Contiki – a Lightweight and Flexible Operating System for Tiny Networked Sensors

Adam Dunkels, Björn Grönvall, Thiemo Voigt

Swedish Institute of Computer Science

IEEE EmNetS-I, 16 November 2004

OUTLINE

- Introduction
- System Overview
- Kernel architecture
- Services
- Communication support
- Discussion
- Conclusion

Introduction

- **Contiki** – an OS for sensor network nodes
- Ported Contiki to a number of platforms
  - MSP430, AVR, HC12, Z80, 6502, x86, ...
- Built a few applications for experimental network deployments

Introduction(cont.)

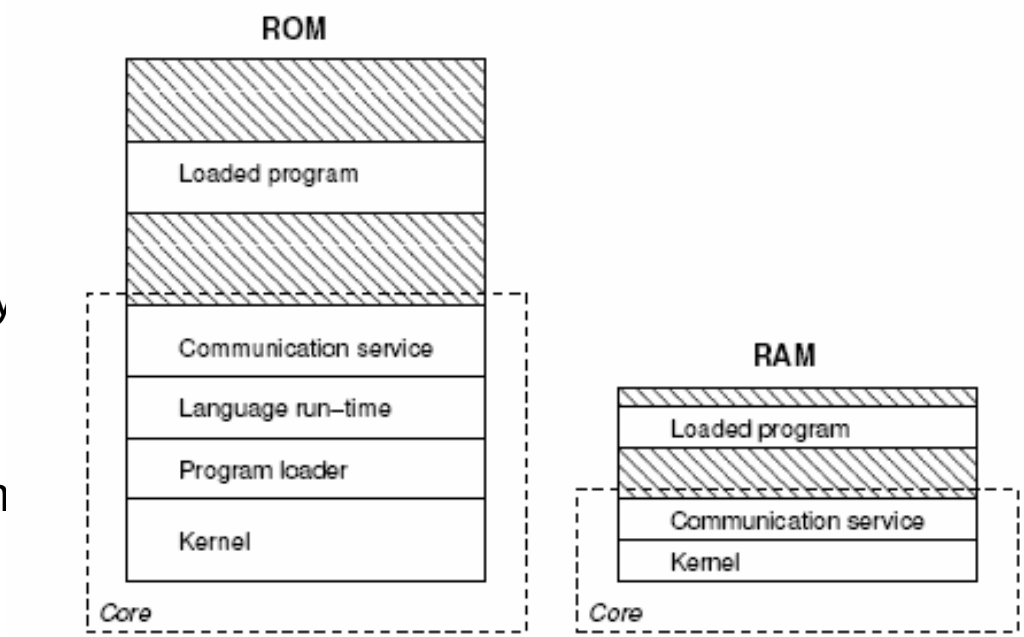
- Contribution
  - Downloading code at run time
    - Selective reprogramming
  - Portability
  - Event-driven systems
    - Event vs multi thread

System Overview

- Hardware target
  - “Mote”-class device
    - 10-100 kilobytes of code ROM
    - 1-10 kilobytes of RAM
    - Communication (radio)
  - ESB from FU Berlin
    - MSP430, 2k RAM, 60k ROM

System Overview(cont.)

- Contiki system consists of
  - kernel ,libraries ,progra m loader ,a set of process
- Services can be dynamically replaced at run time
- Interprocess communication is done by posting event
- All process share the same address space

