6LoWPAN NETWORK USING CONTIKI OPERATING SYSTEM
Kavyashree E D*
Dept of Computer Science, A T M E College of Engineering, India, Karnataka, Mysuru

DOI: https://doi.org/10.21467/proceedings.1.48
* Corresponding author email: kavyashreed@gmail.com

Abstract
Contiki an open source operating system for Wireless Sensor Networks (WSNs) and Internet of Things (IoT). It is the growing field in the technology and it is the evolution of the internet also known as network of network. Each objects, sensors, actuators and systems are networked and automated to provide new services which were not available before. Light weights, resource constraints, low cost motes are used in WSNs which leads to major challenges in security applications. Data collection are carried by sensors and computing can be done by sensor nodes itself or by external device. This paper focuses on the working of 6LoWPAN network using contiki tool with cooja simulator and various WSNs simulation tools are briefed. Paper also presents why contiki is used, its features and advantages of Cooja simulator. Cooja verify and debug the software and supports low power standards like 6Lowpan, coap, RPL in coniki. This work also focuses on data collection in rime with simulated results.

Index Terms- IoT, WSNs, Contiki, Cooja, 6LoWPAN, Rime, Simulator

1 INTRODUCTION
Contiki runs on a tiny low cost power microcontrollers and develop applications that make efficient use of the hardware while providing standardized low power wireless communication for the variety of hardware platform. [1] It is used both in various commercial and non commercial systems for example alarm systems, street lights, industrial monitoring, radiation monitoring, remote house monitoring, sound monitoring, site monitoring etc.[2][3] Contiki has one of the important tool called cooja helps in development allow developers to test their code before running into target hardware. Cooja is a software simulator, it is open source, basically build in java, capable of executing C, C++ programs it is a network simulator designed for wireless sensor networks. Support fully standardized IPV6 and IPV4, support multiple platform such as Z1, Skymote, MicaZ etc.[4][5][6]. Simulations are used to verify the behavior of the system and debug their software. Hardware platform e.g. Skymote, Z1 mote etc. Contiki is used compile and upload program in mote. Different type of mote are Z1mote, Skymote, MicaZ, ESB, Cooja, Wismote mote etc. [7][8][9][10].
To compile and upload a program into a mote. Target need to be set
make target=z1 programname
make programname.upload // (upload compiled program into a mote)
make login target=z1 // (login to view compiled code)

**cd makefile:** makefile is essential to compile harder program in cooja, it link essential files and build system to our program.

`contiki=/home/user/Desktop/contiki-2.7` (linking contiki 2.7 folder)

`include $(CONTIKI)/Makefile.include` (including build system)

To update contiki OS
`cd contiki -2.7 cvc update –dp`

---

**IP Protocol Stack**

| HTTP | RTP |
|------|-----|
| TCP  | UDP | ICMP |
|      | IP  |
| Ethernet MAC |
| Ethernet PHY |

**6LowPAN Protocol Stack**

| Application Protocols |
|-----------------------|
| UDP                   |
| ICMP                  |
| IPv6                  |
| LoWPAN                |
| IEEE 802.15.4 MAC     |
| IEEE 802.15.4 PHY     |

---

**IPv6** [11][12][13][4][15][16]

**Pros**
- No need of DHCP
- Support the multicasting against broadcasting
- Header is designed to reduce header overheads.
- Addresses supported is very large

**Cons**
- Challenging is deployment
- Expensive Multicasting
- Managing long addresses
Why 6LowPAN? [17][18][19] 
It extends already existing IP Network, IPV6+LoWPAN (Lowpower RF), Simple Header Structure, Managed duty cycles for saving power, IP Networking End to End Light Weight i.e. No full fledge TCP/IP stack and related complexity. In contiki we have multiple independent platforms such as cooja, sky, this is how we run hardware code in contiki.

