Contiki OS

Kaushal K | Cdac, Pune | Jun 2022

Agenda

- Features
- Network Stack
- Directory Structure
- Cooja Simulator
- Hands On Excersice

Introduction

- Contiki-NG (Next Generation) is an open source, cross-platform operating system for severely con-strained wireless embedded devices
- It focuses on dependable (reliable and secure) low-powercommunications and standardized protocols, such as 6LoWPAN, IPv6, 6TiSCH, RPL, and CoAP

Features

- Use of the standard C programming language
- Event-based kernel. In conjunction with tickles, platform-specific main loop code (implemented for some platforms)it allowed for fast reaction time to external events and energy-efficient execution
- Cooperative multi-threading-based process API
- Early, standards-compliant support for network protocols such as the Internet Protocol (IP) and IPv6 protocols through the uIP stack

Hardware Support

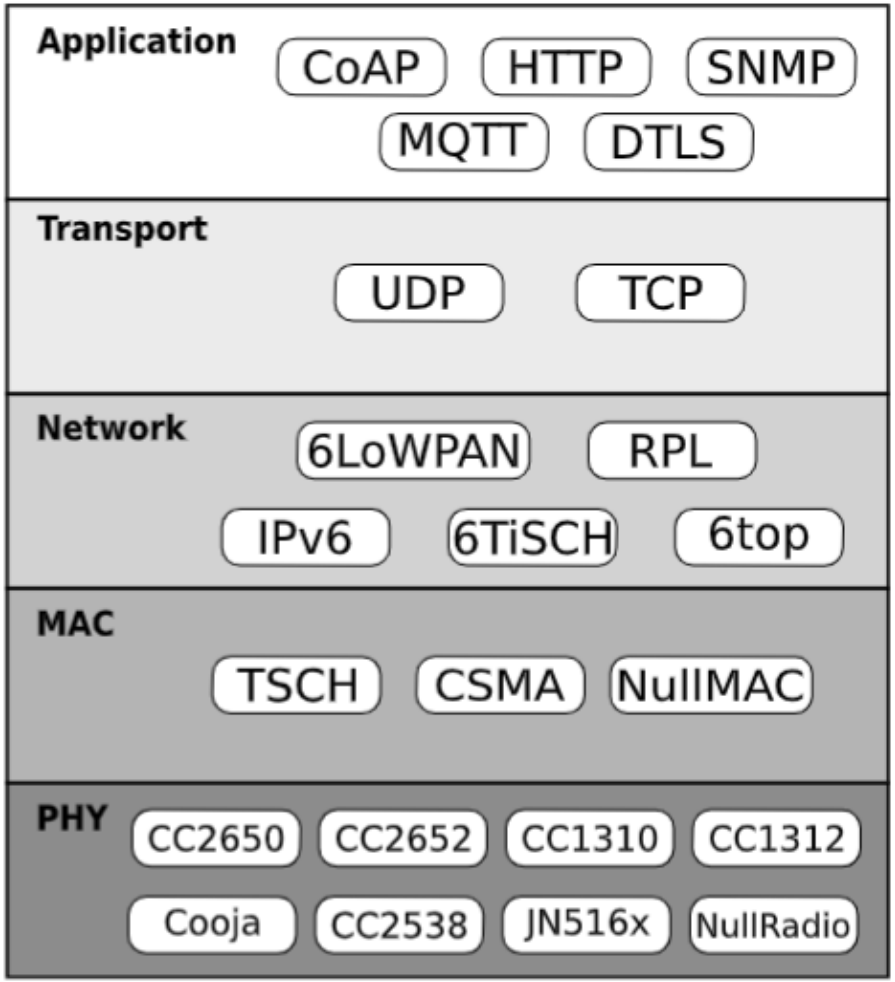
- Contiki-NG primarily targets Arm Cortex-M platforms.
- The official repository includes support for hardware by Nordic Semiconductor, NXP, OpenMote, Texas Instruments, and Zolertia.
- All those platforms are powered by Cortex-M3 or -M4 chips

