Implementation of a Lightweight RPL Protocol Stack with Finite-State Machine and CC2530 Chip

Ping Wang
Key Laboratory of Industrial Internet of Things and Networked Control
Chongqing University of Posts and Telecommunications
Chongqing, China
wangping@cqupt.edu.cn

Heng Wang
Key Laboratory of Industrial Internet of Things and Networked Control
Chongqing University of Posts and Telecommunications
Chongqing, China
wangheng@cqupt.edu.cn

Chenggen Pu
Key Laboratory of Industrial Internet of Things and Networked Control
Chongqing University of Posts and Telecommunications
Chongqing, China
mentospcg@163.com

Abstract—Routing protocol is one of the core technologies of wireless sensor networks. RPL (Routing Protocol for Low-Power and Lossy Networks) routing protocol, designed by ROLL Working Group, is mainly implemented on the TinyOS and Contiki operating system. In order to extend the application fields of RPL, we implement a lightweight RPL routing protocol based on Finite-State Machine protocol stack with CC2530 chip. In this paper, the design of the node hardware is introduced first, which takes the TI wireless sensor network chip CC2530 as a core component. And then we described the IPv6 protocol stack based on Finite-State Machine, layered mechanism is used to make the protocol stack configurable and scalable. Then a lightweight RPL routing protocol is implemented on the above basics. Finally, a RPL experiment platform with nine nodes is built to test the network routing function, the results show that the RPL basic function is achieved with good performance.

Keywords-RPL; WSN; FSM; routing;CC2530

I. INTRODUCTION

Wireless sensor networks, consists of a large number of various types of sensor nodes which transmit information through the wireless networks, is an integrated network system which fuses the collection, processing, transmission and application of information together [1-3]. Wireless sensor nodes are tiny embedded communication devices with sensing and collecting. The function of routing protocol is to solve the problem of data transmission, which is one of the core technologies of wireless sensor networks [4]. And the performance of routing protocol is closely related to the performance of the entire network. A routing protocol for LLNs called RPL is proposed by the IETF ROLL Working Group after analyzing the routing requirements on several application scenarios such as urban networks including smart grid, industrial automation, home and building automation [5, 6].

So far, the RPL routing protocol is mainly implemented on the TinyOS and Contiki operating system [7, 8]. In order to make RPL widely applied to wireless sensor networks fields, we implemented a lightweight RPL routing protocol based on finite-state machine using CC2530, according to the application scenarios with limited node capacity and without operating system. This will lay the foundation for RPL more widely used in wireless sensor networks.

II. SYSTEM STRUCTURE OF WIRELESS SENSOR NETWORKS

A. Hardware structure of sensor node

Sensor nodes are usually consist of four parts: the power supply unit, the data acquisition unit, the data processing unit (including microcontroller and storage), and the wireless communication unit. Among them, the microcontroller of the data processing unit controls the other three units.

In this paper, the hardware platform is based on CC2530 wireless module [9, 10]. As an upgrade version of CC2430 chips, CC2530 has more excellent performance and abundant resource. And with memory space up to 256kbyte and 8051 microprocessor running at 32 MHz, CC2530 has great advantages in coding space and computing capacity comparing with other chips at same price. It is worth mentioning that CC2530 possesses the 2.4GHz wireless RF front accorded with the IEEE 802.15.4 standard, hardware CSMA/CA function, address

filtering, ACK automatic reply function, digital received signal strength indicator(RSSI) and link quality indication(LQI).

B. *Software structure of sensor node*

The experiment platform in this paper is based on 6LoWSN stack, which is a self-developed IPv6 wireless sensor network protocol stack in our laboratory mainly using the Finite State Machine (FSM). In order to make the protocol stack configurable and scalable, layered mechanism is used in 6LoWSN. The protocol stack from top to bottom is divided into four layers, namely the application layer, network layer, adaptation layer and data link layer. The hierarchical structure of 6LoWSN protocol stack is shown in Fig.1.

