Implementation of a Network-wide Time Synchronization Strategy for WirelessHART

Sheng Zhang, Zhongsheng Xiang, Jie Chen

Graduate School at Shenzhen, Tsinghua University, Shenzhen, 518055, China

Keywords: network-wide time synchronization, WirelessHART, OMNeT++.

Abstract. WirelessHART is an international wireless communication standard proposed by HART Communication Foundation. As a crucial technique for wireless networks, time synchronization plays an important role especially in WirelessHART networks. For constant and reliable communication, it is necessary to provide precise network-wide time synchronization for TDMA-based WirelessHART. The network-wide time synchronization (NWTS) on an open source platform named OMNeT++ was implemented in this paper. In addition, we not only adopted a proposed Kalman-Filtering algorithm to ensure the precise synchronization but also designed a NWTS mechanism for WirelessHART. Moreover, we detailedly designed the mechanism of selecting and synchronizing the time reference sources (TRSs). Finally, we evaluated this implementation and made a comparison to TPSN in the aspect of error, robustness and power consumption.

Introduction

The WirelessHART standard, launched by HART Communication Foundation, is a wireless sensor network communication protocol applying to industrial process control [1-2]. Because of the TDMA mechanism used in the MAC layer [12], node's behaviors are scheduled by a series of super frames. It is important to synchronize the whole network's time so that the message can be transmitted correctly and the network will work on well.

Every node in WirelessHART network has its own local clock. With the surrounding temperature changes and time flies, the frequency of oscillator varies and will result in time deviation between different nodes. If the deviation crosses a threshold, nodes will be divorced from the network. Hence, it's essential to design an effective synchronization mechanism. In WirelessHART standard, however, there's no specification of how to realize the time synchronization of the entire network and how to select the time reference sources.

So far, several time synchronization algorithms have been proposed for wireless sensor network, such as RBS (Reference Broadcast Synchronization) [3], FTSP (Flooding Time Synchronization Protocol) [4], TPSN (Time Synchronization Protocol for Sensor Networks) [5], DMTS (Delay Measurement Time Synchronization) [11], etc. However, these algorithms are not suitable for WirelessHART network considering of limited energy, precise time and mesh topology. De Biasi etc. [6] researched the clock drift of WirelessHART on TrueTime which is a matlab-based tool. However, they did not implement the entire protocol stack nor the whole network time synchronization. Huang etc. [7] utilized the OMNeT++ simulator to study the 802.11 network time synchronization. However, they just synchronized two nodes rather than realize network-wide synchronization. Fang promoted an energy-efficient time synchronization algorithm for WirelessHART networks based on OMNeT++ [9]. However, they compared their algorithm with RBS neglecting the deviation of frequency and never considered the network-wide synchronization.

In the open source OMNeT++ simulator [8], we implemented the whole WirelessHART network including the network manager and the node model. Moreover, we not only designed the local clock model and the packet exchange model in detail but also implemented the network-wide time synchronization applicable to the mesh network. We adopted a precise Kalman-Filtering synchronization algorithm and designed a network-wide time synchronization mechanism for WirelessHART network. Finally, some simulating cases were tested to evaluate this implementation.

The rest of the paper is organized as follows: Section 2 depicts the local time model and packet exchange model. Section 3 describes the details about how to implement network-wide time synchronization (NWTS). Section 4 simulates NWTS and compares it to TPSN. Section 5 concludes this paper.

Design of the time synchronization model

Local clock model. The node's time is usually generated from crystal oscillator which is greatly influenced by manufacturing process and surrounding environments. Generally speaking, clock frequency is different from the normal value that will result in deviation of offset and skew. In OMNeT++ simulator, we utilize the local clock model to represent the real clock in nodes.

