DREET: A Data-Reliable, Energy-Efficient Transport Layer Protocol

Gang Han
College of Computer Science, National University of Defense Technology, NUDT, Changsha, P. R. China
Email: ganghan@nudt.edu.cn

Jia Lu, Xinbiao Gan, Huanzhong Li and Wenhua Dou
College of Computer Science, National University of Defense Technology, NUDT, Changsha, P. R. China
Email: {lujia, xbgan, lihuanzhong, whdou}@nudt.edu.cn

Abstract—**In order to achieve reliable event detection, wireless sensor networks (WSN) should guarantee the reliable transport of sensor data from sensor nodes to the sink. So it is very important to keep a limited deviation between sensor data (the data sensed by sensor node) and sink data (the data delivered to applications). We put forward a data reliability notion to depict this limited deviation in this paper. Noticing that sensor data vary slightly in relative short time unless certain events occur, we construct a small subset of sensor data as key data, and provide reliable transport only for these key data instead of for all sensor data. This solution decreases retransmission and energy while keeping data reliability. Based on the data reliability notion, we propose DREET, a data-reliable energy-efficient transport layer protocol. Simulation results show that DREET provides an optimal event detection rate and relative standard deviation (RSD) between sensor data and sink data, by introducing a tiny increase to energy consumption (no more than 1.97% in our simulation).**

Index Terms—**wireless sensor networks, reliable transport layer protocol, data-reliable, energy-efficient**

I. Introduction

Wireless Sensor Network (WSN) generally consists of tens or thousands of sensor nodes and one or several sinks (or base stations). The sensor nodes can sense physical information and transmit them to the sink [1]. WSN can be used for many applications such as habitat monitoring, in-door monitoring, target tracking, security surveillance, etc. [2]. In order to support these applications, it is necessary for WSN to supply reliable communications between sensor nodes and the sink. The functionalities and design of a reliable transport solution for WSN are the main issues addressed in this paper.

In the traditional wired networks, TCP is adopted to provide an end-to-end reliable transport. However, S.Ganesh et al. [3] elaborate the drawbacks of TCP, such as large overhead, poor throughput and energy inefficiency, which make it unsuitable for WSN.PSFQ [4] may be the earliest transport protocol for WSN. It requires each intermediate node to create and maintain a data cache for local loss recovery, and guarantees strict end-to-end reliable data delivery. However, it consumes too much buffer and energy, both of which are severely limited in WSN.

Unlike the strict reliability in PSFQ, STCP [5] provides variable reliability. In other words, it is tolerant for a small proportion of packets to be lost in STCP. Reliability is a measure of the fraction of packets that are successfully received. The sink determines whether to inform the source node to retransmit a lost packet according to the current reliability.

Event-To-Sink Reliable Transport (ESRT) [6] is developed for reliable event detection. The sink computes the reliability factor by a ratio of the number of data packets received from the source nodes to that required by the application. And according to this ratio, the sink predicts the suitable reporting frequency of the source nodes in the next interval. ESRT is concerned with the collective reliability of all the data flows in an event-detection process, instead of reliability of each individual flow.

PSFQ and STCP focus on the reliability of each packet, while ESRT pays more attention to the number of packets received from related source nodes. However, what concerns the sink most is only the sensor data. Hence, in this paper, we put forward a *data reliability* notion at transport layer. This is a truly novel aspect of our work and is the main theme of the proposed Data-Reliable, Energy-Efficient Transport (DREET) Protocol for WSN.

Such a notion (*data reliability*) means that the deviation between the data sensed at sensor nodes and those delivered to application layer at sink should be constrained to a limited range (they may not be the same due to packet loss caused by the unreliable channel).

A. Cerpa et al. observe that a large amount of temporal correlation exists among habitat monitoring data [7]. In fact, in most scenarios, the data of sensors (light, sound, temperature, seismic, etc.) vary slightly in a relatively short time (several or tens of seconds), unless some event occurs. By leveraging this observation, DREET achieves high data reliability with minimum overhead and energy expenditure.

Manuscript received July 14, 2011; revised September 20, 2011; accepted December 30, 2011.
This work was supported by Reliable Wireless Sensor Network Program of National University of Defense Technology.

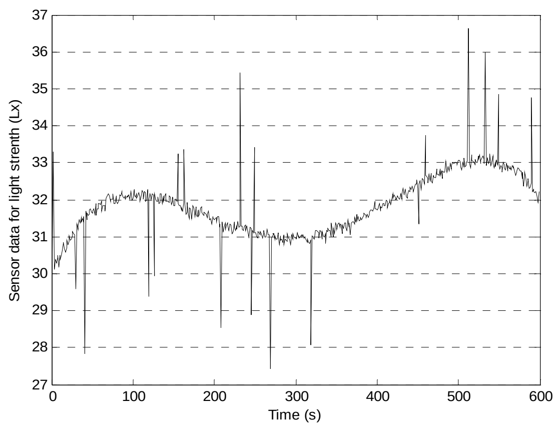

Figure 1. Sensor data for light strength (in Lux) during 600 seconds.

II. Data-Reliable Transport in WSN

We studied application of event detection in WSN, and found that it is more important to ensure the reliability of the event information implied by sensor data than the sensor data themselves. We select a subset of sensor data—called *key data*—to represent the event information, and transport them in a strictly reliable way. The subset is only a small proportion of sensor data, so the overhead of retransmission and acknowledgement is very low. We take *detection rate* and *RSD* (relative standard deviation) to characterize the data-reliability of our solution.

A. Problem Definition

Consider typical WSN application of event detection based on the report of sensor nodes observing certain type of event. However, data got by the sink is not always the same as that transmitted by sensor nodes, as a result of packet loss. Now we give the definition of the two different types of data.

