Barry Porter

Lancaster University

Lorien

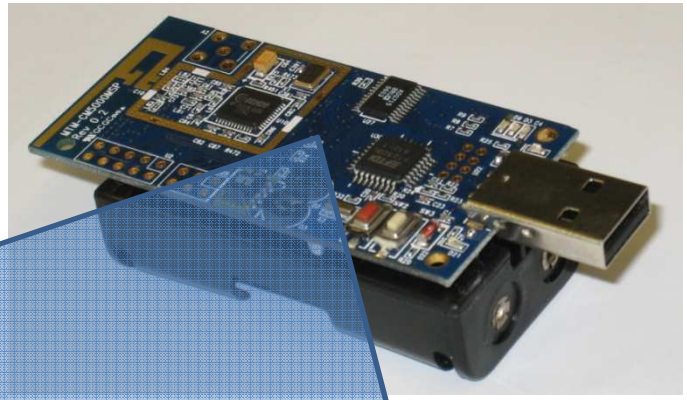
- A pure dynamic component-oriented operating system for WSNs (currently TelosB)
- Whole system architecture - 90% of ROM - modifiable at runtime as components
- All components strongly separated & compiled independently

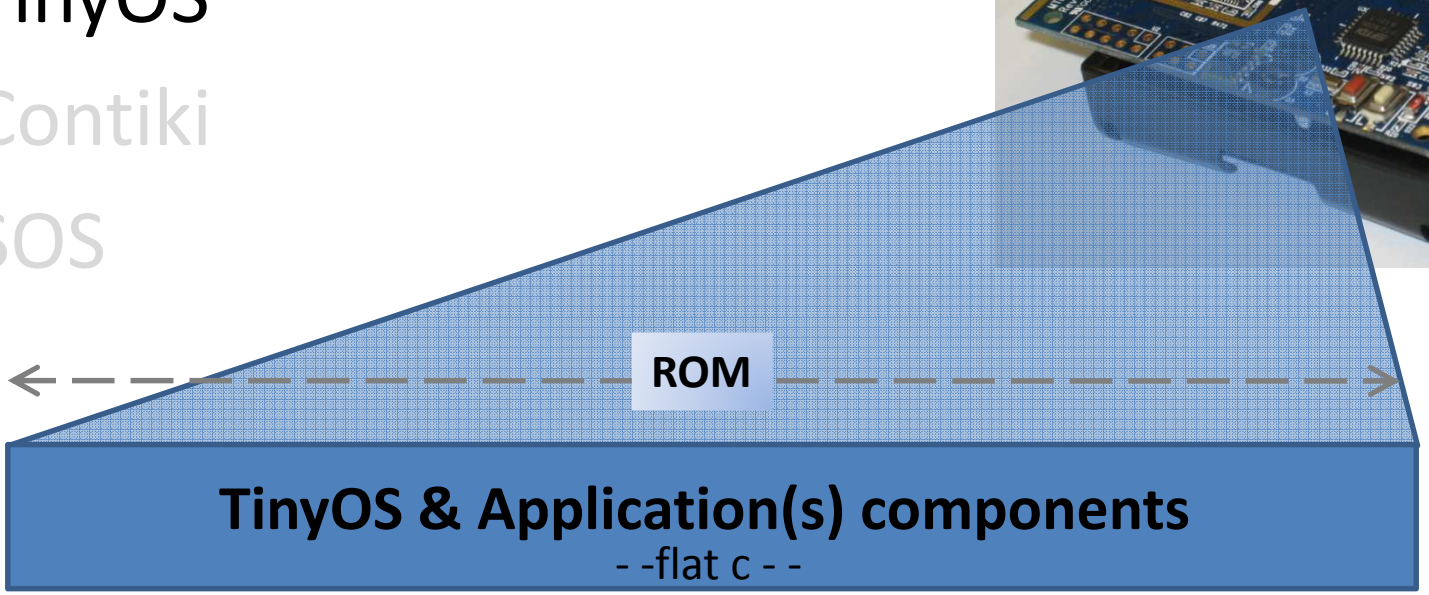
OS Survey

- TinyOS
- Contiki