i) Common node's clock model

Let t and $f_j(t)$ denote the accurate reference time and clock j's frequency at reference time t. $C_j(t)$ is defined as the local time of nodes at the reference time t and can be expressed as:

$$C_j(t) = \int_{t_0}^{t} f_j(t)dt + C_j(t_0) \tag{1}$$

We can think that the frequency is stable in a short time. Then, $C_j(t)$ should be updated as:

$$C_j(t) = f_j(t_0)(t - t_0) + C_j(t_0) \tag{2}$$

ii) Network manager's clock model

Network manager is the root node in the WirelessHART network and is responsible for the network-wide time service. So, all nodes should be synchronizing to network manager and it's clock model $C_{NM}(t)$ can be expressed as:

$$C_{NM}(t) = t \tag{3}$$

Package exchange model. In our simulator, we adopted the Kalman-Filtering method proposed by paper [16] to synchronize *node i* with *node j* based on two-way message exchange. Kalman-Filtering technology just needs to record the previous estimated value and the Kalman-Filtering gain. The current estimated value can be calculated according to the current measured value. There is no need to record all the historical data. This algorithm regards the clock offset and the frequency deviation as the state variables that include noise. Then the optimal estimation of the clock offset and frequency deviation can be conducted by the Kalman-Filtering method. As is illustrated in Fig. 1, in the nth synchronous cycle, *node j* sends a synchronous request to *node i* at the reference time $t1$ and *node i* receives the timestamp at time $t2$. Then, *node i* sends a response message embedded with the estimated time lag to *node j*. *Node j* catches the packet at time $t4$ and records the local clock's time $C_{L,j,n}(t_4)$. According to paper [16], the Kalman gain can be calculated as:

$$K(n) = \frac{P(n \mid n-1)}{P(n \mid n-1) + R} \tag{4}$$

Therefore, the optimal estimation of time offset should be expressed as:

$$\hat{x}(n \mid n) = \hat{x}(n \mid n-1) + K(n)[y(n) - \hat{x}(n \mid n-1)] \tag{5}$$

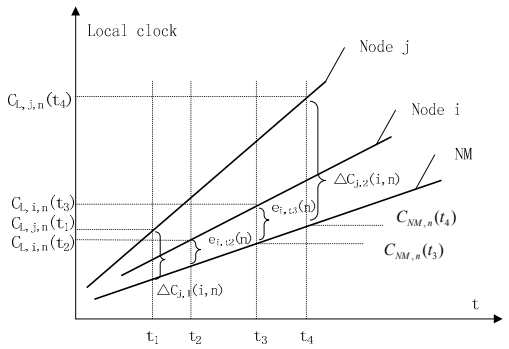

Fig.1: Local clock and packet exchange model

Design of network-wide time synchronization

The NM needs to coordinate the synchronization only of a few TRS nodes, and all the other nodes synchronize with the TRSs privately. In this section, we implemented the network-wide time synchronization by selecting the time reference sources (TRSs) and synchronizing these TRSs. **Selecting the TRSs (time reference sources).**TRS selecting mechanism is put forward dynamically while the network construction is in progress. Detailed steps are as follows: Firstly, network manager is configured as a TRS while starting the network. Secondly, one node is going to join the network by selecting a node agent. If there is a TRS in the node's neighborhood, the node selects the TRS as the node agent. Subsequently, the node joins the network by synchronizing to the node agent. If the node won't receive any broadcast information from a TRS for a time, the node will select any neighbor node as a node agent and synchronize to the node agent which is regarded as a TRS temporarily. Then, the node will apply for TRS while requesting to join the network. Lastly, other nodes will utilize these steps to join the network until every node finishes this process. A set of brief procedures are depicted in Fig. 2.

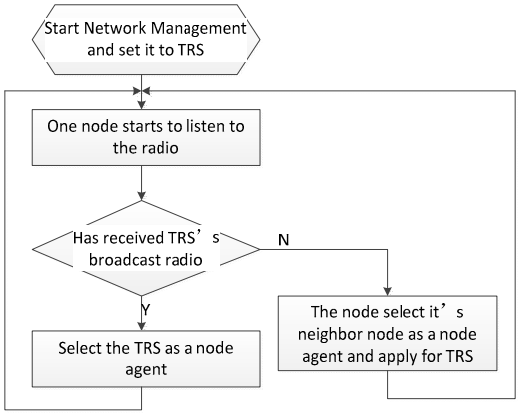

Fig.2: Process of selecting TRSs

Synchronizing the TRSs. After all the TRSs are determined, it is necessary to synchronize these TRSs. Network manager starts time synchronization between the TRSs every once in a while. It's easy to prove that there exits at least one common neighbor node between any pair of TRSs. So we can utilize the common neighbor node to synchronize TRSs. Assuming *a* and *b* are a pair of TRSs, node *c* is their common neighbor node. We can use <*a, c, b*> to represent this relationship and select *a* to be the master node. According to Fig. 3, this algorithm selects the top stack element as the TRS (*x*), and then traverses the set *A* to find out all the nodes that have common neighbor node with *x*. If there is no matching node, the algorithm selects the element which is below the top stack element. For handling pair <*x, j, i*>, we firstly let *j* synchronize with *x* and then synchronize *i* to *j*.

