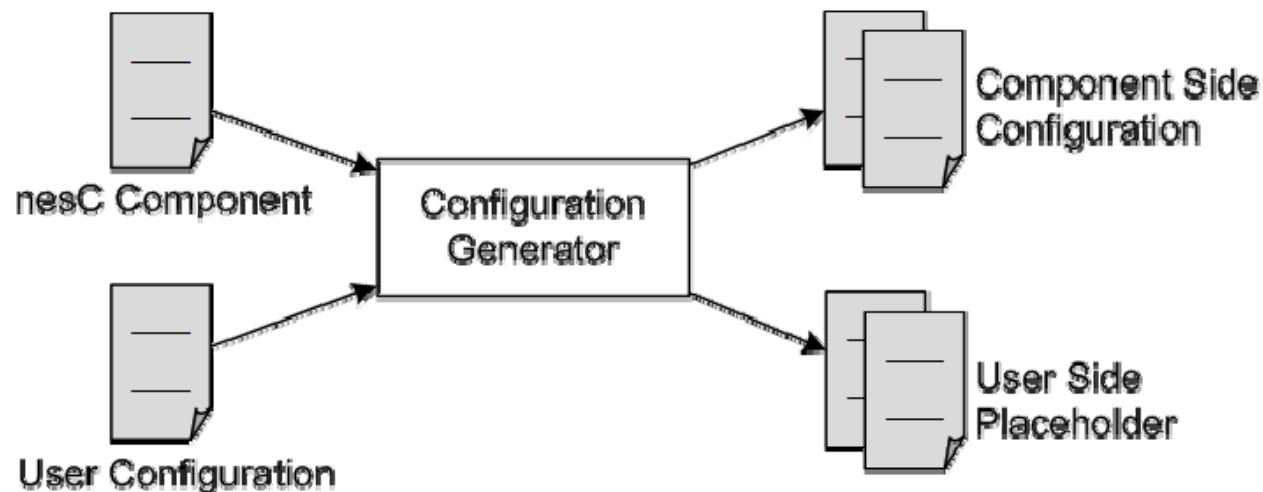
- ▶ Disambiguates NCC's operation



A Normal TinyOS app

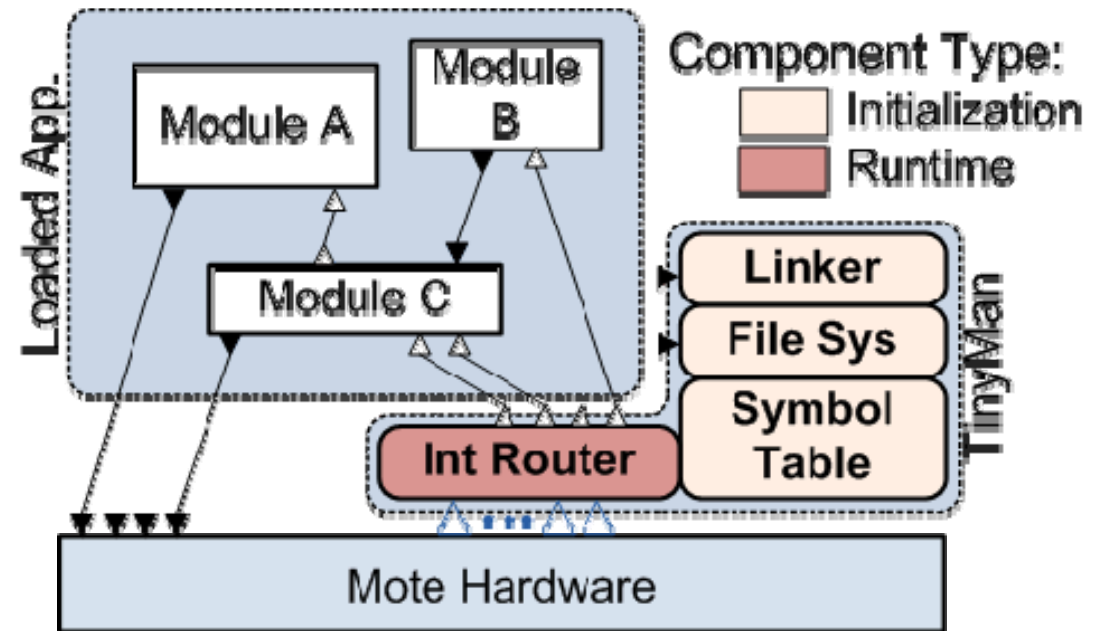


Separately compiled TinyOS app

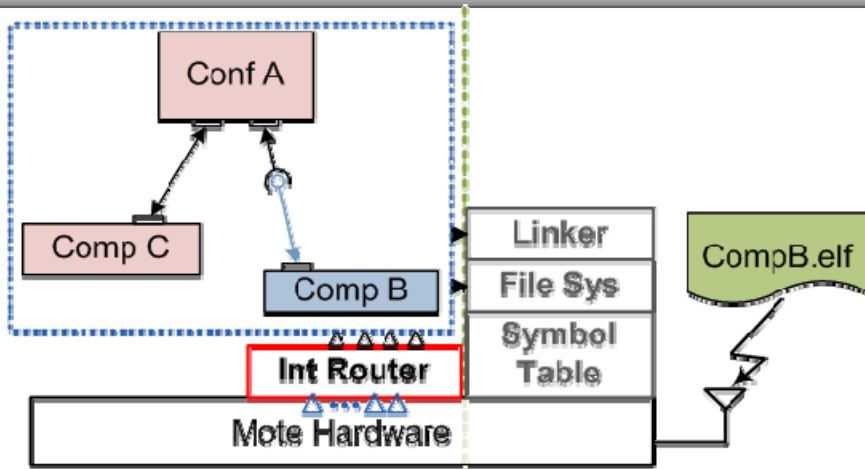


TinyMan

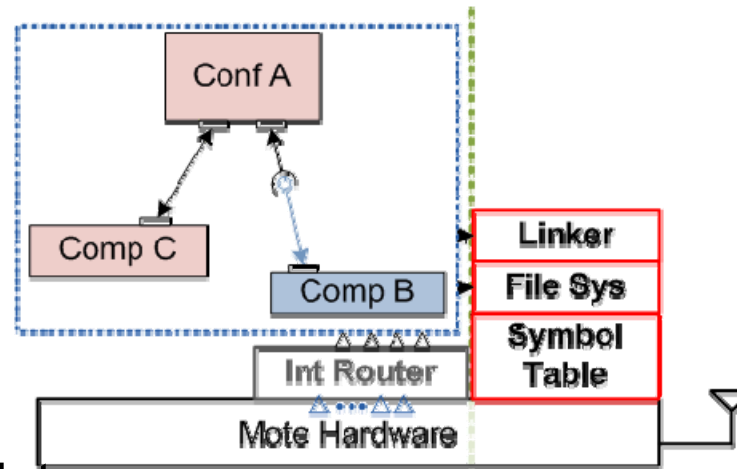
- ▶ Linker
 - Memory Manager: Manages allocation of internal flash
 - Loader: loads the linked Objects into program memory
- ▶ Interrupt Router
 - Reroutes interrupts to the functions registered as ISRs
- ▶ File System
 - Manages storage and retrieval of ELF objects to and from non-volatile storage.
- ▶ Symbol Table
 - Manages list of available symbols for cross linking support