- SOS

OS Survey

- TinyOS
- Contiki
- SOS

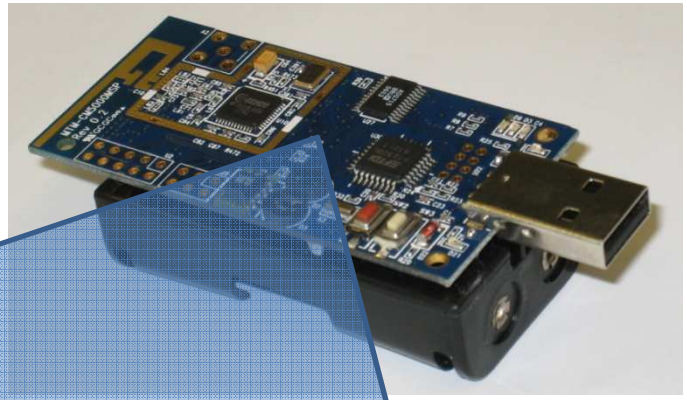


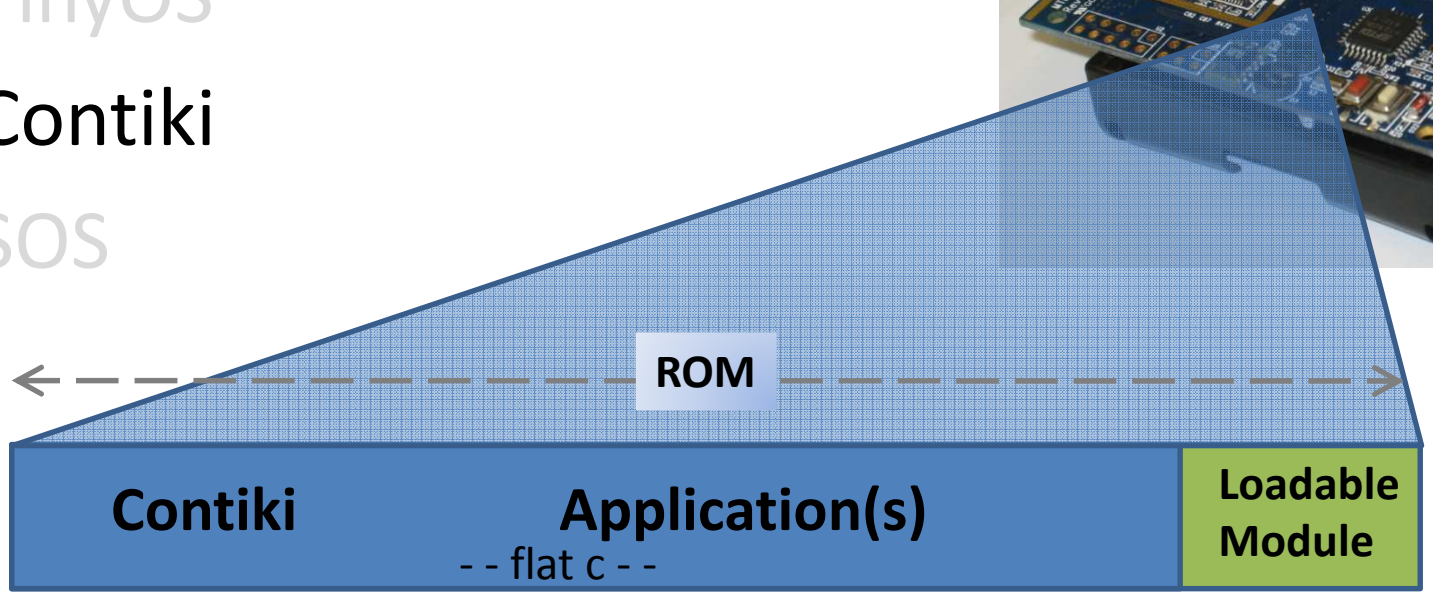


- Components are separate prior to compilation
- Post-compilation the whole system is a monolithic C program image
- This image is injected directly into the MCU's onboard flash from a PC
- Remote reprogramming by sending a new image or binary patch