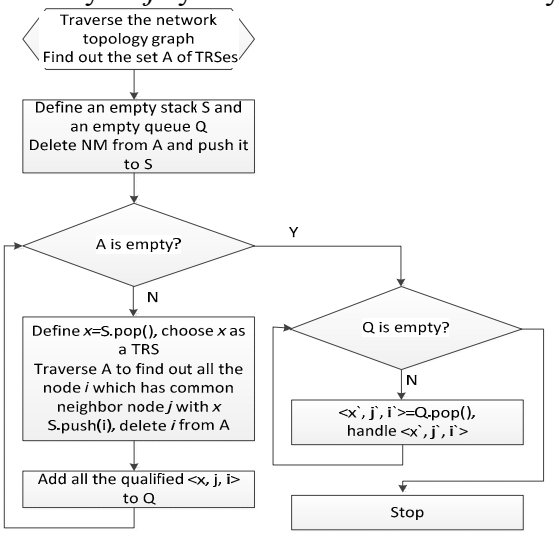

Fig.3: Process of synchronizing TRSs

Testing the NWTS mechanism. Time synchronization algorithm is running in the network manager and the data link layer. In this test, there are 8 WirelessHART nodes and a NM (network manager). Fig. 4 illustrates the process of selecting TRSs. Firstly, NM (network manager) starts and configures itself as a TRS. Secondly, *node 1, 2* select NM as their node agent to synchronize with and join the network. Thirdly, when *node 3* starts, there is no TRS around. It randomly selects *node 1* or *node 2* as a node agent and later applies to the NM for becoming a TRS. Fourthly, *node 4* and *node 5* select *node 3*. *Node 6* randomly selects *node 4* or *node 5* and applies for TRS. Finally, *node 7* and *node 8* use the same steps to synchronize to the TRS and join the network.

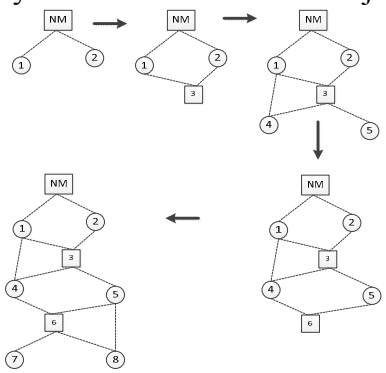

Fig.4: Test of NWTS

In the above section, we implement the selection of TRSs. Next it is time for the synchronization between TRSs. Traversing the network topology graph, we can conclude that *A= {*NM*, 3, 6}* according to Fig. 4. NM is deleted from *A* and pushed on a stack *S*. Then *A= {3, 6}* and isn't empty. So traverse *A* to find out qualified *3* with common *nodes 1, 2*. Furthermore, *3* is pushed on *S* and deleted from *A*. Qualified pair <NM*, 1, 3*> is added to the queue *Q*. Then, *A= {6}*, and *S.pop()= 3* . Next we can find out the qualified pair <*3, 4, 6*> and add it to *Q*. Finally, Kalman Filtering method is used to handle these two pairs.

Test the implementation

In this section, we implemented the WirelessHART emulation system. On this basis, the above network-wide time synchronization mechanism is simulated. We compare this algorithm with TPSN in the aspect of synchronization error, network robustness and power consumption. Parameters of the simulation scenario are configured in table 1.