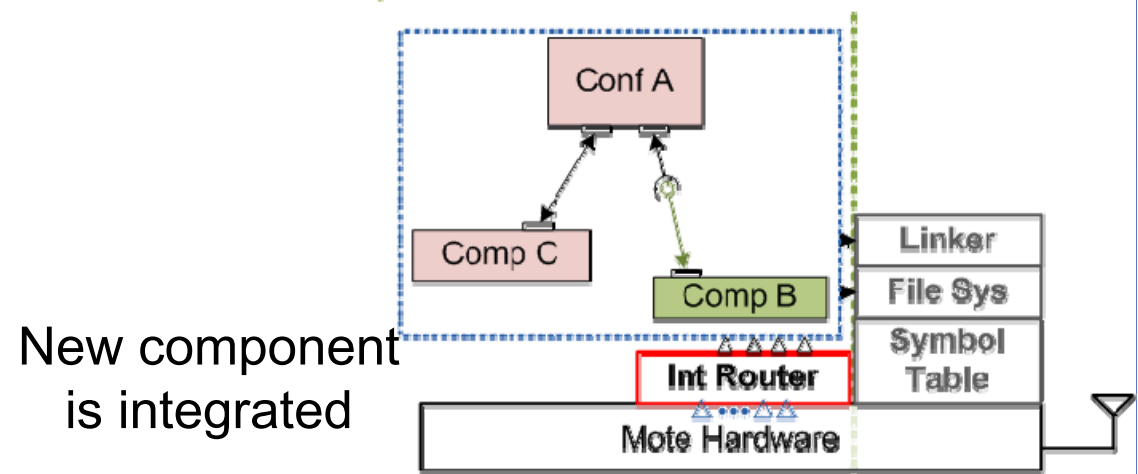
Architecture – Update Procedure



A new version of a component is received



After an update command is received



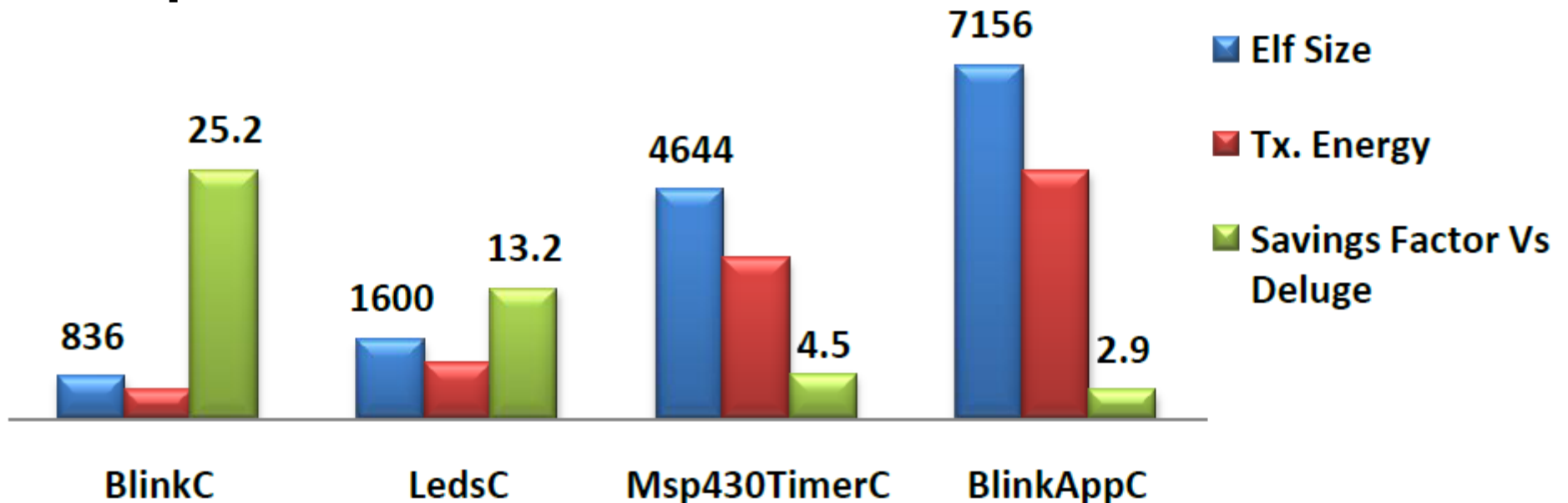
New component is integrated

Evaluation – Update Cost

- **Update Cost in terms of Energy Include**

- ▶ Multihop communication cost
- ▶ Cost of processing a received update

- **Multihop Communication Cost**



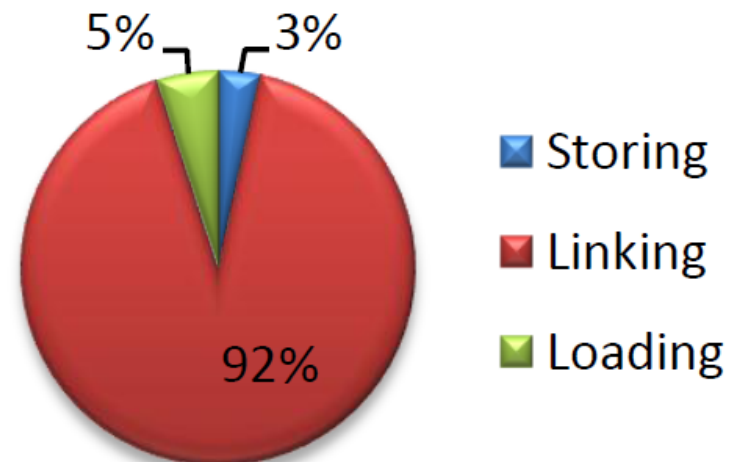
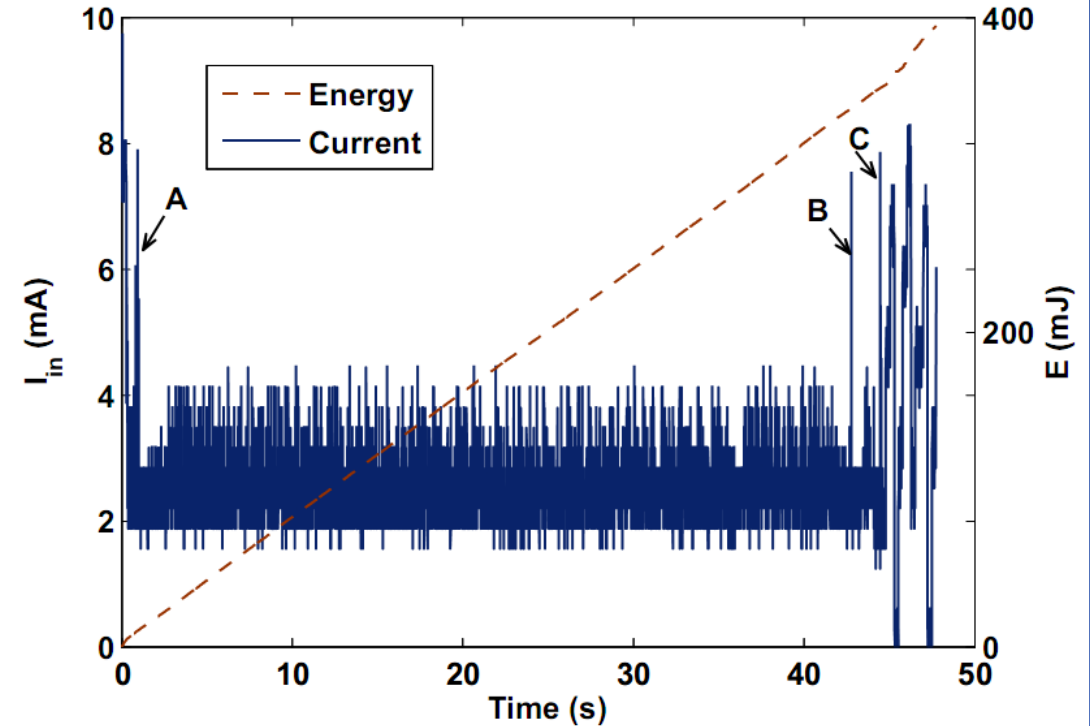
- ▶ Transfer cost proportional to the size of object

Moderate energy consumption for transfer

• Update Processing Cost

- ▶ Linking consumes most energy
 - Exponentially proportional number of string comparison operations
- ▶ Does not depend on size of update

Linking: prime candidate for optimization



Conclusion

- ▶ Sensor network applications reconfigured
 - Remotely, Incrementally, With selectable granularity
- ▶ Transparent operation – user friendly
- ▶ Reasonable Memory footprint
- ▶ Almost zero performance overhead