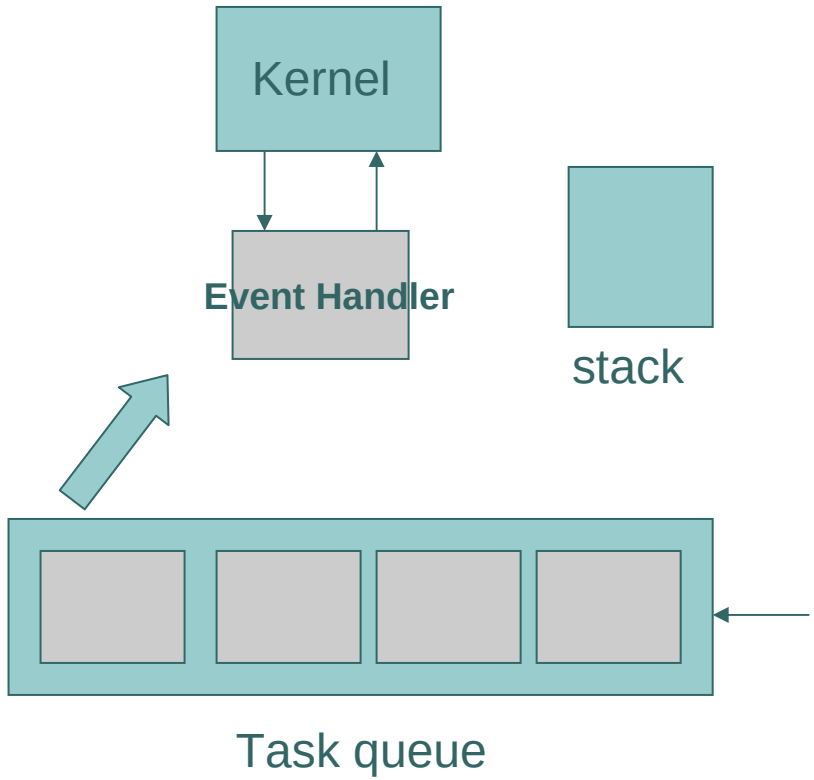


Kernel architecture

- Event-driven vs multi-threaded
- Event-driven (TinyOS)
  - low context switching overhead, fits well for reactive systems
  - Not suitable for e.g. long running computations
    - Public/private key cryptography
- Multi-threaded
  - Suitable for long running computations
  - Requires more resources(stack)
- Trade-offs: preemption, size

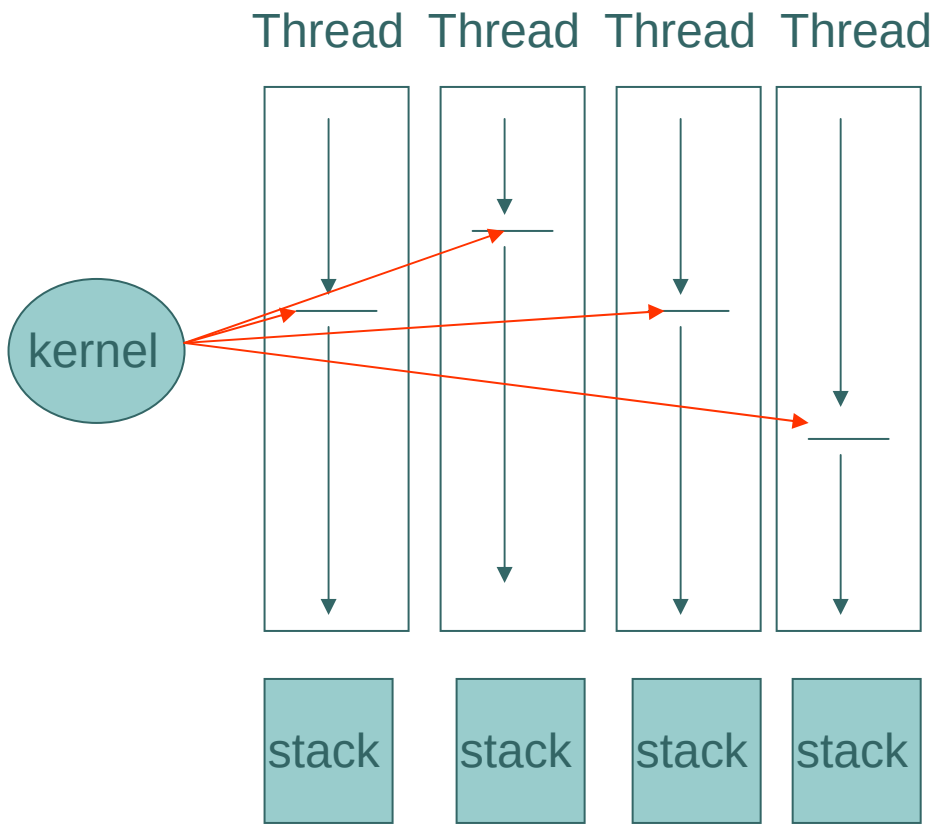
Kernel architecture(cont.)