Table 1: Parameters of the scenario

Para. type	Value	Para. type	Value
Simulation area	200 ×250	Pass loss coefficient	2
Node numbers	60	Carrier frequency	2.4GHZ
Max transmit power	20mW	Channel numbers	15
Attenuation threshold	-110dBm	Propagation model	RiceModel

Fig. 5 demonstrates the synchronization variance varies with different hops. It is obvious that synchronization variance of TPSN grows linearly with the hop increasing. Particularly in the tenth hop, the variance reaches 30 units. On the contrary, variance of NWTS is stable and maintained at a low stage. Especially at the tenth hop, the value is about 7 units that are far below 30 units.

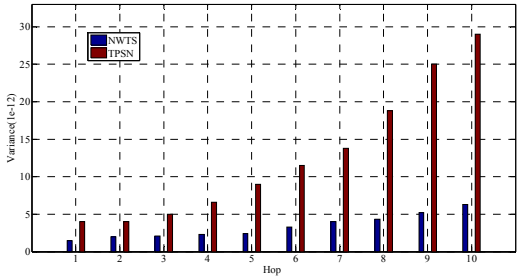

Fig.5: Error with multi hops

Because of the influence of channel quality, packet loss is common in the process of transmitting data. If the synchronization error reaches the max threshold, there is a high possibility that synchronization will fail. Furthermore, it will lead nodes unable to communication normally and fail to connect with the network. Supposing that the clock jitter is 42ppm and synchronous cycle is 5s, the clock offset will reach above 200us once the synchronization fails. Assuming the probability of transmission failure is 0.1% and Kalman method is not adopted， the probability of breaking away from the network in the nth cycle is:

$$p(n) = 10^{-3} * 0.999^{n-1} \tag{6}$$

If we adopt the Kalman method, the probability of breaking away from the network in the *n*th cycle is:

$$p(n) = 10^{-6} * 0.999^{n-2} \tag{7}$$

We can see the network reliability increases 999 times. Considering a harsh environment, packet loss rate is 5%. Meanwhile the node will drop off the network once synchronization fails because of packet loss. As Fig. 6 shows, NWTS using the Kalman predicting method performs better than TPSN with respect to reliability and robustness.

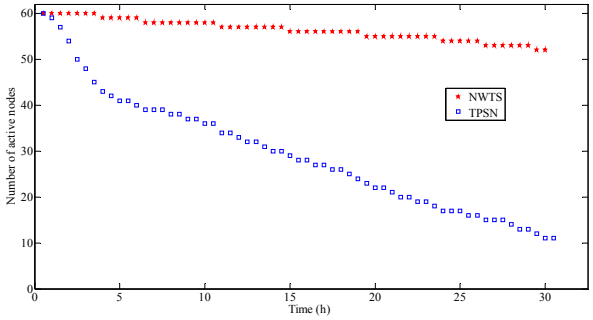

Fig.6: Robustness of the network

According to Kalman algorithm, local clock's frequency will be close to the reference clock after long time of frequency compensation. As a result, clock offset will be decreased over time. Therefore, in order to reduce the energy consumption of time synchronization, it is important to lengthen synchronizing cycle appropriately. We use a self-adapting method to adjust the time synchronizing cycle. As is depicted in Fig. 7, α is the adjusting coefficient of the synchronizing cycle, β is the judging coefficient, θ is the number of the synchronizing cycle, △ C(n) is the time deviation in the *n*th synchronization, Max_Sycn_Num is the max number of synchronization and Max_Sycn_Period is the max value of synchronizing cycle. The node can adjust the synchronizing cycle according to this algorithm.

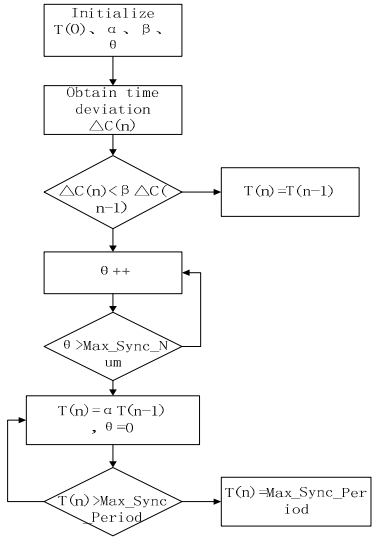

Fig.7: The self-adapting algorithm

We set α=1.01,β=0.9 and θ=6. As Fig. 8 illustrates, the power consumption of TPSN is constant and stays at a high level. For NWTS, the self-adapting algorithm lengthens the synchronizing cycle

and reduces the power consumption of synchronization. After 900 cycles, the synchronizing cycle and the power consumption tend to be stable.