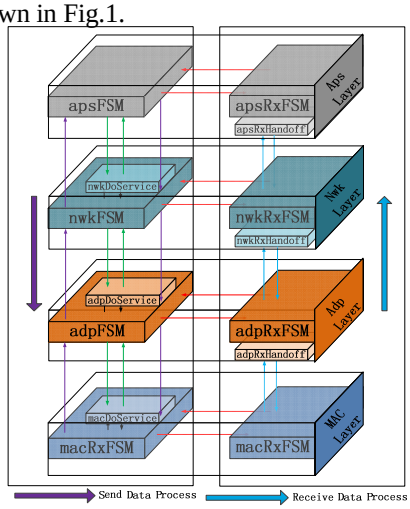

Figure 1. Hierarchical structure of protocol stack

For each layer of the protocol stack have a master Finite State Machine and a receive Finite State Machine. The master FSM is the main body of protocols implemented in every layer, consisting of the corresponding functions and services of the protocols. The receive FSM in each layer is responsible for receiving data from the lower layer, after parsing the incoming data it may pass the data to the master FSM or handoff to the up layer's receive FSM depends on the incoming data.

III. THE IMPLEMENTATION OF LIGHTWEIGHT RPL

A. *Overview of RPL*

Using objective function and a set of metrics/constraints, RPL builds a Destination Oriented Directed Acyclic Graph (DODAG) to establish route, per DODAG per Root. The objective function operates on a combination of metrics and constraints to compute the 'optimal' path. The RPL routing protocol specifies a set of new ICMPv6 control messages to exchange graph information and build topology, which are DODAG Information Solicitation (DIS), DODAG Information Object (DIO) and Destination Advertisement Object (DIO) respectively. RPL use DIO build and maintain DODAGs, and use DAO to establish downward route.

The objective function (OF) of RPL defines that how a node select and optimize the routes in a RPL instance. The OF is marked by the OCP of the configuration option of DIO. An OF defines how nodes translate one or more metrics and constraints into a value called Rank, which approximates the node's distance from a DODAG root. And the OF also defines how nodes select parents. RPL supports point-to-multipoint (P2MP), multipoint-to-point (MP2P) and point-to-point (P2P) traffic.

B. *Implementation of RPL*

The implementation of lightweight RPL which we discussed in this paper is composed of the following modules: build upward route, establish downward route and routing.

1) *Build upward route*

The DODAG building process is the process of RPL constructing upward route. The building process of the graph starts from the root, which is configured by the system administrator. In this paper, when a new node wants to join the DODAG, it wills multicast a DIS proactively. The non-leaf neighboring nodes that have already joined the DODAG will send DIO through multicast. Then the new node will receive and process DIO message and decide whether to join the graph or not. This process is shown in the Fig.2.

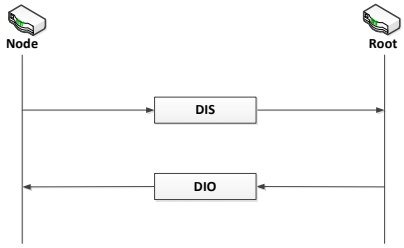

Figure 2. Node joining DODAG process

The program flow chart of the process that a node joins the DODAG and completes upward route is shown in Fig.3. Once a node has joined to a graph, it has a route lead to DODAG Root. Besides, nodes can also join DODAG by listening neighboring nodes of the root, processing the DIO they recieved, and choosing to join a DODAG. All neighboring nodes repeat this process, until a graph edges out from the root to the leaf nodes is established. Consequentially, every node has more than one route to the root, just forwarding the packet to its immediate parent, the packet can arrive Root. This is referred to as upward routing, also called multipoint-to-point (MP2P) mode.