Other Embedded OS

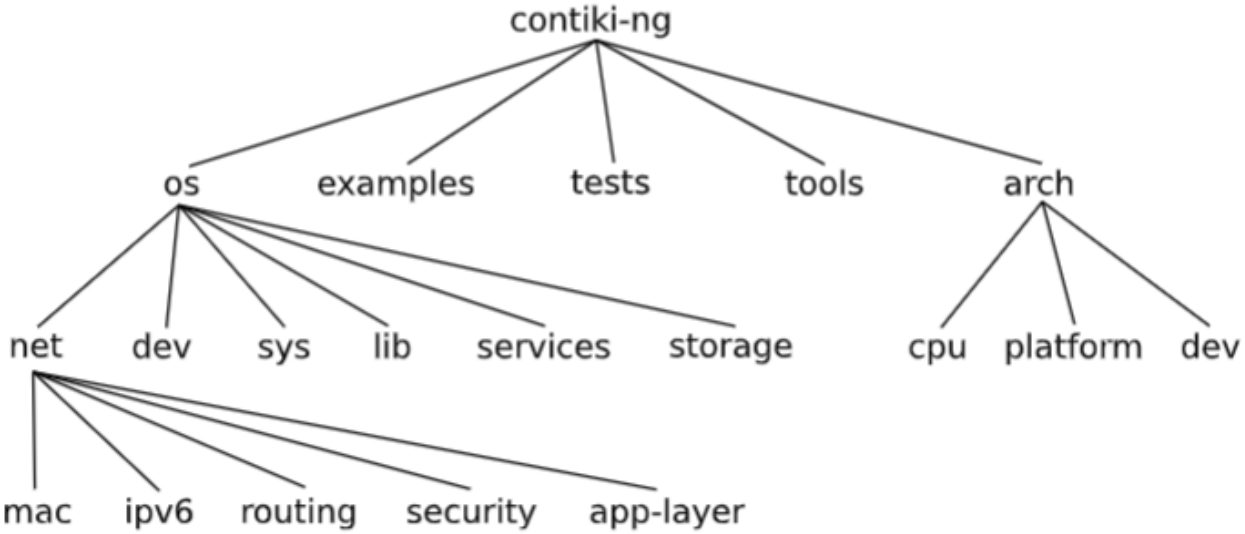
Feature overview of embedded OSs.

Project	Networking	Licence	Language	Threading
Contiki-NG	TSCH, 6LoWPAN, RPL	BSD	C, C++	Cooperative
Contiki	TSCH, 6LoWPAN, RPL	BSD	C	Cooperative
Apache Mynewt	BLE, LoRa, TCP/IP	Apache 2.0	C, C++	Preemptive
Arm Mbed	BLE, LoRa, lwIP	Apache 2.0	C++	Preemptive
FreeRTOS	TCP/IP	MIT	C, C++	Preemptive
RIOT	6LoWPAN, BLE	GNU LGPL	C, C++	Preemptive
TinyOS	6LoWPAN	BSD	nesC	Optional preemptive
Zephyr	BLE, Thread, 6LoWPAN	Apache 2.0	C, C++	Optional preemptive

Network Stack



Directory Structure of Contiki



Directory Structure Description

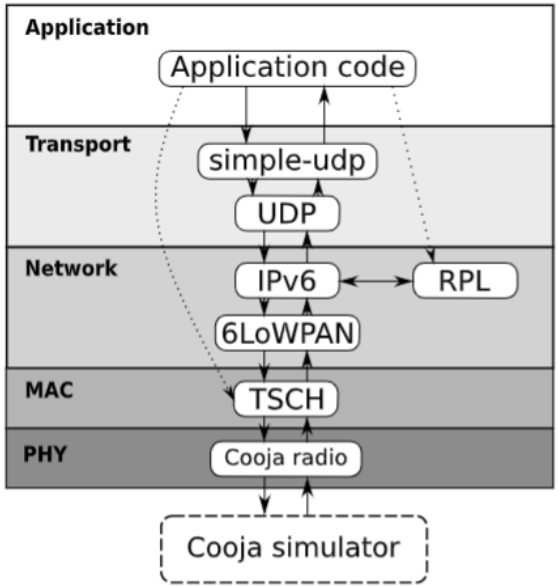
- os
  - Contains the actual Contiki-NG code. Includes the systems primitives such as processes and timers, the networking stack, and all libraries and services. All examples also compile and link to it
- Arch
  - Contains all hardware-dependent code. This includes CPU, device and platform drivers. A list of supported platforms can be found under arch/platforms and its sub-directories. This is where to put your code if you are porting Contiki-NG to your own platform.