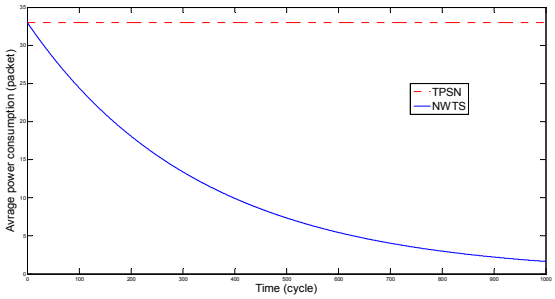

Fig.8: Power consumption

Conclusions

In this paper, OMNeT++ simulator is developed for implementing and simulating the time synchronization of WirelessHART. In addition, a precise Kalman-filtering synchronizing algorithm is adopted and a network-wide synchronization mechanism is put forward specially for the mesh network of WirelessHART. This paper offers an efficient way to analyze and evaluate the performance of network-wide time synchronization. The results also show that NWTS performs better than TPSN with respect to error, robustness and power consumption.

Acknowledgements

The research work was supported by the science and technology project sponsored by Xinjiang, China. Project number is 201312107.

References

[1] HCF SPEC 290 Revision 1.0, Wireless Devices Specification[S].

[2] Reaves, B., & Morris, T. Analysis and mitigation of vulnerabilities in short-range wireless communications for industrial control systems. *International Journal of Critical Infrastructure Protection*, 5(3), pp.154-174, 2012.

[3] Elson, J., Girod, L., & Estrin, D. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36(SI), pp.147-163, 2002.

[4] Maróti M, Kusy B, Simon G, et al. The flooding time synchronization protocol. *Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM*, pp.39-49, 2004.

[5] Ganeriwal S, Kumar R, Srivastava M B. Timing-sync protocol for sensor networks. *Proceedings of the 1st international conference on Embedded networked sensor systems. ACM*, pp.138-149, 2003.

[6] De Biasi M, Snickars C, Landernäs K, et al. Simulation of Process Control with WirelessHART Networks Subject to Clock Drift. *COMPSAC*, pp.1355-1360, 2008.

[7] Huang Y, Yang Y, Li T, et al. An Open Source Simulator for IEEE1588 Time Synchronization over 802.11 Networks. *Modelling Symposium (EMS), 2013 European*, pp. 560-565, 2013.

[8] OMNeT++. http://www.omnetpp.org/

[9] Fang M, Sun Y, Zhang X, et al. ELOTS: Energy-efficient local optimization time synchronization algorithm for WirelessHART networks. *Systems Conference (SysCon), 2014 8th Annual IEEE*, pp.402-406, 2014.

[10] Song J, Han S, Mok A K, et al. A study of process data transmission scheduling in wireless mesh networks. *ISA EXPO Technical Conference*. 2007.

[11] Ping S. Delay measurement time synchronization for wireless sensor networks. *IRB-TR-03-013, Intel Research Berkeley Lab*, 2003.

[12] Gao G, Zhang H, Li L. An OPNET-based simulation approach for the deployment of WirelessHART. *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on.IEEE*, pp.2120-2124, 2012.

[13] Zand P, Dilo A, Havinga P. Implementation of WirelessHART in NS-2 simulator. *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*. *IEEE*, pp.1-8, 2012.

[14] Mo Y, Liu Z, Zheng L, et al. Kalman-consensus filter for time synchronization in wireless sensor networks. *Information and Communications Technologies (IETICT 2013), IET International Conference on. IET*, pp.421-428, 2013.

[15] Zeng, Y., Hu, B., & Feng, H. (2007, August). Time division flooding synchronization protocol for sensor networks. *In Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on . IEEE*, pp.1-8, 2007.

[16] Sheng Z, Xuanzhao W, Dongdong L. An Improved Low Power Time Synchronization Algorithm for WirelessHART Network. *Computer Modelling and Simulation (UKSim), 2013 UKSim 15th International Conference on. IEEE*, pp.672-676, 2013.