OS Survey

- TinyOS
- Contiki
- SOS



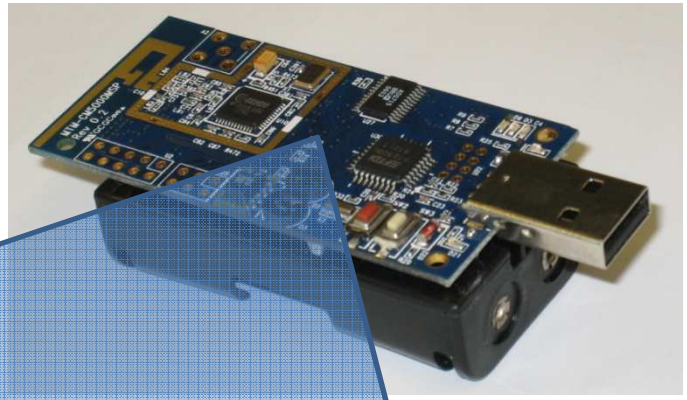


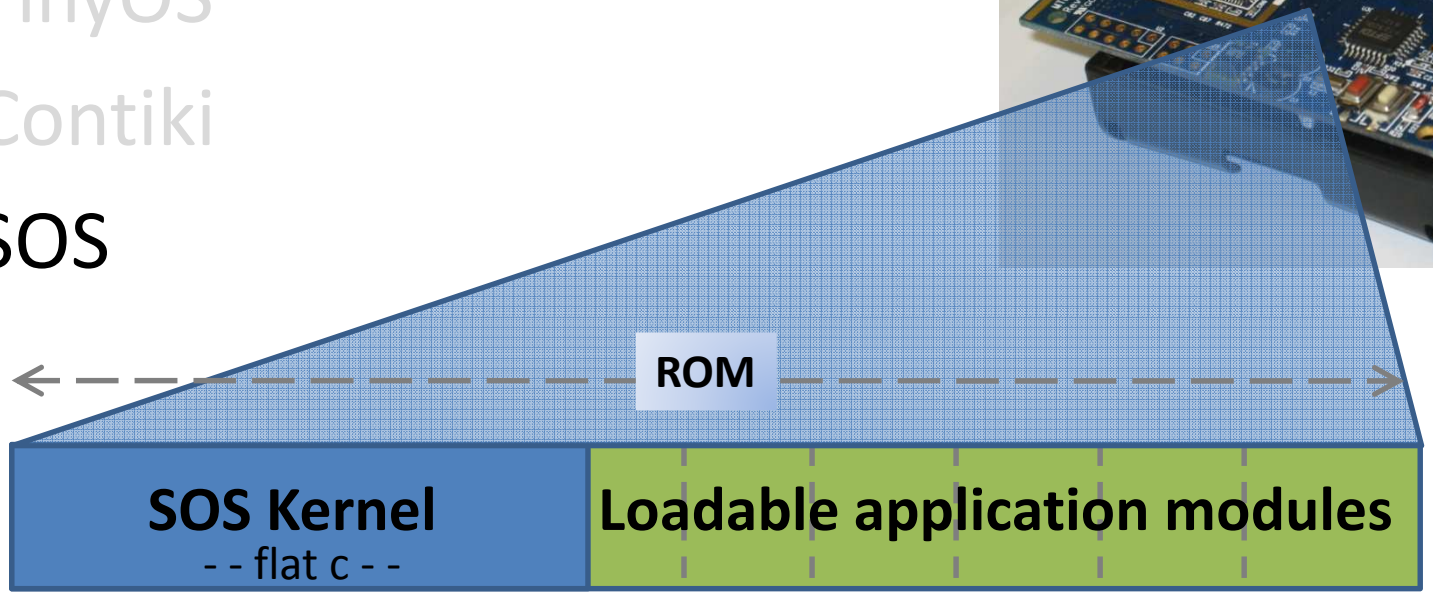
Contiki Module:

process_start()

