2012 International Conference on Information and Computer Networks (ICICN 2012)
IPCSIT vol. 27 (2012) © (2012) IACSIT Press, Singapore

RTT Estimation Method in WSN

Hyun-Chul Yi [1], Hong-Ju Kim [2], Yu-Jong Kim [3], Joon-Young Choi [4 +]

[1 4] Department of Electrical Engineering, Pusan National University, Busan, Korea

[2] KERI (Korea Electrotechnology Research Institute)

[3] Education Program for Samsung Advanced Integrated Circuit, Pusan National University, Busan, Korea

[1 3 4] {schezow, counterror, jyc}@pusan.ac.kr

[2] hjkim0429@keri.re.kr

Abstract. This paper proposes method of estimates a round trip time in Wireless Sensor Network. Wireless network had dynamic transmission capacity because of many reasons. So, flow control is more important issues in the network. Using by proposed method, wireless network can use RTT values to flow control in the network. A network layer in a network protocol stack of a TinyOS, generally used in the WSN, uses ACK information from link layer, whether success or failure of each node, that have implicit information of network flow. Using this stack, we can reduce effort to develop. Although less effort to develop the method, we can get the stable working and simple method.

Keywords: RTT, Network, WSN, Congestion

1. Introduction

In a wired network had uniform transmission capacity of communication channel's, whereas, wireless network had flexible transmission capacity of communications channel's that depend on flow of times caused by RF output power, to change in the mobile environment, etc [1], [2], [3] and [4]. This is a major cause of study about new network protocols, algorithms, etc. In the research area, there have many concepts to deal with network protocol algorithms in the network layer, but, there are less concerning the control of flow protocol in transport layer.

Before the study of flow control algorithm in the transport layer in WSN, we decide to develop and implement estimation method of a RTT (Round Trip Time). RTT estimation method has faster response than methods in [5], [6] and [7] those using network topology lengths, conditions of node's buffer states, etc. Most of all, the proposed method reduced effort of development. We modified TinyOS's implemented library that is used generally in WSN. Therefore, we can reduce effort to develop new library and footprints.

We will describe the proposed method in chapter 2 and we will see the analysis by experiment in chapter 3. After all, we will reach to the conclusion in the chapter 4.

2. RTT Estimation

2.1. RTT Estimation Equation

Equation (1) is common equation to calculate RTT in the network. RTT_{START} is just before to send packet that setup an ACK request bit in a packet header from send node to receive node. RTT_{END} is just after to receive the ACK from receiving node to send node.

$$RTT_{TOTAL} = RTT_{END} - RTT_{START} \qquad (1)$$

+ Corresponding author. Tel.: +82-51-510-2490; fax: +82-51-515-5190.
E-mail address: jyc@pusan.ac.kr.

But, if we don't imply this equation in a TinyOS, we aren't getting accurate a RTT_{TOTAL} value. Because all WSN's motes had precision and limitation of variable, that is, overflow problem. When the RTT_{END} has smaller value than the RTT_{START}, the RTT_{TOTAL} has negative value.

Fig. 1 shows that pseudo code solves this overflow problem. In this code, RTT_{TOTAL} always have positive value. In this paper, we try to prove that the telosb platform has 16bits limitation. In implementation, variable's limitation value is to be 0xFFFF.

```
If ( RTT_END < RTT_START )
    RTT_TOTAL =
    Variable's Limitation Value – RTT_START + RTT_END
ELSE
    RTT_TOTAL = RTT_START – RTT_END
```

Fig. 1: RTT estimation psuedo code

2.2. Modify TinyOS Module

In this paper, we use a 2.x version of TinyOS which implemented in the library, a CTP (Collection Tree Protocol) module of TinyOS [8], [9] and [10].

```
//RTT Setter
command void setRtt(uint16_t rtt);

//RTT Getter
command error_t getRtt(uint16_t* rtt);
```

Fig. 2: RTT value store and load function prototypes

Fig. 2 shows inserted RTT setter and getter codes in a CtpInfo.nc file. The CtpInfo.nc file is an interface between TinyOS's modules. By using this modified file, we allowed upper layer to access and use RTT value.

```
//Rtt Start Time
 RttStart = call Timer.getNow();
```

Fig. 3: RTT_{START} value store code

Fig. 3 shows inserted RTT_{START} estimation code in a CtpForwardingEngineP.nc file. A CtpForwardingEngine.nc file is a module that has functions related with packet sending. This code is called just before to call packet sending function.

```
if(ackPending && call
PacketAcknowledgements.wasAcked(msg))
{
    RttEnd = call RetxmitTimer.getNow();
    if (call RootControl.isRoot())
        call CtpInfo.setRtt(0);
    else if ( RttEnd < RttStart )
        call CtpInfo.setRtt(0xFFFF - RttStart + RttEnd);
    else
        call CtpInfo.setRtt(RttEnd - RttStart);
}
else
        call CtpInfo.setRtt(0xFFFF);
```

Fig. 4: Implementation codes of RTT value store

Fig. 4 shows inserted RTT_{TOTAL} estimation code related with Fig. 2 in the CtpForwardingEngine.nc file. This code is called just after to receive the ACK packet from receive node to send node.

```
unit16_t val;

call CtpInfo.getRtt(&val);
```

Fig. 5: RTT value load code

Fig. 5 shows code that read RTT value. Using by CtpInfo interface, the val variable has the RTT value.

