Real Time Operating System for IoT

Internet of Things

Rahul Shandilya

IoT OS or RTOS

An Internet of Things Operating System is an operating system that is designed to perform within the constraints that are particular to Internet of Things devices, including restrictions on memory, size, power and processing capacity. IoT OSes control systems in cars, traffic and streets lights, Smart TVs, ATMs, airplane controls, point of sale (POS) terminals, digital cameras GPS navigation systems, elevators, digital media receivers and smart meters among many other possibilities. IoT Demands:

- Low Power, Low Cost and Low memory footprint (RAM and ROM)
- Should have provision for IPv6, with 6LoWPAN Adaptation Layer
- Separate Routing Protocol for Low Power and Lousy networks
- New light Weight Application Layer Protocol unlike http but should have a support for http also.

Parameters for suitable IoT OS

- **Footprint**: Since devices are constraint, we expect OS to have low memory, power and processing requirements. The overhead due to the OS should be minimal.
- **Portability**: OS isolates applications from the specifics of the hardware. Usually, OS is ported to different hardware platforms and interfaces to the board support package (BSP) in a standard way, such as using POSIX calls.
- **Modularity**: OS has a kernel core that's mandatory. All other functionality can be included as add-ons if so required by the application.
- **Connectivity**: OS supports different connectivity protocols, such as Ethernet, Wi-Fi, BLE, IEEE 802.15.4, and more.
- **Scalability**: OS must be scalable for any type of device. This means developers and integrators need to be familiar with only one OS for both nodes and gateways.
- **Reliability**: This is essential for mission-critical systems. Often devices are at remote locations and have to work for years without failure. Reliability also implies OS should fulfil certifications for certain applications.
- **Security**: OS has add-ons that bring security to the device by way of secure boot, SSL support, components and drivers for encryption.

Important opensource RTOS

- **Contiki OS**: Contiki is an open source operating system for the Internet of Things. Contiki connects tiny low-cost, low-power microcontrollers to the Internet. Contiki is a powerful toolbox for building complex wireless systems.
- **RIoT OS**:RIOT is a free, open source operating system developed by a grassroots community gathering companies, academia, and hobbyists, distributed all around the world.
- Tiny OS: TinyOS is an embedded, component-based operating system and platform for low-power wireless devices, such as those used in wireless sensor networks (WSNs), smartdust, ubiquitous computing, personal area networks, building automation, and smart meters.
- **Lite OS**: Huawei LiteOS is an IoT-oriented software platform integrating an IoT operating system and middleware. It is an open source operating system for IoT smart terminals.

Comparision

Contiki

Contiki is an open source operating system for the Internet of Things. Contiki connects tiny low-cost, low-power microcontrollers to the Internet. Contiki is a powerful toolbox for building complex wireless systems. Contiki was created by Adam Dunkels in 2002 and has been further developed by a worldwide team of developers from Texas Instruments, Atmel, Cisco and many more.

- A typical Contiki system has memory of the order of kilobytes, a power budget of the order of milliwatts, processing speed measured in megahertz, and communication bandwidth of the order of hundreds of kilobits/second.
- Contiki supports fully standard IPv6 and IPv4, along with the recent low-power wireless standards: 6lowpan, RPL, CoAP.
- Contiki provides multitasking and a built-in Internet Protocol Suite (TCP/IP stack), yet needs only about 10 kilobytes of random-access memory (RAM) and 30 kilobytes of read-only memory (ROM). A full system, including a graphical user interface, needs about 30 kilobytes of RAM.
- Contiki applications are written in standard C, with the Cooja simulator Contiki networks can be emulated before burned into hardware, and Instant Contiki provides an entire development environment in a single download.

Networking

Contiki provides three network mechanisms:

- the uIP TCP/IP stack, which provides IPv4 networking;
- the uIPv6 stack, which provides IPv6 networking;
- the Rime stack, which is a set of custom lightweight networking protocols designed specifically for low-power wireless networks.

Programming Model

To run efficiently on memory-constrained systems, the Contiki programming model is based on protothreads. A protothread is a memory-efficient programming abstraction that shares features of both multi-threading and event-driven programming to attain a low memory overhead. The kernel invokes the protothread of a process in response to an internal or external event.

Simulation

The Contiki system includes a network simulator called Cooja. Cooja simulates networks of Contiki nodes. Cooja can be very useful because of its emulative functions, which help developers in testing applications. This speeds up the development process: without a simulator the developer would have to upload and test every new version of firmware on real hardware.

RIoT OS

RIOT12 is an open-source microkernel operating system for the IoT, licensed as LGPL. It allows C and C++ application programming, and provides both full multi-threading and real-time capabilities.

- RIOT runs on 8-bit (e.g., AVR Atmega), 16-bit (e.g., TI MSP430) and 32-bit hardware (e.g., ARM Cortex).
- A native port also enables RIOT to run as a Linux or MacOS process, enabling the use of standard development and debugging tools such as GNU Compiler Collection, GNU Debugger, Valgrind, Wireshark, an so on.
- RIOT is partly POSIX-compliant and provides multiple network stacks, including IPv6, 6LoWPAN and standard protocols such as RPL, UDP, TCP, and CoAP

TinyOS

TinyOS9 is a free, open-source, BSD-licensed OS designed for lowpower embedded distributed wireless devices used in sensor networks. TinyOS was developed by University of California, Berkeley, Intel Research, and Crossbow Technology

- It has designed to support the intensive concurrent operations required by networked sensors, with minimal hardware requirements.
- It is written in the nesC (Network Embedded Systems C) programming language, which is a version of C optimized to support components and concurrency.
- It is also component-based, supporting event-driven programming of applications for TinyOS.