2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing

SNDS: A Distributed Monitoring and Protocol Analysis System for Wireless Sensor Network

Xin Kuang
Department of Computer Science and Technology
East China Normal University
Shanghai, China
xkuang@ecnu.cn

Jianhua Shen
Department of Computer Science and Technology
East China Normal University
Shanghai, China
jhshen@cs.ecnu.edu.cn

Abstract—**Monitoring a large-scale wireless sensor networks (WSN) is very difficult, not only because it is large and complex, much of difficulty comes from the lack of visual analysis tools. This paper describes *SNDS* (Sensor Network Distributed Sniffer), a distributed monitoring and protocol analysis system for large and complex sensor networks. SNDS is based on the use of sniffers co-deployed with the target sensor network. We mainly study and solve the following difficulties: monitoring sensor networks with large traffic, synchronizing the time among the distributed sniffers accurately, analyzing the protocols efficiently and flexibly. We test the stability and time synchronization accuracy of the whole system. The results show that the SNDS has the capability to work stably, synchronize the time accurately and analyze the wireless protocols efficiently.**

Keywords- WSN; SNDS; time synchroniztion; protocol analysis

I. INTRODUCTION

With the development of WSN, the networks become larger and larger and with the increasingly wide range of its application, many customize protocols are made. These factors make the monitoring of WSN become very difficult. There are some test and analysis tools for WSN, but only for a certain kind of sensor network protocols or custom protocols, therefore there's a lack of flexibility. Some of the tools are limited by the capability of data transmission, they are unable to perform the monitoring of sensor network with large traffic, therefore the study of system with capability to monitor and analyze larger sensor network is necessary and important.

SNDS is a distributed monitoring and protocol analysis system for complex WSNs. SNDS is based on passive Sniffers co-deployed with the target WSN [3]. The sniffers contact to each other via Ethernet. Sniffer is mainly used to monitor a particular channel and transmit the data to the service program. And then the service program will analyze the data and graphically display the result.

II. BACKGRAND AND RELATED WORK

A lot of the existed tools storage the data in the flash or transmit it by the wireless. The monitor node has good mobility. It is a good solution for some small traffic wireless networks [1]. But the traffic of the sensor network is increasingly large and some applications require real time response, so a tool can handle these requirements is needed. Otherwise the wireless manners maybe jam the target sensor networks [5].

We design the SNDS which has a strong and stable performance. SNDS is based on Ethernet and it can capture WSN data, analyze the data and show the status of a large wireless sensor network in real time. So we think SNDS is a good solution to satisfy the stability, flexibility, real-time needs.

III. *SNDS* ARCHITECTURE

SNDS consists of two main components: a sniffer infrastructure for monitoring WSN and transmitting data to the service program; a service program for analyzing data. SNDS can capture packets and analyze the data in real time. We believe this factor is very beneficial for monitoring WSN and analyzing data. See Fig. 1.

A. *Distrubuted Sniffer*

The core of the SNDS infrastructure is a set of network sniffers which can capture the wireless packets and transmit them to the service program for later analysis. The structure of a sniffer is not complex, included a MCU [10] and two RF chips [9]. However, certain factors must be taken into account to achieve the desired level of fidelity in the design.

First of all, in order to realize the accurate and quickness of data transmission we select Ethernet and use TCP to be the transmission protocol [2]. All the sniffers and the service PC a-

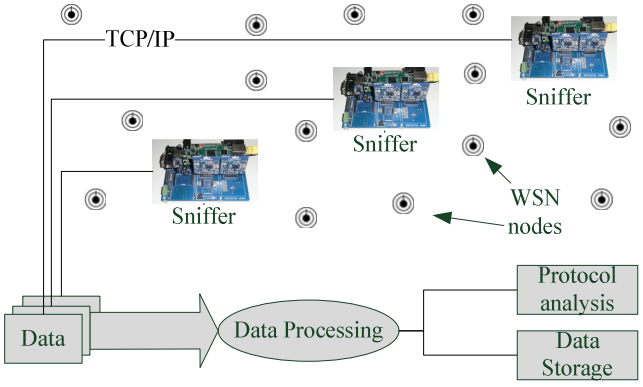

Figure 1. *SNDS* Structure Overview

This program is sponsored by Texas Instrument Innovation Funds.