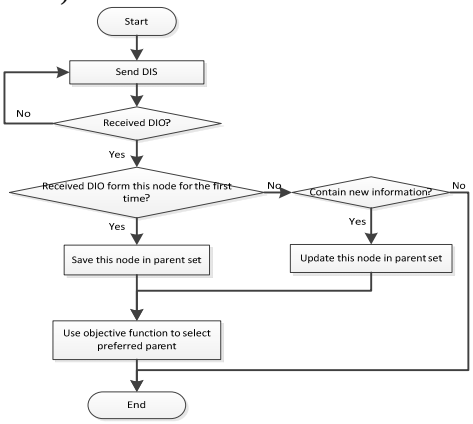

Figure 3. Program flow chart of joining DODAG process

2) Build downward route

Corresponding to MP2P upward communication mode (data stream transmitted from the leaf node to the root node), the network sometimes needs to send the data stream downward. In RPL, there are two modes for the downward route to choose. The first one is called "storing" mode, which the node stores DODAG downward routing table. In the "storing" mode, every downward packet checks its routing table to determine the next hop. The second one is called "non- storing", which the node does not store downward routing table and downward packets is routing by the DODAG root with source routes. In the paper, we choose the "storing" mode to create a downward route, at this moment every non-leaf-node needs store routing tables for their sub-graph.

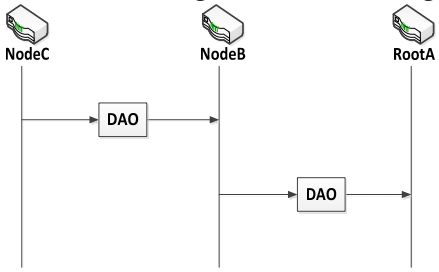

Figure 4. Establish downward route process

In order to realize the downward route, every node that has joined DODAG need send DAO to the parent node. After receiving the DAO message, the node parse and store the prefix information, then report the prefix information to its parent by send a new DAO. Finally, the node sends prefix information to the root to form a full downward path. The building process of downward route is shown in Fig.4, assuming that the node C is a new node that added to DODAG, B is the parent node of C, and A is the root node.

After the parent node received the DAO that sent by child node, it first check whether the child node is in the table of its child nodes or not. If it is in the table, the parent node checks to see whether it contains new path information or not. If it is not in, the parent node adds it into the collection of its child nodes, and then reports the node information to its parent node. The program flow chart of establish downward route in shown in Fig.5.

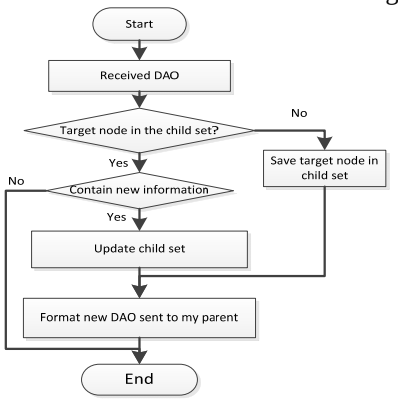

Figure 5. Program flow chart of establish downward route

3) Send and forwarding packets

When a node wants to send data packets，the next hop is selected depending on the destination IPv6 address of the data package. If the destination IPv6 address is root, the next hop address is the preferred parent node. If the IPv6 address of destination is not the root node, it may be a child node, in this case check the collection of its child nodes. If it is in, select the corresponding child as next hop. If it is not, it may be a point to point packets or the packets that will be sent to other networks, at this time，the next hop is the preferred parent node.

IV. SYSTEM TESTING

According to the implementation of lightweight RPL proposed above, the test and verification of it will be done in this section. A small-scale network is formed in the paper. During the test process, nodes are connected with PC through the serial port, and the SmartRF Packet Sniffer is used to capture packets that transmitting in the air. Through the serial port debugging assistant and packet sniffer software, we can monitor the DODAG building process and packets forwarding process.