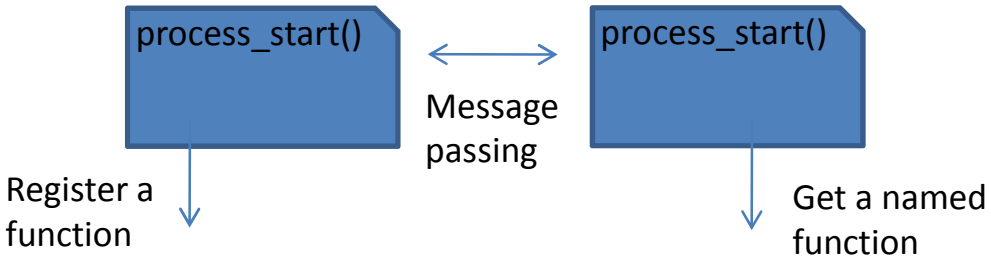
OS Survey

- TinyOS
- Contiki
- SOS



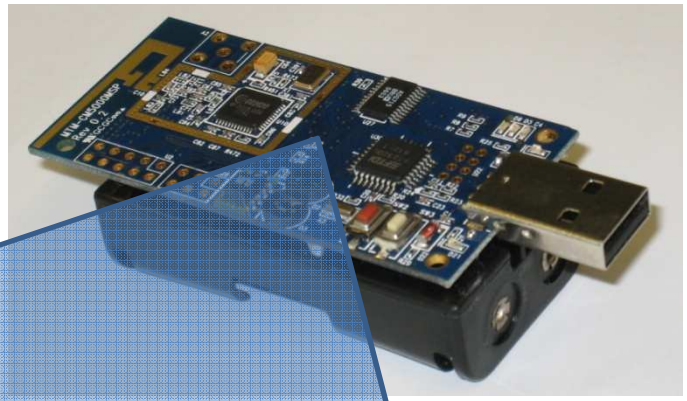


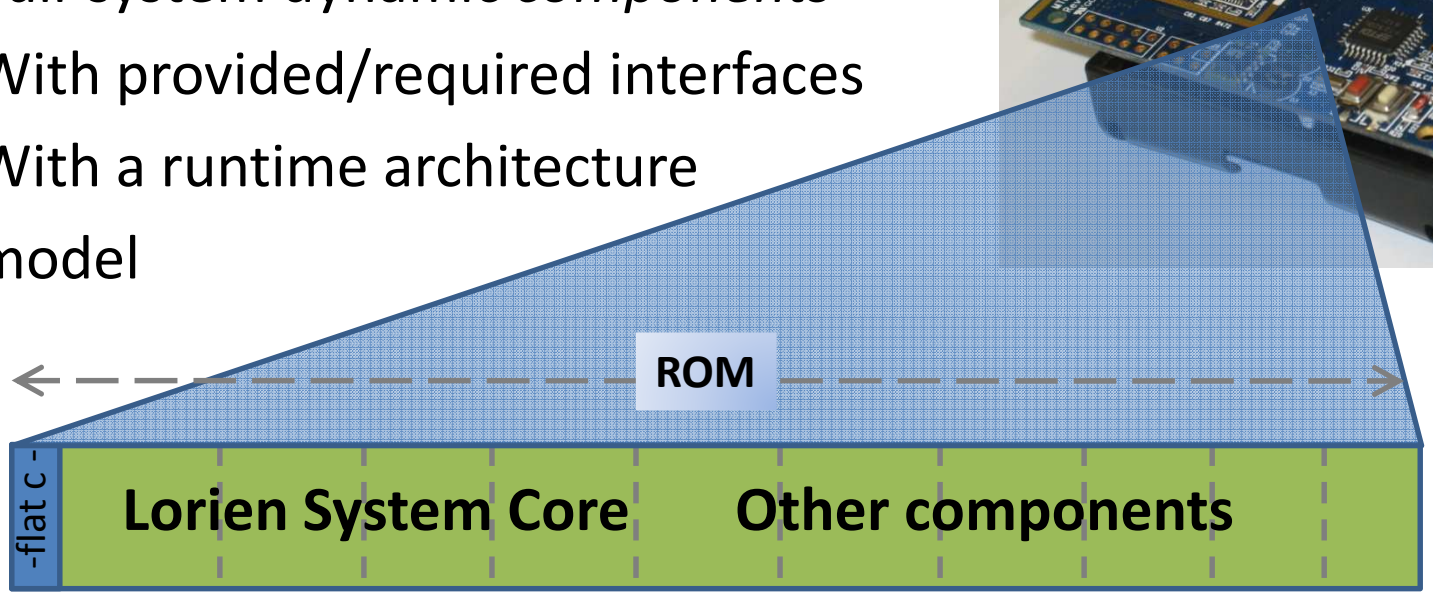
SOS Module:



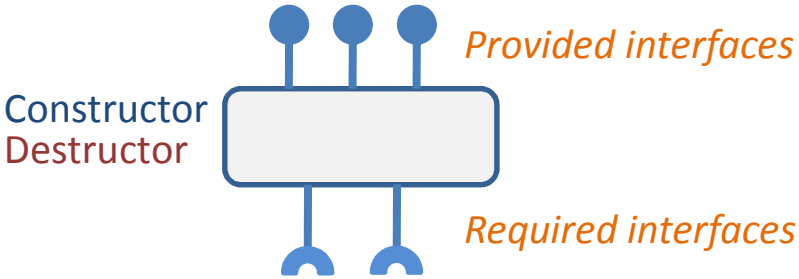
Lorien

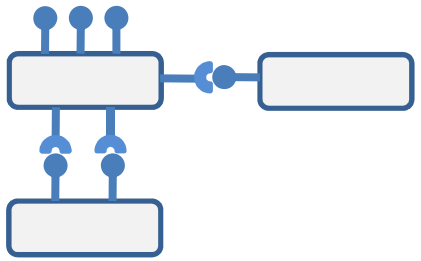
- Full-system dynamic *components*
- With provided/required interfaces
- With a runtime architecture model





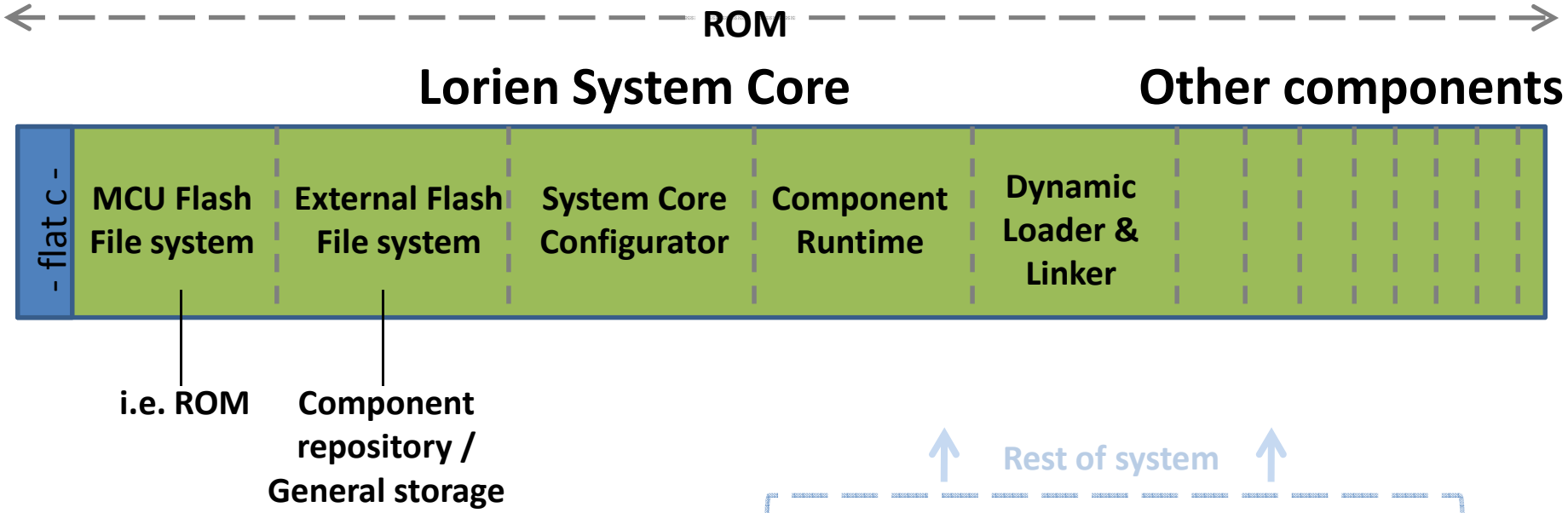
Lorien Component:

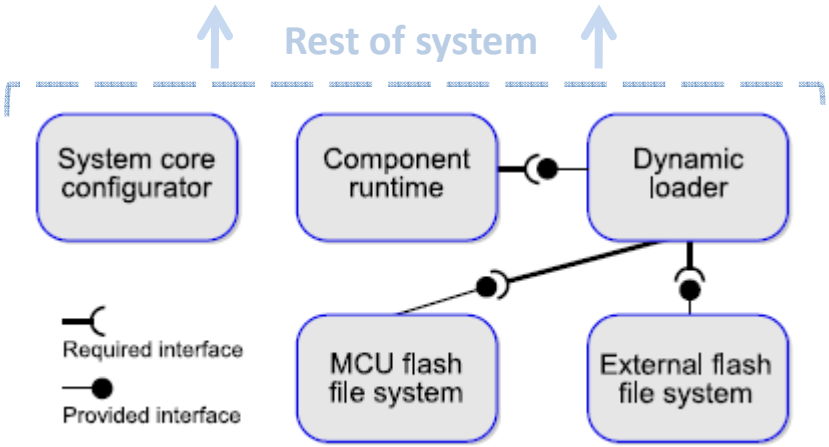




Lorien

- *The core: A componentised component loader*



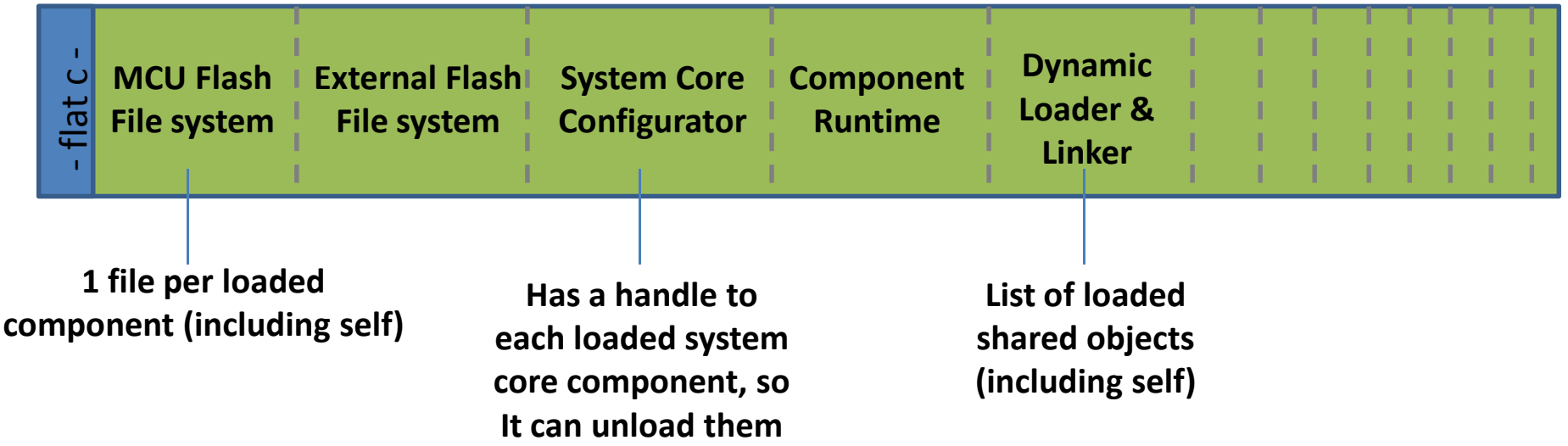


Lorien

- The key problem
  - The entire Lorien system is made of strongly separated, replaceable components with a wholly modifiable system architecture at runtime
  - We wanted a unified programming model across the entire system, so there is nothing special about the system