978-0-7695-4011-5/10 $26.00 © 2010 IEEE
DOI 10.1109/NSWCTC.2010.233

re in the same LAN. Although mobility of SNDS is weaken, transmission reliability and data transmission capabilities have been greatly enhanced.

Sniffers must timestamp packets as they are received by the radio. It is very important to extend the applications of SNDS. We use the 96Mhz timer provided by the STR912 [7], which greatly increase the accuracy of a single sniffer clock. , The time consistency of sniffers which are on a distributed network structure is very important, so we use the network control system IEEE1588 Precision Time Synchronization Protocol (PTP) [6]. This is an agreement for industrial use which can reach nanosecond synchronization precision. PTP Primarily is used to synchronize each node on a distributed network clock. In addition, the agreement is designed for small homogeneous or heterogeneous local area network, designers pay particular attention to the use of resources. So PTP can be used on the low-cost terminal equipment [8].

We chose μC/os-II and Lwip protocol stack. Sniffer contains the wireless packet capture, TCP connections and PTP time synchronization and other tasks. We think TinyOS etc. designed for the wireless sensor networks can not handle such a multi-tasking need. So we chose μC/os-II which is a multi-task real-time embedded operating system.

B. Data Transmission Impletation

Sniffers do not need to configure its IP. They search the service program by UDP and establish TCP connection with the service program.

In order to complete the data transmission, the following tasks are defined:

- *TaskMontior*: a manager task which is used to monitor the sensor network, manage the PTP tasks and data transmission tasks.
- *TaskTcpClient*: a data transmission task.
- *TaskUdpBroadcast*: a server-searching task.
- *TaskUdpReceive*: a receiving task which receives the response of the server.
- *TaskTcpReceive*: a command-handling task.

Workflow: First of all, the *TaskMontior* creates the *TaskTcpClient.* The *TaskTcpClient* creates the *TaskUdpBroadcast* which sends UDP broadcast packets to search the service program and creates *TaskUdpReceive*. After the *TaskUdpReceive* receives the response from the service program, the *TaskTcpClient* establishes TCP connection with the service program and transmits the data. At the same time, the *TaskTcpClient* creates the *TaskTcpReceive* to receive and handle the commands from service program.

C. PTP Impletation

1) PTP typical model

A typical communication model of a PTP system [4] is shown in Fig. 2 The model consists of 15 clock nodes and 4 PTP communication channels. Clock nodes are divided into two categories: ordinary clocks and boundary clocks [6].

Common clock has only one PTP port which connects with a communication channel. In Fig. 2 all clocks are ordinary clocks except the clock 13 and 14. Boundary clock has more than one PTP port and each port connects a different communication channel. In Fig. 2, the clock 13 and 14 is boundary clock. A boundary clock primarily is used to generate a time distribution tree. If PTP message can not go through some gateway node, the node should be boundary clock, and each communication channel has a PTP port. At any time, there is at most one port of the boundary clock in the slave state. The other ports in the communication channels should be in the master state. If all ports of the boundary or ordinary clock are in the master state, then the clock becomes *the most advanced clock* in the entire communication model, it is the root of the time distribution tree. In a PTP system, all the clock nodes can automatically form a time distribution tree through the messaging mechanism among the nodes and the best master clock algorithm.

This is a standard PTP implementation model. In the SNDS, we simplify PTP.

2) PTP implementation model

In the SNDS, all the clocks are in the same LAN, see Fig. 3. Based on this model, the following simplifications are done:

- There are no boundary clocks in the SNDS. All clocks are normal clock. And each clock has only one port which connects to the network.
- All clocks in the SNDS are in the same domain. When SNDS is stable, only one clock would become a master clock, the other clocks synchronize with the master clock, so the master clock is also *the most advanced clock* in this domain. When a more accurate clock is added into system, all clocks would re-select the master clock and the synchronization source.