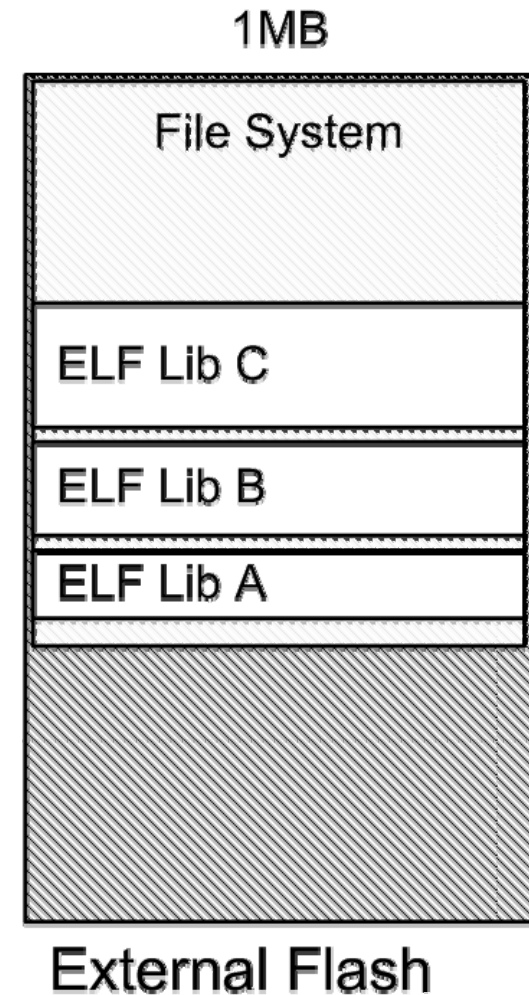
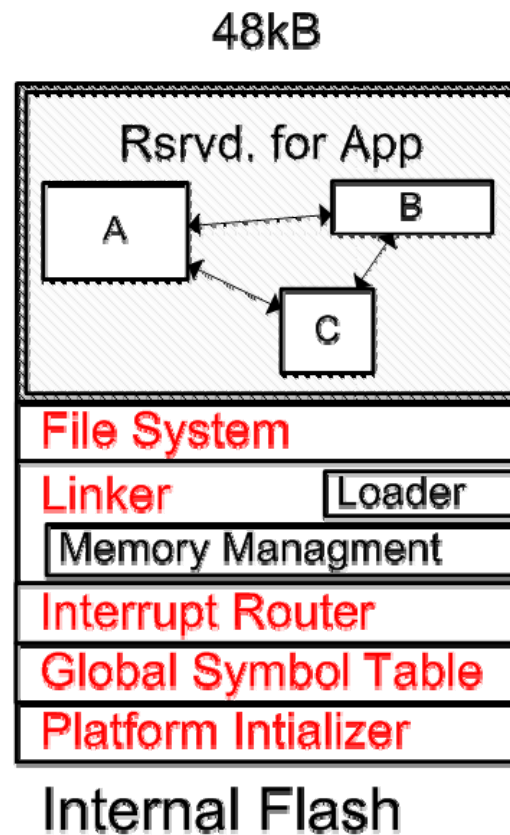
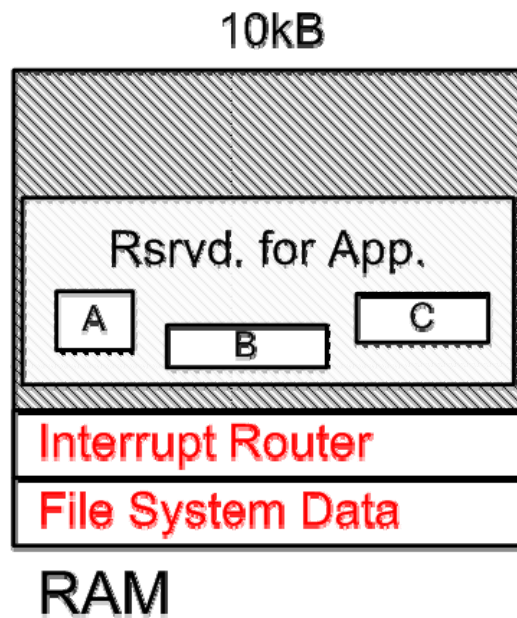
Future Work

- ▶ Automating the Configuration Generator
- ▶ ELF format optimization
 - Size Reduction due to 16 bit addressing is 8%
- ▶ Wiring compatibility checks
- ▶ Versioning scheme

Thank you Questions ?



