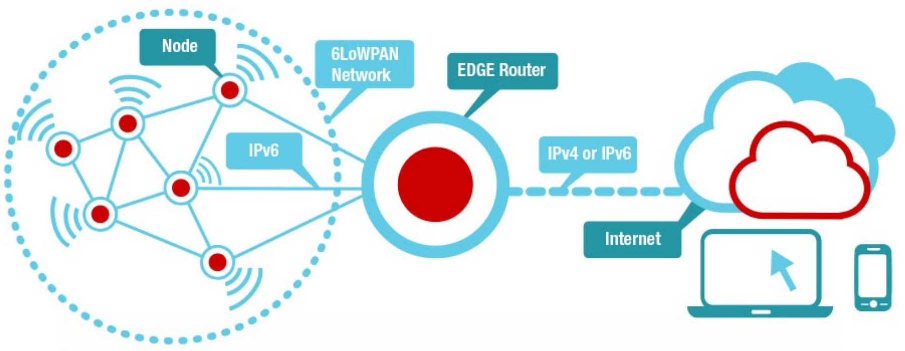

Contiki

The Open Source OS for the Internet of Things

It owes its popularity to being very lightweight (by modern standards), mature, and flexible.

CONTIKI-OPERATING SYSTEM

Contiki is an operating system for networked, memory-constrained systems with a focus on low-power wireless Internet of Things devices. Extant uses for Contiki include systems for street lighting, sound monitoring for smart cities, radiation monitoring, and alarms.[1] It is open-source software released under a BSD license.

Contiki was created by Adam Dunkels in 2002[2] and has been further developed by a worldwide team of developers from Texas Instruments, Atmel, Cisco, ENEA, ETH Zurich, Redwire, RWTH Aachen University, Oxford University, SAP, Sensinode, Swedish Institute of Computer Science, ST Microelectronics, Zolertia, and many others.[3] Contiki gained popularity because of its built in TCP/IP stack and lightweight preemptive scheduling over event-driven kernel [4] which is a very motivating feature for IoT. The name Contiki comes from Thor Heyerdahl's famous Kon-Tiki raft.

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).[1] A full system, including a graphical user interface, needs about 30 kilobytes of RAM.[5]

Hardware

Contiki is designed to run on types of hardware devices that are severely constrained in memory, power, processing power, and communication bandwidth. A typical Contiki system has memory on the order of kilobytes, a power budget on the order of milliwatts, processing speed measured in megaHertz, and communication bandwidth on the order of hundreds of kilobits/second. Such systems include many types of embedded systems, and old 8-bit computers.

Networking

Contiki provides three network mechanisms: the uIP TCP/IP stack,[6] which provides IPv4 networking, the uIPv6 stack,[7] which provides IPv6 networking, and the Rime stack, which is a set of custom lightweight networking protocols designed for low-power wireless networks. The IPv6 stack was contributed by Cisco and was, when released, the smallest IPv6 stack to receive the IPv6 Ready certification.[8] The IPv6 stack also contains the Routing Protocol for Low power and Lossy Networks (RPL) routing protocol for low-power lossy IPv6 networks and the 6LoWPAN header compression and adaptation layer for IEEE 802.15.4 links.

Rime is an alternative network stack, for use when the overhead of the IPv4 or IPv6 stacks is prohibitive. The Rime stack provides a set of communication primitives for low-power wireless systems. The default primitives are single-hop unicast, single-hop broadcast, multi-hop unicast, network flooding, and address-free data collection. The primitives can be used on their own or combined to form more complex protocols and mechanisms.[9]

Low-power operation

Many Contiki systems are severely power-constrained. Battery operated wireless sensors may need to provide years of unattended operation and with little means to recharge or replace batteries. Contiki provides a set of mechanisms to reduce the power consumption of systems on which it runs. The default mechanism for attaining low-power operation of the radio is called ContikiMAC.[10] With ContikiMAC, nodes can be running in low-power mode and still be able to receive and relay radio messages.

Simulation

The Contiki system includes a network simulator called Cooja, which simulates networks of Contiki nodes.[11] The nodes may belong to either of three classes: emulated nodes, where the entire hardware of each node is emulated, Cooja nodes, where the Contiki code for the node is compiled for and executed on the simulation host, or Java nodes, where the behavior of the node must be reimplemented as a

Java class. One Cooja simulation may contain a mix of nodes from any of the three classes. Emulated nodes can also be used to include non-Contiki nodes in a simulated network.

In Contiki 2.6, platforms with the TI MSP430 and Atmel AVR microcontrollers can be emulated.

Programming model

To run efficiently on small-memory systems, the Contiki programming model is based on protothreads.[12][13] A protothread is a memory-efficient programming abstraction that shares features of both multithreading and event-driven programming to attain a low memory overhead of each protothread. The kernel invokes the protothread of a process in response to an internal or external event. Examples of internal events are timers that fire or messages being posted from other processes. Examples of external events are sensors that trigger or incoming packets from a radio neighbor.

Protothreads are cooperatively scheduled. Thus, a Contiki process must always explicitly yield control back to the kernel at regular intervals. Contiki processes may use a special protothread construct to block waiting for events while yielding control to the kernel between each event invocation.

Features

Screenshot of the VNC server running on the Atmel AVR port of Contiki
Contiki supports per-process optional preemptive multithreading, inter-process communication using message passing through events, as well as an optional graphical user interface (GUI) subsystem with either direct graphic support for locally connected terminals or networked virtual display with Virtual Network Computing (VNC) or over Telnet.

A full installation of Contiki includes the following features:

Multitasking kernel
Optional per-application preemptive multithreading
Protothreads
Internet Protocol Suite (TCP/IP) networking, including IPv6
Windowing system and GUI
Networked remote display using Virtual Network Computing
A web browser (claimed to be the world's smallest)
Personal web server
Simple telnet client
Screensaver
Ports

Communication components in Contiki

uIP and uIPv6. There are several advantages of having the Internet protocol stack embedded in the OS. It allows communication with the existing devices/computers in the Internet. For this reason, the IP stack has been trimmedtofit1kBofRAMandafew kB of ROM, in comparison with Linuxbased IP stack that requires 1MB RAM. This trimming of memory comes at the cost of reduced throughput.

uIPv6 is in contrast to the IPv6certified version of IPv6, which was developed by Cisco. Support of IP stack along with TCP and UDP protocols makes it possible for devices running Contiki to directly interact with a Web server or tweak messages to the Internet.

LoWPAN (IEEE 802.15.4) and 6LoWPAN. Low-power wireless personal area networks have the characteristics of small packet sizes, low data rates, low-power devices and large number of devices. The IPv6 extension of the same is the 6LoWPAN standard developed by IETF working group. For saving power, the transmission and receiving periods are short spaced along with long periods of sleep mode by devices. Also, for power constraint reasons the communication is hopbased and short-range.