Although in the SNDS we simplify PTP, all PTP communication and synchronization rules are realized. All the clocks

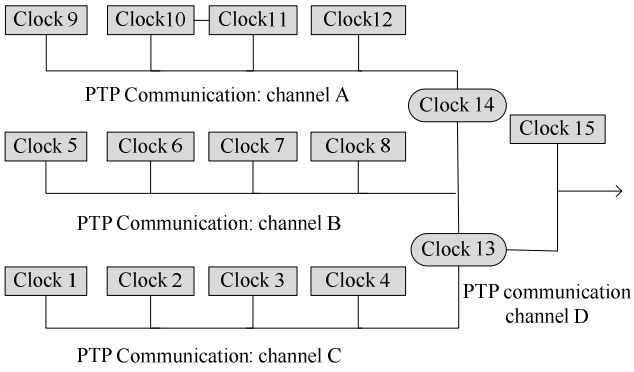

Figure 2. PTP communication model

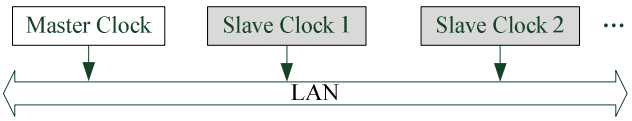

Figure 3. PTP implementation model

don't need join the system in turn. After joining the network, the node can automatically select state and synchronize clock. The message formats are strictly followed with the agreement of PTP. So if other PTP clock with different platform is added to the system, the node is also able to synchronize clock and does not need to change them much.

PTP is a common synchronization protocol, so it doesn't rule the communication medium. There is only one filed of sending message which contains the type of the communication network. So it is not optimized for specific network and applications. In the SNDS, we optimize PTP for our platform.

In order to synchronize time, the following tasks are defined and implementation of the three tasks is shown in Fig. 4:

- *TaskTimeoutHandle*: it is used to deal with a variety of time-out events of the PTP clocks in accordance with the port state.
- *TaskReceivePTPMessage*: most of the time this task will in waiting state. After receiving the PTP message, it will deal with it in accordance with the port state.
- *TaskStateChange*: it is used to deal with the state-changing events of PTP clocks.

The priorities of the three tasks are from low to high, but higher than the task *TaskMonitor* and data transmission tasks.

IV. PROTOCOL ANALYSIS

A. Service Program Structure

The service program structure consists of four modules:

Ethernet monitoring module is a TCP server for monitoring sniffer connections, getting data from the Ethernet and sending configuration messages to sniffers.

MAC layer analysis module is to analyze the received wireless frames in accordance with the standard IEEE802.15.4.

Data processing module is to analyze data field of the frames in accordance with different network protocols and application layer protocols.

Database management module is to storage the processed data.

Configure module is to configure the radio channel of sniffers, network protocols and so on.

B. Protocol Analysis

The main function of the service program is wireless sensor network protocol analysis. The protocol format can be configured as needed to complete specific sensor network analysis

The service program analyzes the data in accordance with the users' protocol configuration. Otherwise service program can be configured to filter specific types of packets, such as filtering to all "Data" packets.

Protocol analysis is completed by two threads. One is the original data processing and analysis thread in accordance with the specific protocol format; the other is the display thread which gets data from the processed packets buffer and graphically displays the packets. See Fig. 5.