2.3. TinyOS Visualization Tool

In the TinyOS, there is included visualization tool, the MViz. Using by this tool, we could see the network changing information to a graph and a table.

```
$ tos-mviz -comm serial@COM3:telosb MVizMsg
C:\cygwin\opt\tinyos-2.x\support\sdk\java\tinyos.jar;.
Making MVizMsg
Loading interval
Loading origin
Loading count
Loading reading
Loading etx
Loading link <route>
Loading parent
Loading rtt
./mote.gif
serial@COM3:115200: resynchronising.....hybus
```

Fig. 6: Mviz execution message

Fig. 6 shows the command to execute the MViz. For execute this command, a mote must connect to host PC. This mote has node id to 0 and to be a root node. The packets will be sent to top-level root node through each node's packet. The top-level root node collects packets through each node and sends the node's information to a host pc.

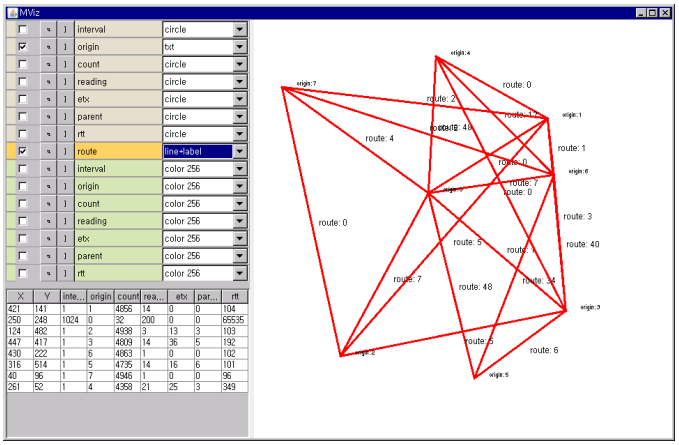

Fig. 7: Mviz graph

Fig. 7 shows the MViz graph includes all nodes' information. In the graph, left upper parts are options to visible of right area. Left lower parts are table that had node's interval (packet sending interval), origin (node id), count (sending count), reading (sensor value), etx (link quality from self node to parent node), parent (self node's parent node id), rtt (RTT from self node to parent node) information.

3. RTT Estimation Experiments

In this chapter, we will test the performance of the proposed method by changing the packet sending interval. Using a total of 8 motes, we make Multi-Hop Ad Hoc network. The performance compares a base interval that is 1024 ticks, with 512, 256, 128, 64, 32, 16, 8 and 4 ticks.

3.1. A Large Interval

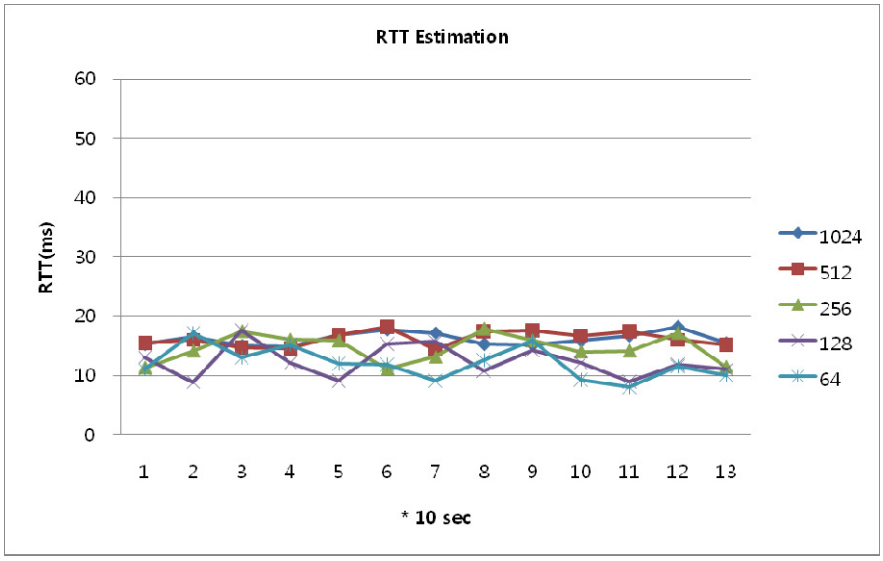

Fig. 8: RTT values in a large interval

Fig. 8 shows the RTT value changes from various packet sending intervals. In this graph, the networks of 64 to 1024 intervals have upper bound of RTT to 20ms. The RTT is estimate continuously in a bounded range.