2 FEATURES OF CONTIKI AND SIMULATION TOOLS IN WSN’s

This section represents various simulation tools used in sensor networks and why contiki is used and its features. Simulator used for developing and testing protocols of WSN’s. Simulation is performed in design stage. Execution time in simulation is very less and cost of simulating multiple nodes is very less. Below table represents various WSN simulation tools [8][9][10].

| Simulators  | Features                                      | OS              | Merits                          | De-Merits                              | Platform(s) | License          |
|------------|-----------------------------------------------|-----------------|---------------------------------|----------------------------------------|-------------|------------------|
| NS-2       | Real time network simulator                    | Linux, Windows  | Protocols are publically available | Must be familiar with scripting language | C++         | GPL              |
| NS-3       | Open environment for researchers               | Linux windows   | Not an extension to NS-1, new simulator | Supports only IPV4                     | C++, Python  | GPL              |
| OMNeT++    | Less power consumption                        | Windows         | Sophisticated graphical user interface | Compatibility Issues                   | C++         | Commercial, Academic |
| MATLAB     | Numerical computation, Visualization          | Windows, Linux, macOS | Additional toolboxes, Graphical output. | High space complexity               | C, C++, Java, Python | Open source |
| Contiki    | Real time                                     | VMware          | Highly Portable, Multitasking    | Resource constrained system            | C           | Open Source      |
| Castalia   | Sensing modeling provisions                   | Windows, Linux  | Implementation of MAC protocols | Not a sensor specific platform         | C++         | Commercial, Academic |
| J-sim      | Provide GUI library for animation, debugging and, tracing | Linux, Windows XP | Provide mobile networks and sensor networks | Low efficiency in simulation             | Java        | Commercial       |
| QualNet    | Provide scalability via supporting parallel execution | Solaris, Linux, Windows | Supports distributed computation in system | Expensive                             | C++         | Commercial       |
| TOSSIM     | Easy to monitor packet traffic                 | Windows         | Visualization tools are available | Compilation issues                    | NesC, C++, python | Open source |

Why Contiki - It is open source OS for WSNs and IoT. It connects tiny low power, low cost microcontrollers to the internet, it is also powerful toolbox for building complex wireless
systems and it is used for both commercial and non commercial systems, no need to write code, full source code is available in one instant contiki download [11].

**Internet Standards**
Contiki works on low power, and provide internet standards. Supports IPv4 and IPv6 and low power wireless standards such as 6LoWPAN, CoaP,RPL. Contiki MAC and wireless routers can be battery operated.

**Rapid Development**
In Contiki development is fast and easy. Applications of contiki are written in C, Single download of Instant contiki provides an development environment. In Cooja simulator contiki networks is emulated before burned into hardware.

**Hardware Selection**
Contiki operate on a range of wireless device which is low power, these can be purchased easily in online so it supports selection of hardware.

**Active Community Support**
Contiki is developed by a team of world-wide developers with the contributions from Cisco, Atmel, Redwire LLC, ETH, Thingsquare, SAP, and many more developers led by Adam Dunkels of Thingsquare.

### 2.1 FEATURES OF CONTIKI
Contiki has huge features some of the features are briefed below [11]:

**Dynamic Module Loading**
Contiki performs Dynamic Module Loading and linking at run time. Module loader can load, relocate, and link standard ELF files that can optionally be stripped of their debugging symbols to keep their size down. This is useful in various applications where the behavior is intended to change after deploying it. Path is contiki/core/loader/

**Cooja Network Simulator**
It helps the programmers to see their applications run in large scale networks or in high detail on fully emulated hardware devices. Contiki devices build complex wireless network. So developing and debugging programs in such a network is difficult so cooja helps in providing simulation environment where developers can easily debug their software. Contiki/tools/cooja/

**Build Systems**
It is simple to compile applications in contiki platform. It is easy to try out various applications on a variety of different platforms, even if the hardware is not available can perform the operations using the cooja simulator to emulate hardware devices. Path is contiki/Makefile.include.

**Hardware Platforms**
Contiki operates on a range of a Hardware platforms and it is easy to port to new hardware. Hardware platforms such as cc2538dk, exp5438,z1, wismote, sky, micaz, cooja, RE-mote etc.
Memory Footprint
Contiki run in very less amount of memory. IPv6 network need less than 10 k RAM and RPL routing needs less than 30 k ROM.

Regression Tests
In contiki code regression test is performed on cooja simulator so that contiki code work as expected. Test scripts is used in regression as a initial point for configuring up simulations or to investigate how various contiki mechanisms work. Code in contiki/regression-test/.