- Event-driven(TinyOS)
  - Processes do not run without events
  - Event occurs: kernel invokes event handler
  - Event handler runs to completion (explicit `return;`)



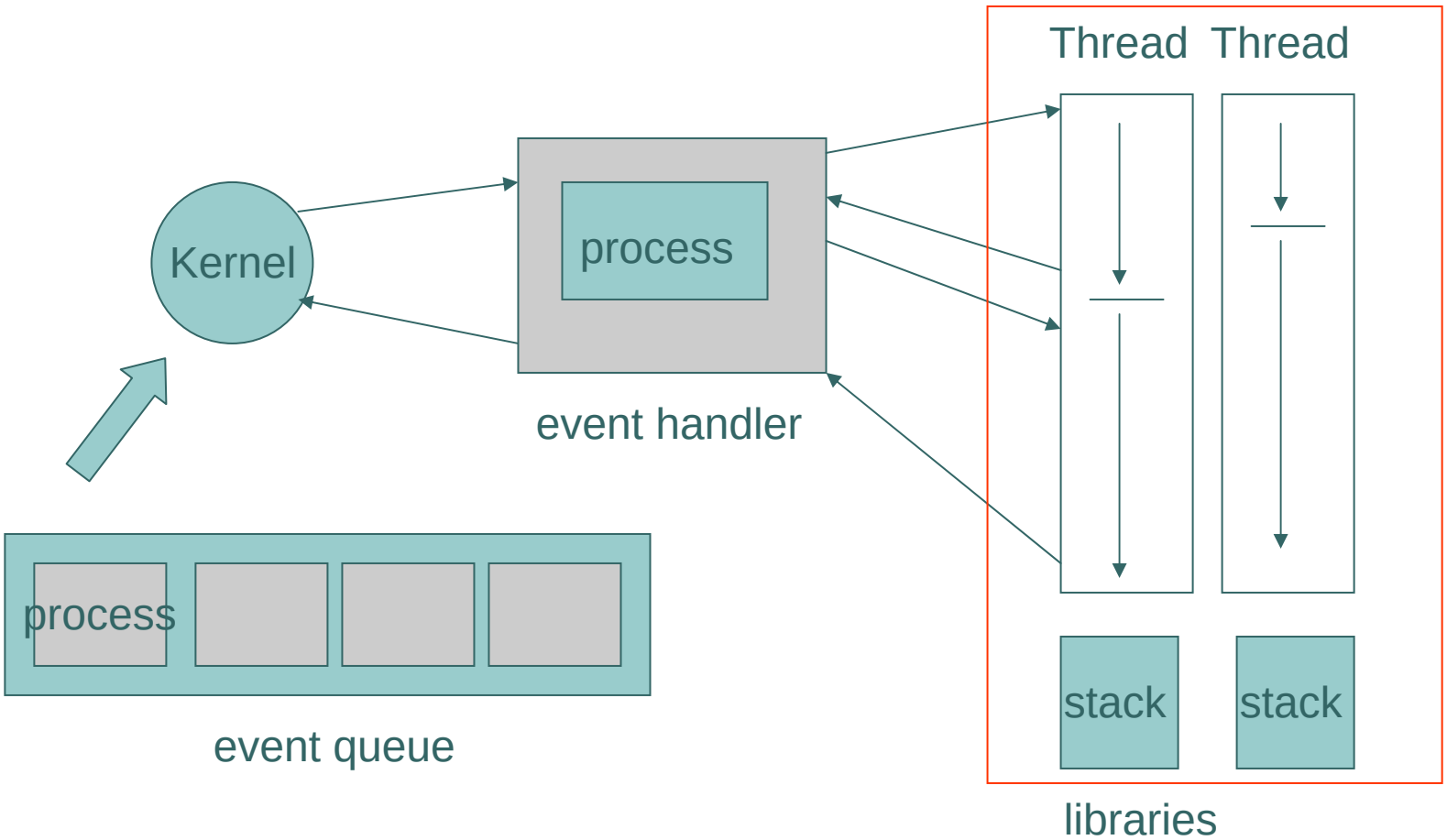
Kernel architecture(cont.)