3.2. A Small Interval

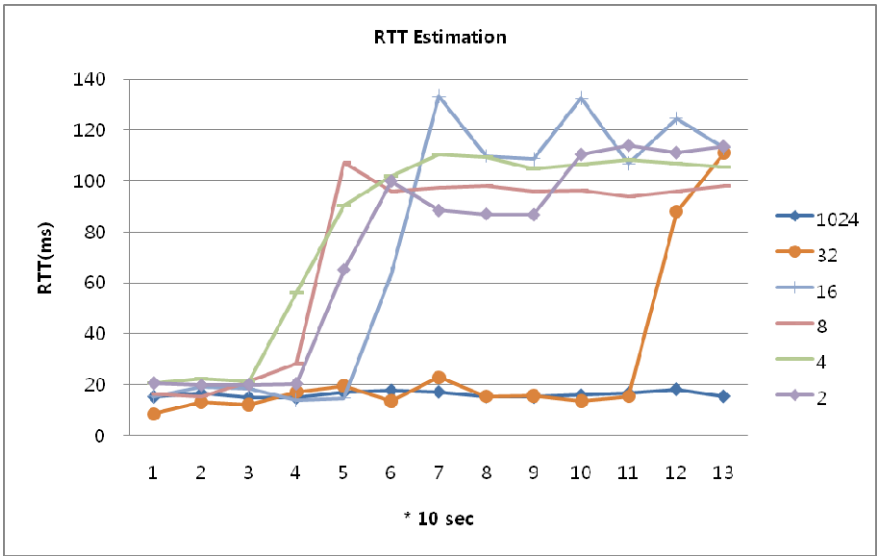

Fig. 9: RTT values in a small interval

Fig. 9 shows the RTT value changes from 32 to 2 intervals. Despite of Fig. 8, less or equal 32 intervals make RTT values significantly increase. In the 1024 interval of network, the RTT value increases after 110

seconds. And the other small intervals, 16 to 2, make the RTT values significantly increase after 30~50 seconds.

Smaller and smaller intervals make a sending rate increase. And then, a number of motes containing packets increase. As time goes on, the number is larger than first time. This is a major cause of increasing delay of the RTT processing.

4. Conclusion

This paper proposes a method for estimates round trip time in WSN. The proposed method has modified method of TinyOS's library. We can see the performance of method by real experiments. By increasing rate of packet sending, the RTT value changes. A large interval makes network have a large RTT in average. In other hand, a small interval makes network have a small RTT in average. This shows that proposed method is stable working.

5. References

[1] Sanjay Shakkottai and Theodore S. Rappaport, "Cross-Layer Design for Wireless Networks", IEEE Communications Magazine, pp. 74-80, Oct. 2003

[2] Shigang Chen, and Na Yang, "Cogestion Avoidance Based on Lightweight Buffer Management in Sensor Netowrks", IEEE Trans. Parallel and Distributed Systems, Vol. 17, No. 9, Sep. 2006.

[3] Yogesh G. Iyer, Shashidhar Gandham, S. Venkatesan, "STCP: A Generic Tranport Layer Protocol for Wireless Sensor Networks", Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN), pp. 449-454, Oct. 2005.

[4] Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks", First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), Nov. 2003.

[5] Yoon-Pil Sung, Myung-June Youn, Jai-Yong Lee, "A Precautionary Congestion Control Scheme in WSNs : Cross layer approach", The 23rd International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2008), pp. 1321-1324, July. 2008.

[6] Joahyoung Lee, Donggug Gim, Inbum Jung, "Adaptive Congestion Control for Effective Data Transmission in Wireless Sensor Networks", Korea Information Processing Society C, Vol. 16C, No. 2, pp. 237-244, April, 2009.

[7] jisoo Shin, Changjin Suh, "LECSEN : Link Exchanged Chain in Sensor Networks", Korea Information Processing Society C, Vol. 15C, No. 4, pp. 273-280. Aug. 2009.

[8] "TinyOS", source-http://www.tinyos.net/

[9] Philip Levis, Neil Patel, David Culler and Scott Shenker, "A Self-Regulating Algorithm for Code Maintenance and Propagation in Wireless Sensor Networks", In Proceedings of the First USENIX Conference on Networked Systems Design and Implementation (NSDI), 2004.

[10] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, and Philip Levis, "Four Bit Wireless Link Estimation", In Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets VI), November 2007.