core components – they can be loaded, instantiated, unloaded etc.
  - So who loads the loader?
    - The initial system image has to appear as if the system core loaded itself

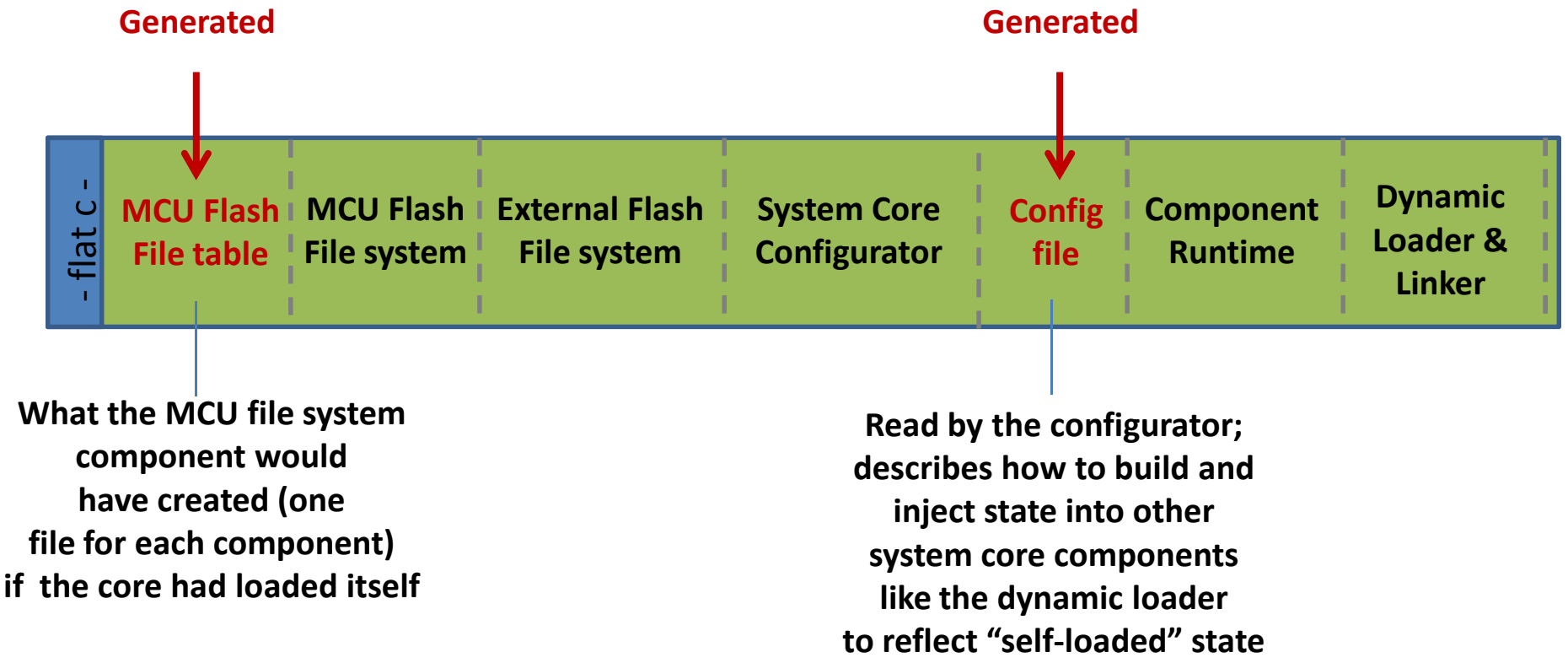
The Lorien Toolchain

- Creating a system snapshot in time
  - Or: pre-loading the loader
  - If the system core really had loaded itself, there would be certain state in each component reflecting this