- Multi-threaded
  - Preemption
  - Thread runs until next blocking statement
  - **Each thread requires its own stack**
    - **Larger memory usage**
  - Locking problems (race condition)



Kernel architecture(cont.)

- Combine **Event-driven** and **Multi-threaded**

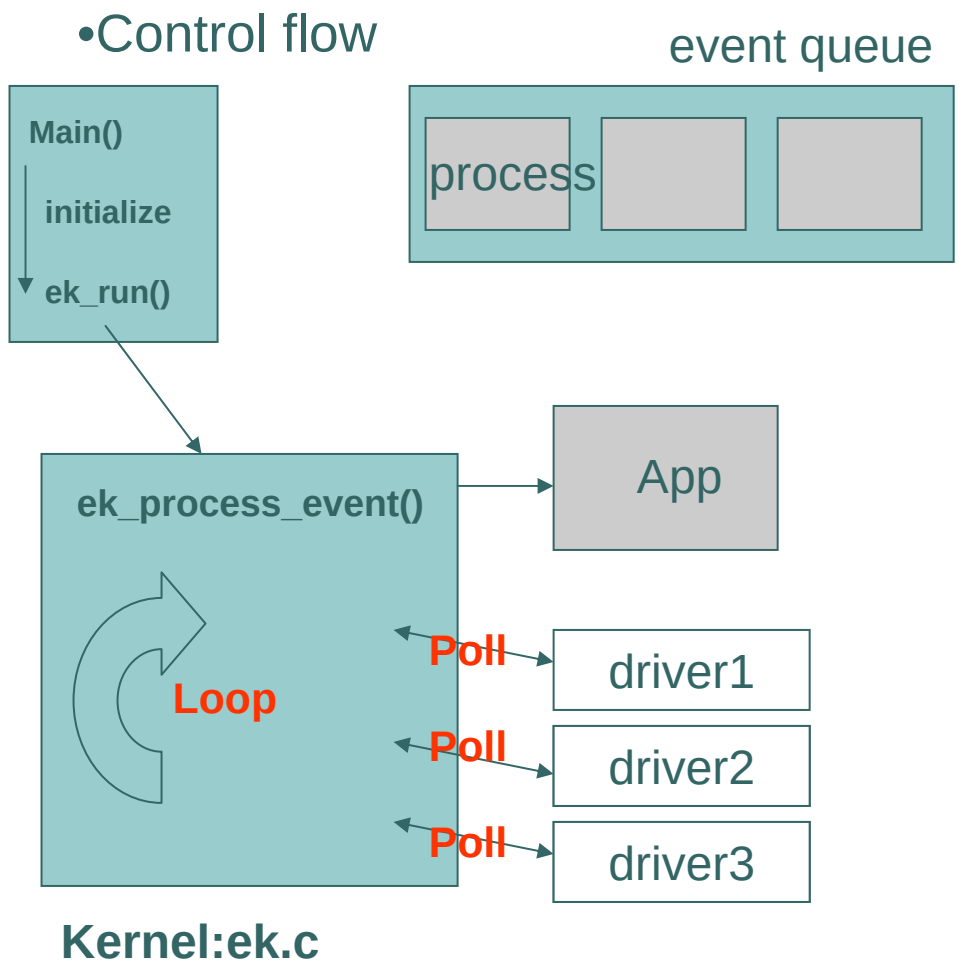


Kernel architecture(cont.)