Sys Internals

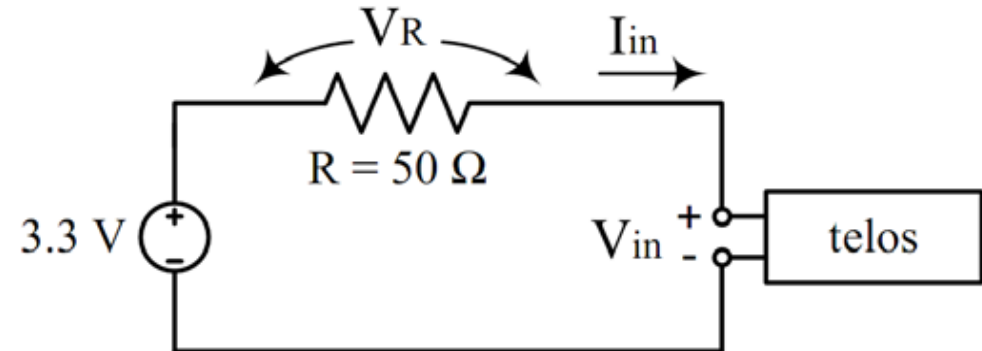


Related Work

Category	Solution	Update Mechanism	Pros /Cons
Image Replacement	Xnp	New Binary Image	+ No linking Required + Relatively transparent operation + No Execution Overhead - Large communication overhead
	Deluge		
Virtual Machine	Maté	New Script	+ small communication overhead - Coarse configuration granularity - Large execution overhead
Dynamic Operating System	Impala	New Application	- Fixed kernel + App side API
	Contiki	New ELF File	- Independent from TinyOS - Multiple ELFs not supported
	SOS	New PIC Module	- PIC modules
	FlexCup	New Module	- Not compatible with new nesC features
	FiGaRo	New Module	- Clean slate approach, A new C API used

- **Evaluation Criteria**

- ▶ Update Cost
- ▶ ELF format suitability
- ▶ Memory Footprint
- ▶ Performance Overhead

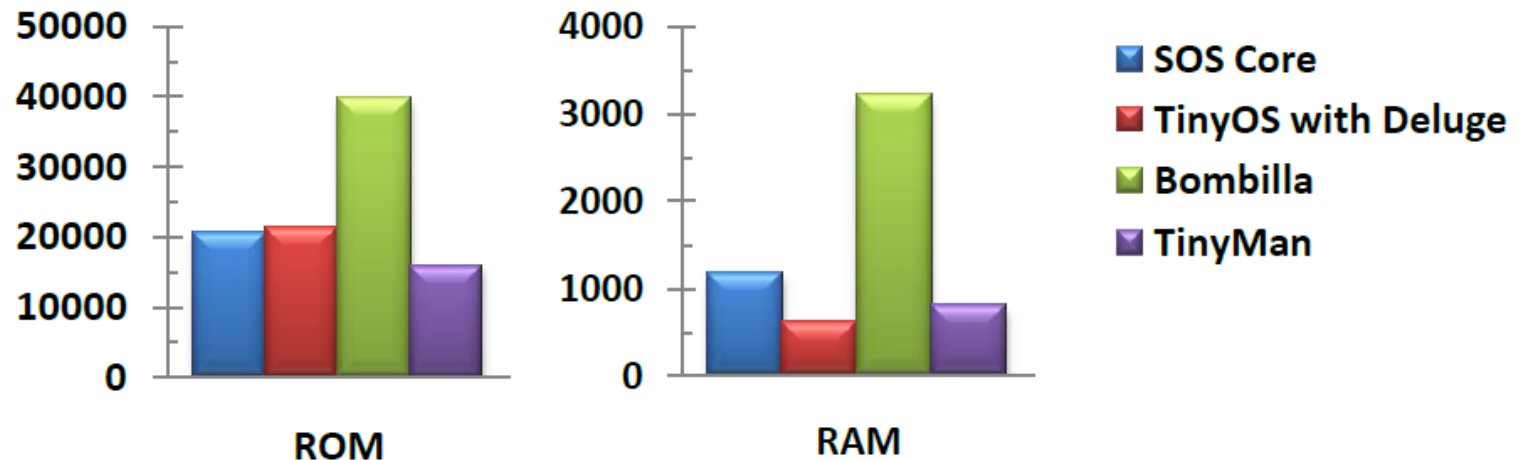


- **Setup**

- ▶ Platform used: telos rev B
- ▶ Applications Used
 - FFT Calculator: A processor intensive application with no IO
 - Blink: An IO intensive application, Uses Timers
 - Radio ping: Application to analyze Multihop Transfer Costs