The Lorien Toolchain

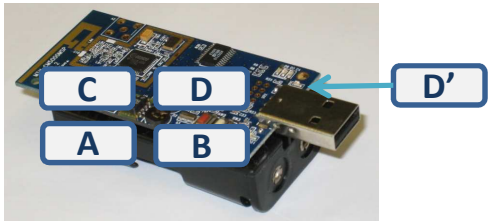
- Creating a system snapshot in time
  - Toolchain generates an image to flash to nodes...



Implications

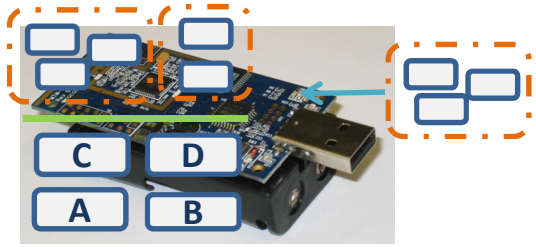
- What’s now possible with Lorien

Implications



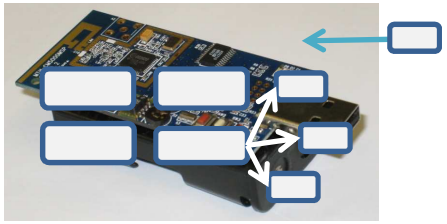
- Component-based update of code:
  - Send a new version of a component to your deployed system
  - Saves bandwidth & energy of sending an entire image
  - Saves effort on the node of applying the update (unload the old component, load the new one)

Implications



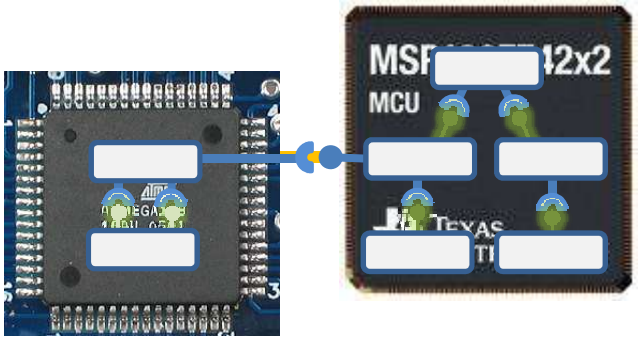
- Multi-user environment:
  - At Lancaster we have a WSN deployed around the building
  - With the Lorien system core running on each node, multiple researchers can develop applications and upload them to the WSN
    - Without disturbing already-deployed applications, or needing access to those applications' source code
  - A researcher can even develop a new file system implementation to try out and upload this to the network, still without disturbing – or needing access to the source code of – anything else

Implications



- Adaptive systems
  - Develop a system which can select from a pool of different components offering the same functionality based on environmental conditions
  - Upload new components to this system to test out the good operating conditions & ranges in which the system selects these components for use

Implications



- Multi-processing unit platforms
  - Emerging WSN hardware platforms use multiple programmable chips
    - Dedicated image processing chips
    - Programmable radio chips
  - Lorien can be distributed across multiple chips & components across those chips can interact in the normal way without needing to be aware of the different address spaces & communication buses involved
    - Components can also migrate between different chips at runtime, potentially shutting down chips at low load times
    - No single-image-based OS can do this

Future work

- Lots of different components to develop & evaluate
  - Concurrency models
  - MAC protocols
  - Routing algorithms (like Wiselib – can be made into a component)
  - etc...

Get involved

- http://opencomc.sourceforge.net/
- Lots of tutorials & quick guides