Contiki Shell
Contiki supports an optional command line-shell, which contain set of commands that are useful during code implementing and debugging in contiki systems. Shell commands are combined in a potent ways. Application defines their own shell commands that work with the existing commands. Contiki/apps/shell/

Protothreads
Contiki OS to save memory and to provide good control flow in the code it uses a mechanism, Known as protothreads, it is assortment of multi-threaded and event driven programming mechanisms. Event handlers can be made to block for events to occur. Code in contiki/core/sys/pt.h

Coffee Flash File System
Coffee is known as Lightweight flash file system. This file system application programs can perform various open, close, read from, write to, and append to files on the external flash, without having to worry about flash sectors needed to be erased before writing or flash wear-leveling. Performance of coffee is within 95% of the raw throughput of the flash memory. Contiki/core/cfs/cfs-coffee.[ch].

Examples
There are various rich examples in contiki source code tree to help the beginners. Show program shows how to interact with the platform hardware, some programs show how to network code, and other program demonstrate the various aspects of contiki system and corresponding cooja simulator available. Can find the code in contiki/examples/.

3 EXPERIMENTAL RESULTS USING SIMULATOR
This section represents the working of 6Lowpan in contiki cooja simulator using hardware platform such as cooja, sky. This show running hardware code in contiki using cooja simulator. Cooja has huge advantages some of them are listed below [7][15][18][19][20][21][22].

- It is an open source
- It has simple UI flexible, efficient, and scalable.
- Supports multiple platforms like sky mote, Z1mote, etc.,
- Support Native TCP/IP
- Uses basic C, C++ codes programming language.
- Supports Fully standardized IPv4 and IPv6.
- Supports low power standards like 6Lowpan, coap, RPL, etc.

**6LoWPAN**: Extends already existing IP Network, IPV6+LoWPAN (Lowpower RF), Simple Header Structure, IP Networking End to End Light Weight i.e No full fledge TCP/IP stack and related complexity and Managed duty cycles for saving power.

Fig 4.1: Displays Network window, consisting of 3 client motes and 6 sensor motes with LED and packets distribution to each nodes.

Fig 4.2: Displaying 9 Motes Timeline with different events like radio traffic, radio on/off, radio channel, LED colors, and radio events, black dots represents the radio events, red, blue and green line show the glow of correspondent LEDs lights on network window.
Fig 4.3: Displaying radio messages of 6LoWPAN packets, using 802.15.4 data in physical layer. IPv6 ICMPv6 means using 6Lowpan application with skymote.

Fig 4.4: Motes console output initially, showing the time for each mote communication, mote ID, and Messages from one mote to another mote. Mote with ID-5 is a UDP server and ID-3 is a client. Client and server connection is established using local/remote port 5678/8765. ID-1,2,3 are client, ID-4,5,6,7,8,9 are server.
Fig 4.5: Motes console output window prints results, transferring and receiving data ‘Hello’ in each Motes. ID 3 is the client sending data ‘Hello 4’ to ID 4. ID 4 is a server receiving data ‘hello 4’ from the client 3.

Fig 4.6: Stack Watcher to identify stack push and pops and can monitor stack overflows.
3.1 DATA COLLECTION WITH RIME [7]
Rime Stack is one of the feature of Contiki: For instant if bandwidth is at top or when the full IPV6 stack network is overload contiki uses rime, it is tailored wireless networking stack in contiki. It perform operation like sending a message to a specified neighbor or all neighbors, also support difficult mechanisms like network flooding and address free multi hop semi reliable scalable data collection. To save power it operates on sleepy routers. Code path is available in contiki/core/net/rime/.
Once the simulation is started we have different windows like network window, simulation control window, notes window, mote output window, timeline window.
- **Network Window**: Where we can deploy nodes and apply attributes to it. we can create mote by following steps Motes->Add motes->create new mote type->sky mote.
- **Simulation Control Window**: Control the particular simulation in the form start, pause, step and reload. Step to run the simulation for short interval of time that is 0.1 milli second, Reload helps to load the program automatically without having to do manually.
- **Notes Window**: Can take down important motes regarding particular simulation save.
- **Mote Output Window**: Display output of the simulation message passed between motes, if there is a base station it show how the messages of all node communicate to it.
- **Timeline Window**: Show how Simulation proceeds with respect to time in seconds

Fig 4.1.1: Deployment of motes in Network window, LEDs for each mote with background grid to manually position X and Y axis to these motes. As red line and green line is passed in timeline output ,red and green LED gets lighted up in network window.
Fig 4.1.2: Description of the sky mote platform with sky mote ID, sky mote type, identifier, contiki source, contiki firmware, and compiled commands TARGET=sky.