- **Memory Footprint - On telosB**

- 7.7 % of RAM, 32 % of total program memory, 1.4% of total non-volatile storage



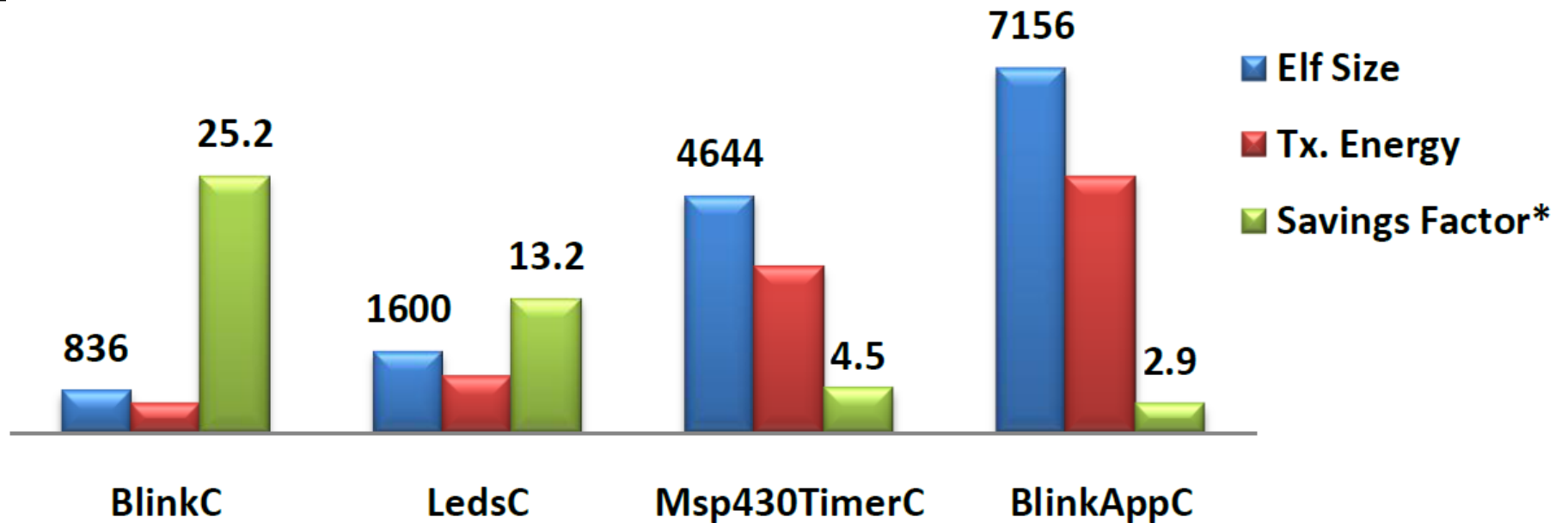
System Name	Flash ROM (bytes)	RAM (bytes)
SOS Core	20464	1163
TinyOS with Deluge	21132	597
Bombilla Virtual Machine	39746	3196
TinyMan	15826	792