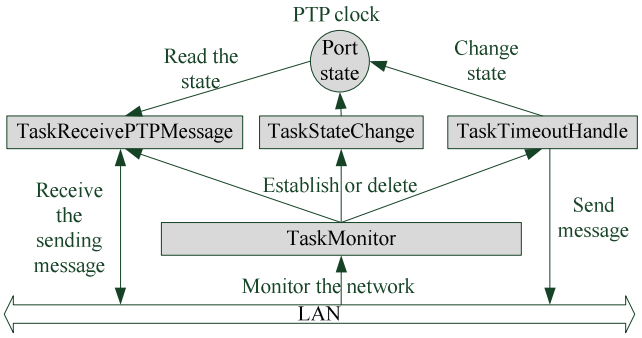

Figure 4. The implementation of PTP tasks

V. SYSTEM EVALUATION

A. Test Environment and Conditions

There are 20 target nodes in target WSN which uses ZigBee protocol. The nodes are randomly deployed in 40 square meters.

The service PC conditions: Intel Core™ 2 Duo CPU E4400 2.00GHZ; 1GB RAM; Windows XP SP3.

B. System Stability Test

We conduct the following tests to evaluate the stability of the SNDS. The focus of the evaluations is the continuous working time and the lost of the transmission data. It should be noted that the CC2420 can not completely capture all the wireless packets, so we just focus on whether all of the captured packets is parsed.

- Test 1: Each node in the target WSN sends 20 packets per second and the fixed-length of each packet is 70 Bytes. We use 3 sniffers to capture the packets. When capturing one packet, sniffer will add a serial number on the packets, so we can know weather the SNDS lost data.

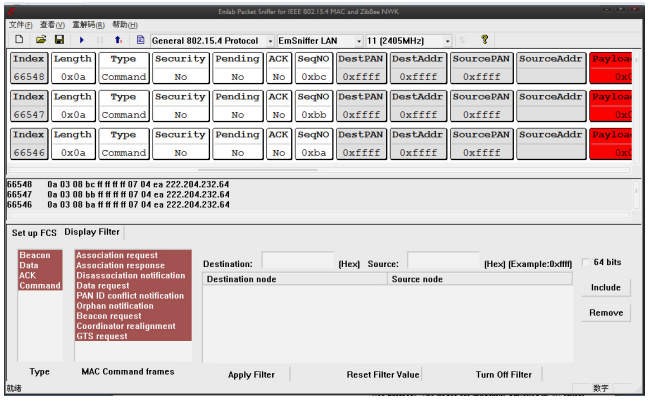

Figure 5. Protocol analysis view

TABLE I. THE RESULTS OF SYSTEM STABILITY TEST

Sniffers	*Nodes*	*Length of Packets*	*Work Time*	*Result*
3	20	70Bytes	13 Hours	stable
3	20	127Bytes	17 Hours	stable

- Test 2: This test method is almost the same with the test 1 except the length of the packets is changed to be the maximum (127Bytes).

C. PTP synchronization Accuracy Test

First, we test the skew between master clock and slave clock, when the slave clock does not modify the local time. It means that PTP does not work.

See Fig. 6, X-axis is stands for the time that PC sends test messages to sniffers, the unit is seconds; Y-axis is stand for the time difference between slave clock and master clock when they received test message, the unit is microseconds. In the test, PC sends a test message every second. The dots is stand for the time difference between slave clock and master clock.

As we can see in Fig. 6, if the slave clock did not modify the local time in accordance with PTP, then the time skew from the main clock would be bigger and bigger as time goes by, this increase was mainly due to the difference of the crystal speeds. Although the offset values increased, the dots showed a certain degree of linear relationship. All the dots were basically in the range of line *L*.

In the SNDS, we set the sniffers synchronize time every 2 seconds. We analyzed 800 samples. See the results in table II.