A. Build DODAG test

Applying nine nodes to form the testing network, the nodes use 16-bit short address as link layer address. The process of a node joining DODOAG is shown in Figure 2. A node which wants to join the DODAG multicast the DIS message firstly, then waits the DIO message replied by the non-leaf nodes which have joined the graph before. Second, node parse the received DIO message, and choose the preferred parent node by OF. Through this it has a path to the Root, which means the upward route is set up. The information of serial port debugging assistant of node sends DIS and receives DIO and sends DAO to its parent is shown in Fig.6. The information of serial port debugging assistant of node receives DIS and replies DIO message and deals the DAO message is shown in Fig.7. The screenshot of packet sniffer is shown in Fig.8.

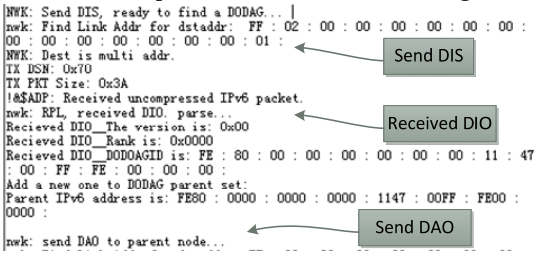

Figure 6. Node joinning DODAG

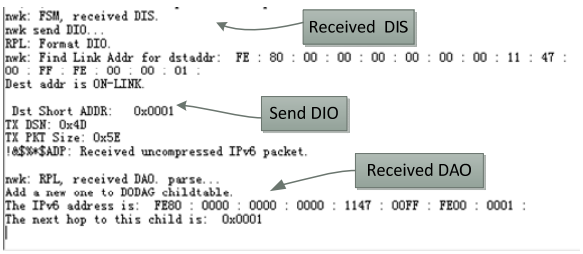

Figure 7. Process new node join DODAG

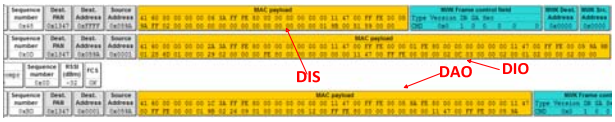

Figure 8. Screenshot of Packet Sniffer of node join DODAG process

B. *Packet forwarding test*

The nine nodes are arranged in a certain distance, and the topology of network after the establishment of the DODAG is shown in Fig.9. The test platform is showed in The Fig .10.

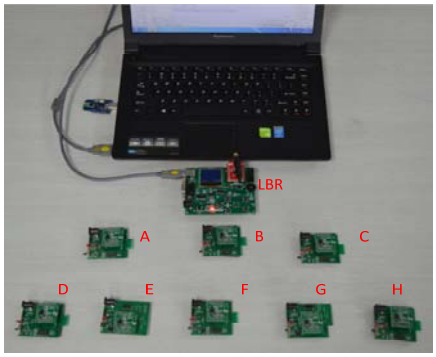

Figure 9. RPL experiment platform

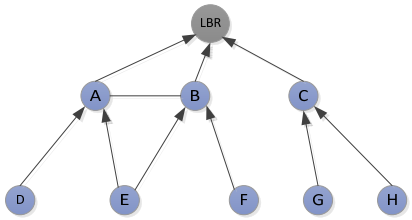

Figure 10. Topology of the network

Take node sending data to LBR for example, node D will send data to its preferred parent node A firstly, as shown in Fig.11. Then A forwards the packet to its preferred parent node LBR, as shown in Fig.12. It is easy to see LBR received packet form D in Fig.13. If LBR wants to send packets to node D, it will check the child nodes table. After finding the next hop to node D, LBR will transmit the packet to corresponding node, here the next hop is A. And then node A will use the same operation to forward the packet to node D in the end. The screenshot of Packet Sniffer of the forwarding process is shown in Fig.14.

```
Dst IPv6 address is: FE80 : 0000 : 0000 : 0000 : 1147 : 00FF : FE00 : 0000 :
NWK: This is not for me , do up route...
Next hop is: 0x0001
TX DSN: 0xD4
TX PKT Size: 0x46
!#$%*Sensor Data Sent Successfully!
```