- **Performance Overhead**

- ▶ A worst case delay of 23 cycles in interrupt processing

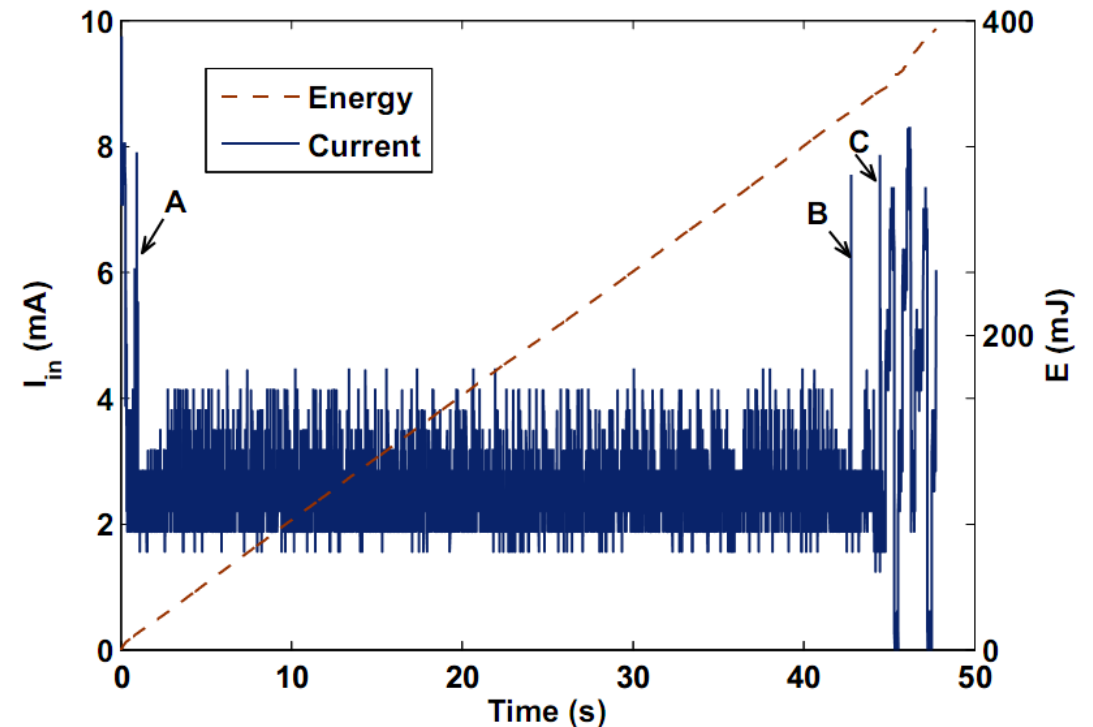
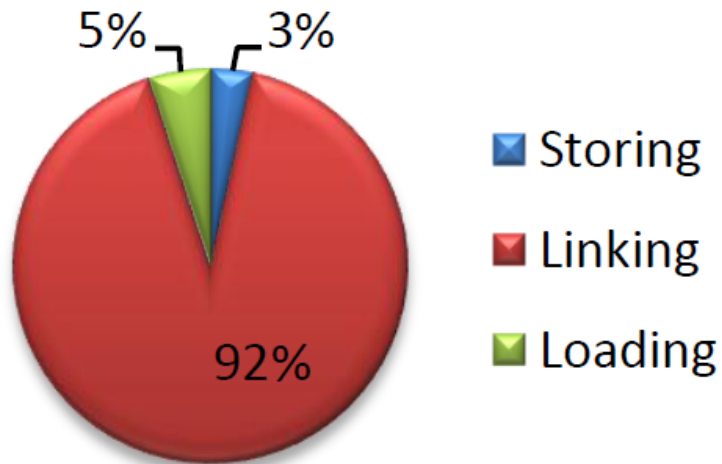
Multihop Communication Cost

App.	Components	ELF Size	Tx. Energy (mJ)	Saving Factor (Vs. Deluge)
Blink	BlinkC	836	58.77	25.2
	LedsC	1600	112.48	13.2
	Msp430TimerC	4644	326.47	4.5
	BlinkAppC	7156	503.06	2.9
	Total		1000.78	--



Update Processing Cost

Application	Steps	Size (B)	Time (s)	Energy Consumed (mJ)
Blink	Storing	14236	0.9	11.4
	Linking	14236	43.2	305.1
	Loading	4756	1.5	16.7
	Total		45.6	333.2



Detailed Memory Footprint

Component		Flash ROM (bytes)	RAM (bytes)
File System		5630	228
Platform Initializer		454	0
Linker	Linker Main	368	4
	Memory Management	344	84
	Loader	2460	24
Interrupt Router		1092	30
Global Symbol Table		1028	74
Node Runtime		390	0
Hardware Drivers	UART	102	259
	Flash Write	158	4
	SPI	54	0
	Ex Flash	726	0
Library Reference		3020	85
Total		15826	792

ELF Sections Detail