Results analysis: From the statistic of "＜50us", "≥ 50us" and "Maximum Offset", we can estimate the stability and jitter of time offset after time synchronization. The samples which are less than 50 microseconds is more, the stability of SNDS is better. We set 50 microseconds to be the threshold, because in 2 seconds the time offset (Sync Interval) is estimated to be 20 microseconds according to Fig. 6. But effected by the jitter crystal, hardware sending and receiving messages delays and other factors, the clock time offset may increase more greatly. In Fig. 6, the biggest time offset was 50 microseconds. After

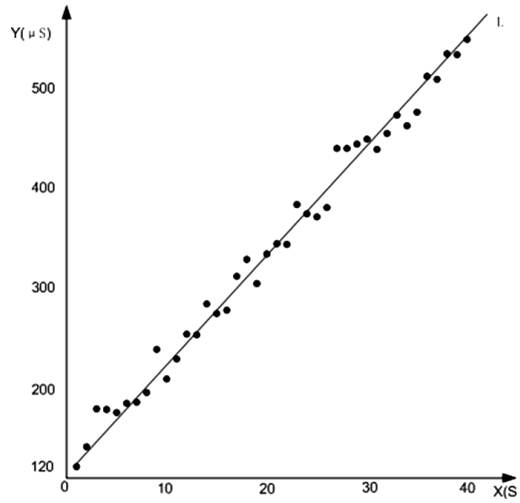

Figure 6. The Time skew between master clock and slave clock without PTP

TABLE II. THE RESULTS OF SYSTEM STABILITY TEST

Samples	*Sync Interval*	*＜ 50us*	*≥ 50us*	*Maximum Offset*	*Synchronization Precision*
800	2s	784	16	90us	17us

the time synchronization if the time offset was greater than 50 microseconds, this should be due to the clock using the PTP to modify the local time.

VI. CONCLUSIONS AND FUTURE WORKS

It is not easy to analyze sensor network information such as the network running status and operating efficiency because of some features of WSN. So the analysis and debugging technology is very important in development of WSN. It is still in the growth and there are many problems which are worth in-depth study and exploration. The research needs propose new and more effective ways to handle those problems.

In this paper we implement a platform for sensor network monitoring and it still has improvement space:

- Synchronization precision can be further improved. In the SNDS the sniffers obtain local time from the timer of STR912 whose time precision is only up to microsecond. The synchronization precision can not exceed the accuracy. If using a more accurate time source, the sniffers can synchronize time more accurately. Otherwise Ethernet MAC chips with hardware support for IEEE1588 are also helpful.
- In the service program more information can be processed and displayed, not only the protocols. Based on the microsecond level time synchronization accuracy, SNDS can provide more information of the behavioral characteristics of sensor networks.

REFERENCES

[1] Bor-rong Chen, Geoffrey Peterson, Geoff Mainland and Matt Welsh, "LiveNet: using passive monitoring to reconstruct sensor network dynamics," IEEE DCOSS, 2008.

[2] Yu Yang, Peng Xia, Liang Huang, Quan Zhou, Yongjun Xu, Xiaowei Li, "SNAMP: A multi-sniffer and multi-view visualization platform for wireless sensor networks," IEEE ICIEA, 2006.

[3] M. Ringwald, M. Cortesi, K. Romer, and A. Vialetti, "Demo abstract: Passive inspection of deployed sensor networks with snif," In K. Langendoen and T. Voigt, editors, Adjunct Proceedings of the 4th European Confererence on Wireless Sensor Networks, 2007, pp 45–46.

[4] John C. Eidson, "Measurement Control and Communication Using IEEE1588," ISBN: 987-1-84628-250-8, 2006.

[5] S.Rost, H.Balakrishnan, "Memento: A Health Monitoring System for Wireless Sensor Networks," In IEEESECON, Reston, VA, Sept. 2006.

[6] Martin T, Jose S, "MOTE-VIEW: A sensor network monitoring and management tool," Proceeding of Embedded Networked Sensors, 2005, pp. 11-18.

[7] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "A system for simulation, emulation, and deployment of heterogeneous sensor networks," In Proc.Second ACM Conference on Embedded Networked Sensor Systems, 2004.

[8] IEEE std. 1588-2002, "Precision Clock Synchronization protocol for networked measurement and control systems," Sept. 2004

[9] STMicroelectronics, STR91xF DataSheet, http://www.st.com, 2006.

[10] Texas Instruments, CC2420 Datasheet, www.ti.com, 2007.