Figure 11. D send packet to LBR via A

```
!#$%*$ADP: Received uncompressed IPv6 packet.
Dst IPv6 address is: FE80 : 0000 : 0000 : 0000 : 1147 : 00FF : FE00 : 0000 :
NWK: This is not for me , do up route...
Next hop is: 0x0000
TX DSN: 0x13
TX PKT Size: 0x46
```

Figure 12. A forwarding packet

```
!#$%*$%$%$ADP: Received uncompressed IPv6 packet.
UDP packet
Data Received.
Source IP:
FE80 : 0000 : 0000 : 0000 : 1147 : 00FF : FE00 : 059A :
Source Port. 0xF0B3, Service Type. 0x01
, Msg Length: 0x05
Msg:  0x01  0x01  0x00  0x01  0x21
```

Figure 13. LBR received packet form D

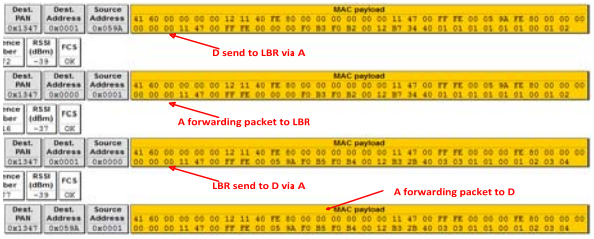

Figure 14. Screenshot of Packet Sniffer of forwarding packet process

The test result above shows that the lightweight RPL routing protocol discussed in this paper achieves the DODAG building and packet forwarding functions.

V. CONCLUSION

In this paper, we describe the hardware architecture of wireless sensor network firstly, and then present the software architecture of the WSN node which based on finite-state machine. Second, the lightweight RPL routing protocol is implemented in the paper. Finally, through test the DODAG constructing and multi-hop forwarding mechanisms of RPL, the validity of the lightweight RPL routing protocol that presented in the paper is verified.

ACKNOWLEDGMENT

This work was supported by the National High Technology Research and Development Program of China (Grant No. 2015AA043801), and the Chongqing science and technology R & D base construction plan (International Science and technology cooperation project) (Grant No. cstc2013gjhz40002).

REFERENCES

[1] Yick, Jennifer, Biswanath Mukherjee, and Dipak Ghosal. "Wireless sensor network survey." Computer networks 52.12 (2008): 2292-2330.

[2] Nowak, Thorsten, et al. "Combined Localization and Data Transmission in Energy-Constrained Wireless Sensor Networks." Wireless Sensors and Sensor Networks (WiSNet), 2015 IEEE Topical Conference on. 2015.

[3] Lewis, Franck L. "Wireless sensor networks." Smart environments: technologies, protocols, and applications (2004): 11-46.

[4] Akkaya, Kemal, and Mohamed Younis. "A survey on routing protocols for wireless sensor networks." Ad hoc networks 3.3 (2005): 325-349.

[5] Vasseur, J., Navneet Agarwal, Jonathan Hui, Zach Shelby, Paul Bertrand, and Cedric Chauvenet. "RPL: The IP routing protocol designed for low power and lossy networks." Internet Protocol for Smart Objects (IPSO) Alliance (2011).

[6] Winter, T., P. Thubert, and R. A. Team. "RPL: IPv6 routing protocol for low power and lossy networks, RFC 6550." IETF ROLL WG, Tech. Rep (2012).

[7] Dunkels, Adam. "The contiki operating system." Web page. Visited Oct 24 (2006).

[8] Levis, Philip, et al. "TinyOS: An operating system for sensor networks." Ambient intelligence. Springer Berlin Heidelberg, 2005. 115-148.

[9] Instruments, Texas. "CC2530 data sheet." 2010—10-05) http://www. ti. com (2009).

[10] Li, J. B., and Y. Z. Hu. "Design of ZigBee network based on CC2530." Electronic Design Engineering 19.16 (2011): 108-111.