- **Contiki: kernel** is **event-based**
  - Most programs run directly on top of the kernel
- **Multi-threading** implemented as a library
- Threads only used if **explicitly** needed
  - Long running computations, ...
- **Preemption** possible
  - Responsive system with running computations

Kernel architecture(cont.)

- Two level scheduling hierarchy
  - Event scheduler that dispatches event to running process
  - Periodically call processor's polling handler

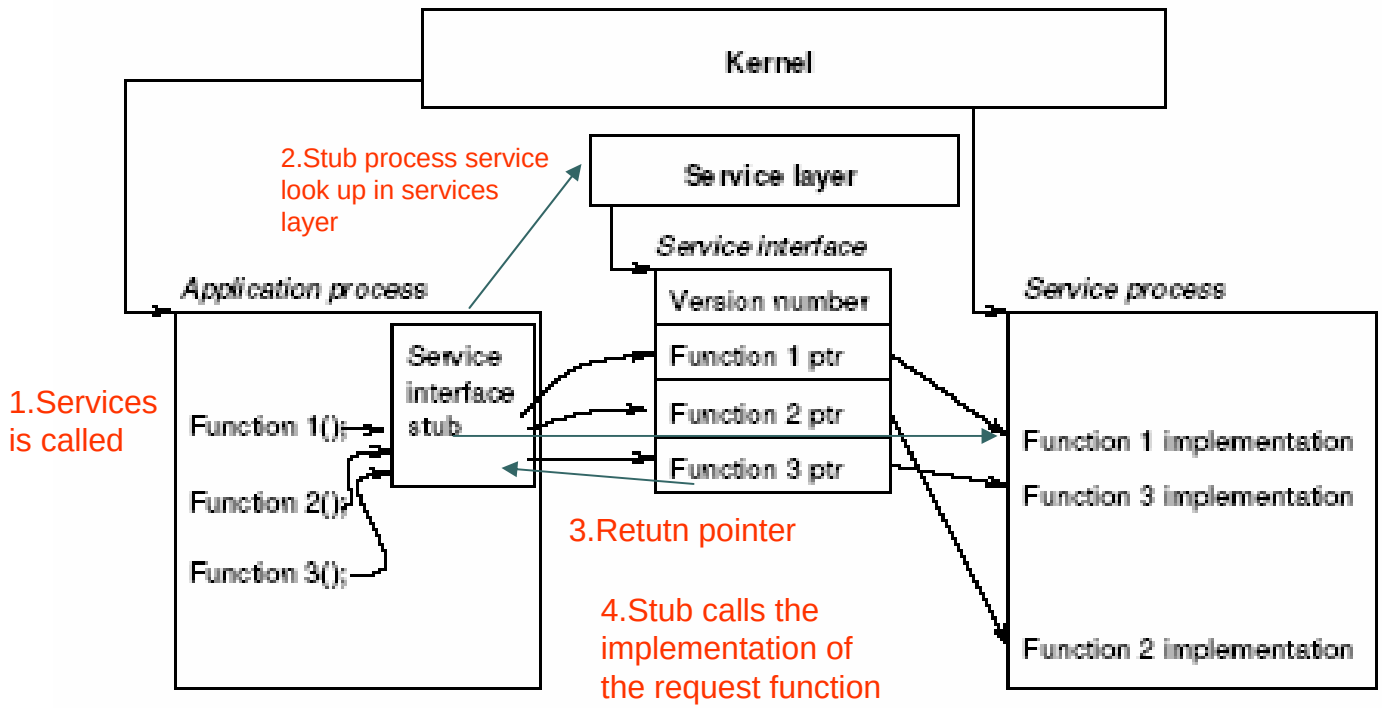


Kernel architecture(cont.)