Directory Structure

- Examples
  - Contains ready-to-use example projects. Shows how to use networking, libraries and storage services. Includes a RPL border router, the slip-radio interface etc.
- Tools
  - Contains tools, which are not to be included in a Contiki-NG firmware, but rather intended to run on a computer. Includes flashing tools, the Cooja simulator (as a submodule), Docker and Vagrant scripts, and more

Cooja Simulator

- It is a simulator for IEEE 802.15.4 networks of devices running Contiki-based firmware.



Set up Contiki

- Install Docker on Ubuntu
  - sudo apt-get install docker-ce
- sudo usermod -aG docker <your-user>
- docker pull contiker/contiki-ng
- Execute commands
  - git clone https://github.com/contiki-ng/contiki-ng.git
  - cd contiki-ng
  - git submodule update --init –recursive
- Add the alias.txt content in ~/.bashrc
- Finally execute
  - contiker

Build System

- Each project must have a Makefile at its root, use ‘make’ command to build the project
- By default, TARGET=native, use command ‘make targets’ to see list
- In that Target you can set the Board
- The build system will create firmware files in current dir
- It will also create build/$TARGET
- Choose deployment type as deploy/testing/debug using BUILD_DIR_CONFIG
- Use clean and distclean to clear project builds

More Build Options

- viewconf
- Targets
- %.ramprof
- usage

Multitasking and Scheduling

- A process will be idle until it receive an event
- Performs the chunk of work and then suspends its own work until next event
- Cooperative scheduler never forces a context switch between running process
- Hence long processes should be split in other process

Processes

- Contiki processes are built on top of a lightweight threading library – protothreads
- The process() method uses two args: Process variable and Process name

```
#include "contiki.h" /* Main include file for OS-specific modules. */
#include <stdio.h> /* For printf. */

PROCESS(test_proc, "Test process");
AUTOSTART_PROCESSES(&test_proc);
```

Process Running

```
PROCESS_THREAD(test_proc, ev, data)
{
  PROCESS_BEGIN();

  printf("Hello, world!\n");

  PROCESS_END();
}
```

Events

- Asynchronous
  - Events are put in queue and dispatched in round-robin sequence. It can a process specific or broadcast
  - Polling mechanism as well. It is a high priority event for specific task
- Synchronous
  - Causes the receiving process to get scheduled immediately. No broadcasting

Interrupts

- The cooperative nature makes sure the scheduler will not interrupt other process, until it is coming from a Hardware interrupts
- The code execution based on context can be categorized
  - Code running in Interrupts
  - Code running outside Interrupts Service Routine
- When same resource is getting used by ISR and main thread program then problem arises, so fundamental synchronization like Mutexes and Critical Sections are used

Timers

- To check if a time period has passed, waking up the system from low power mode at schedule times
- Build on ‘clock’ module, with sub as
  - Timer
    - A simple timer, without built-in notification (caller must check if expired). Safe from interrupt.
  - Etimer
    - Schedules events to Contiki-NG processes. Unsafe from interrupt.
  - Ctimer
    - Schedules calls to a callback function. Unsafe from interrupt.
  - Rtimer
    - real-time task scheduling, with execution from ISR. Safe from interrupt.

File Systems

- Coffee
  - It is a minimalistic, yet fully functional file system that operates with the peculiar characteristics of flash memories and EEPROM

```
int fd;
char buf[] = "Hello, World!";

fd = cfs_open("test", CFS_READ | CFS_WRITE);
if(fd >= 0) {
  cfs_write(fd, buf, sizeof(buf));
  cfs_seek(fd, 0, CFS_SEEK_SET);
  cfs_read(fd, buf, sizeof(buf));
  printf("Read message: %s\n", buf);
  cfs_close(fd);
}
```

File Systems

- Antelope
  - is a lightweight, relational database management system for resource-constrained IoT devices. It has been designed to run on top of a file system such as Coffee.
  - Antelope Query Language (AQL)

```
CREATE RELATION faithful;
CREATE ATTRIBUTE eruption DOMAIN LONG IN faithful;
CREATE ATTRIBUTE recharge DOMAIN LONG IN faithful;
```

```
SELECT recharge, eruption FROM faithful WHERE recharge > 5000 AND eruption >= 60000 AND eruption < 90000;
```

Thank You