Fig 4.1.3: Packets simulation gets started in network window and correspondent message gets printed in mote output window, simulation can be controlled by simulation control output, and timeline shows simulation with respect to time.

4 CONCLUSION

This paper focuses on various simulator tools for WSNs. One of the tools in WSNs is Contiki operating system used in domains like IoT and WSNs and used for real-time application due to its real-time capability. It provides vast features to its users like rapid development, active community support, regression testing, dynamic module loading and linking, Memory footprint, and the...
prime benefit in Contiki is source code is available for beginners by instant Contiki download and applications are written in C language. Contiki working is shown by using cooja simulator to compile and debug the software programs. Low power standards 6LoWPAN and working of Rime is showed using sky mote hardware platform using cooja simulator.

REFERENCES

[1] A Dunkels, Björn Grönvall, To Voigt, Contiki - a Lightweight and Flexible OS for Tiny Networked Sensors, 29th Annual IEEE International Conference on Local Computer Networks.

[2] Patrick Kugler, Philipp Nordhus and Bjørn Eskofier, Shimmer, Cooja and Contiki: A New Toolset for the Simulation of On-node Signal Processing Algorithms, 2013 IEEE.

[3] T Vortler, B Höckner, P Hofstedt, Thomas Klitz, Formal Verification of Software for the Contiki Operating System Considering Interrupts, 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits & Systems.

[4] Moataz F. Youssef, Khaled M. F. Elsayed and Ahmed H. Zahran, Contiki-AMAC – The Enhanced Adaptive Radio Duty Cycling Protocol: Proposal and Analysis, 2016 International Workshop on Scalable Internet of Things, IEEE.

[5] Uzair A. Nomani, Behailu Negash, Amir M. Rahmani, Pasi Liljeberg, and Hannu Tenhunen, From Threads to Events: Adapting a Lightweight Middleware for Contiki OS, 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCCN).

[6] Kavyashree E D1, Vidyashree H D2, Anil Kumar B H3, A Survey of Internet of Things (IoT) - Applications, Merits, Demerits & Challenges, IJIRCE Volume 6, Issue 2, Feb 2018.

[7] https://www.contiki-os.org/videos [Available 30 March 2018].

[8] www.contiki-os.org [Available 30 March 2018].

[9] M Korkalainen, M Sallinen, Niilo Kärkkäinen, P Tukeva, Survey of WSN's Simulation Tools for Demanding Applications, 2009 Fifth International Conference on Networking and Services.

[10] Andhe Dharanil, Shantharam Nayakz, Manjuprasad B3, Hardware and Simulation Perspective Study on Wireless Sensor Networks, IAIFRC, Volume 1, Issue 7, July 2014.

[11] https://pdfs.semanticscholar.org [Available 30 March 2018].

[12] www.comnets.uni-bremen.de/fileadmin/ComNets/WSN-book/presentation.pdf [Available 30 March 2018].

[13] M Kovatsch, S Duquennoy, Adam Dunkels, A Low-Power CoAP for Contiki, 2011 Eighth IEEE International Conference on Mobile Ad-Hoc and Sensor Systems.

[14] Simon Duquennoy, Atis Elsts, Beshr Al Nahas and George Oikonomou, TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation, 2017 13th International Conference on Distributed Computing in Sensor Systems.

[15] https://pdfs.semanticscholar.org [Available 30 March 2018].

[16] Kamaldeep, Manisha Malik, Dr. Maitreyee Dutta, Contiki-based mitigation of UDP flooding attacks in the IoT, International Conference on Computing, Communication and Automation (ICCCA2017).

[17] c.o. Iwendi, A.R. Allen, Enhanced Security Technique For WSN Nodes, University of Aberdeen, Scotland, UK.

[18] Challouf Sabri, Kriaa Lobna, Saidane Leila Azzouz, Comparison of IoT constrained devices operating systems : A Survey, 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications.

[19] www.contiki-os.pptx. [Available 30 March 2018].