- Loadable programs
  - Run-time relocation function and a binary format that contain relocation information
  - Loader check sufficient memory space
  - Loader call initialization function
- Power save mode

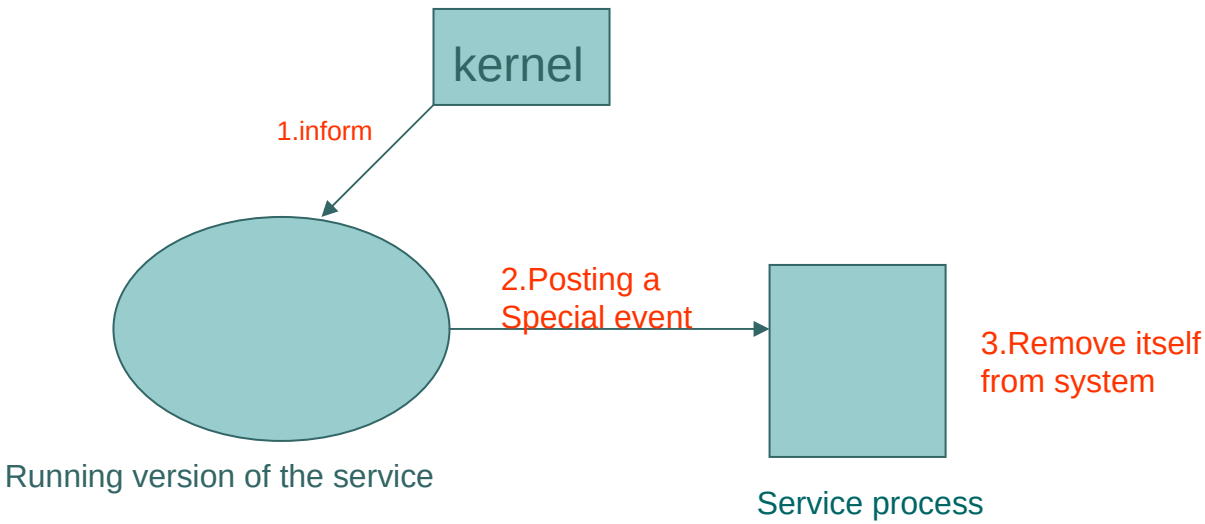
Services

- **An application function calling a service**



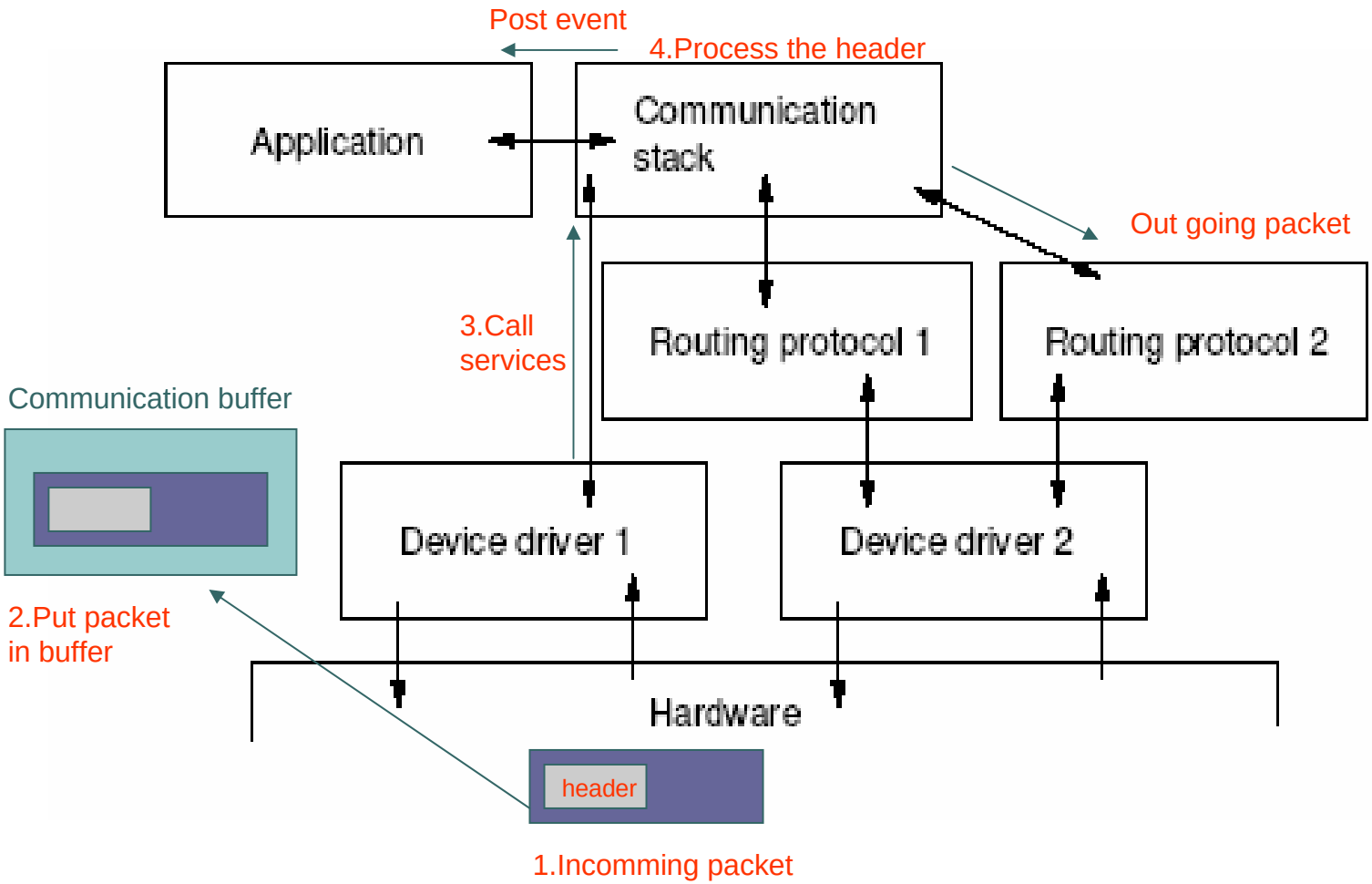
Services(cont.)

- Service replacement
  - Kernel provides a special mechanism for replacing a process and retaining the process ID



Communication support

- **Loosely coupled communication stack**



Discussion

- Reprogramming sensor nodes
  - 40 nodes dynamic distributed alarm system
  - Manual wired reprogramming complete system image
    - One node >> 30sec
    - 40 node >> 30 min
  - Over the air reprogramming a single component of application
    - 2 min

Discussion(cont.)

- Program typically much smaller than entire system image (1-10%)
  - Much quicker to transfer over the radio

Discussion(cont.)

- Code size
  - TinyOs < Contiki <Mantis

Module	Code size (AVR)	Code size (MSP430)	RAM usage
Kernel	1044	810	$10 + + 4e + 2p$
Service layer	128	110	0
Program loader	-	658	8
Multi-threading	678	582	$8 + s$
Timer library	90	60	0
Replicator stub	182	98	4
Replicator	1752	1558	200
Total	3874	3876	$230 + 4e + + 2p + s$

- **Size of the compiled code, in bytes**

Conclusion

- Contiki – OS for “mote”-class sensor nodes
- Contiki explores trade-offs in
  - static vs dynamic
  - event-driven vs multi-threaded
- Loadable programs, works well
  - Static linking can be a problem
- Threads on an event-driven kernel
  - Multi-threading suitable for certain applications