Definition 1. Sensor data represent the data sensed by sensor nodes and transmitted to the sink.

Definition 2. Sink data represent the data delivered to the application of sink. Here, the lost data have been recovered by retransmission or numerical method.

In preceding discussions, we got a conclusion that sensor data varies very slightly in a relatively short time. So chances are that the sink can recover the lost data (caused by packet loss) with a little deviation. And this is achieved by numerical method instead of retransmission, which improves the energy efficiency and congestion of WSN.

However, sensor data do not always vary smoothly. It goes up or down rapidly when a certain event occurs (as in Fig. 1), and maybe that event is just what the sink detects. As a result, this kind of data is of greater importance than others, and we call it the *key data*. We guarantee that all *key data* must be received by the sink. If a packet with *key data* is not acknowledged in a given time, the sensor node will retransmit it. Note that, the quantity of *key data* is much less than that of other data, so the acknowledgement (ACK) and retransmission of packets with *key data* will consume only tiny proportion of total energy and traffic. *Key data* is now defined recursively as follows:

Definition 3. The first sensor data of each event-detection process is *key data*. For a given α (we call it *variation factor* in this paper), if the current sensor data is (1+α) times greater or (1-α) times less than the latest *key data*, it is a new *key data*.

Given a sensor data sequence $S = \{s_0, s_1, \ldots, s_n\}$, the *key data* sequence *S'*, will be a subsequence of *S*. At first, *S'* is empty. Then let s_0 be a *key data*, and add it to *S'*. Let *i* go from *1* to *n*, if s_i is (1 + α) times greater or (1 - α) times less than the last element of *S'*, add s_i to *S'*.

For example, there is a sensor data sequence: $S = \{30, 31, 32, 29, 35, 31, 33, 24, 29\}$, and α = 0.1, then we get a *key data* sequence: $S' = \{30, 35, 31, 24, 29\}$.

In Fig. 1, if we let α = 0.02, there are only 43 elements in the *key data* sequence, while the sensor data sequence has 600 elements. In other words, we just have to provide reliable transport for 43 packets (assume that there's one sensor data per packet).

Note that the value of the *variation factor* (α) directly affects the number of *key data,* which is related to the amount of transmission and energy consumption. It is an evident trend that a large α leads to less energy consumption and vice versa. However, too large α may increase the deviation between sensor data and sink data, while a too little α may raise the number of retransmitted packets. An optimal α is decided by both the energy constraint and requirement of application, and we have to make a tradeoff.

B. Evaluation Methodology

As is referred before, there will be a deviation between sensor data and sink data. To give a quantitative evaluation, we have to define the measurement of this deviation first. In this paper, we mainly focus on the event detection and event features. So, it is reasonable to evaluate our solution based on *detection rate* of events and *RSD* (relative standard deviation). The definition of the two measurements is shown below:

Definition 4. Detection rate represents the probability that application detects an event successfully. And we consider it as the number of events detected by sink to the number of events occurring at sensor node.

Definition 5. RSD represents the relative standard deviation between sensor data and sink data. Here the lost part of sink data has been recovered in numerical method.

The events occurring at sensor node can be observed from sensor data, assuming that every event is accompanied with a steep change of sensor data. As depicted in Fig. 1, there are 19 steep changes observed, and we consider them as 19 events. The same means can be applied in counting the number of events detected by the sink. And then, we get the *detection rate*. We can calculate the *RSD* by routine means as how relative standard deviation is calculated and it is shown below.

$$RSD = \sqrt{\sum \left(\left(D_{sink} - D_{sensor} \right) / D_{sensor} \right)^2 / N}$$

```
DREET_Send(current);
    current_packet = DREET_Encapsulate(current);
    if(current > last_key_data * (1 + alpha) or
      current < last_key_data * (1 - alpha))
            current_packet->ACK_flag = '00';
            add current_packet to key_data_list;
            if(no timer has been set)
                    settimer(timeout_value);
            last_key_data = current;
    else
            current_packet->ACK_flag = '01';
    send(current_packet);

CallWhenTimeOut();
    find out the timeout packet pkt;
    send(pkt);
    settimer(timeout_value);

OnReceivingACK();
    record the acked packet pkt;
    remove pkt from key_data_list;
    if(the timer isn't directed to the first packet in
key_data_list;)
            reset timer and direct it to the first packet in
key_data_list;
```

Figure 3. The transmitting process of *key data* and ordinary data at sensor side.

Here, D_{sensor}, D_{sink} and N represent sensor data, sink data and the number of sensor data respectively.

Besides, the energy consumption of sensor nodes draws much attention, as well. A good transport layer protocol should not introduce too much energy consumption. In this paper, we focus on the energy consumed by communication. We setup experimental environment and set reasonable values to parameters. We take the average energy consumption of each sensor node to evaluate energy efficiency of our protocol.

III. DATA-RELIABLE, ENERGY-EFFICIENT TRANSPORT LAYER PROTOCOL

Based on the notion of *data reliability*, we propose DREET (Data-Reliable, Energy-Efficient Transport Layer Protocol), a novel solution to address the transport problem in WSN. The primary motive of DREET is to achieve high *detection rate* and low *RSD* with minimum energy consumption, and to help accomplish this, DREET provides reliable transport just for *key data,* instead of for all sensor data.

In order for the sink to distinguish the packets with *key data* from those with ordinary data, a two-bit *ACK_flag* is set to the header of DREET packets to indicate packets with *key data* ('00'), ordinary packets ('01') and ACK packets ('10'). As shown in Fig. 2, the function *DREET_Send()* depicts the main transmitting process. Each time a sensor node gets the sensor data, it first

```
DREET_Receive(packet);
    add(packet, recv_buf);
    if(packet->ACK_flag = '00')
        send ACK to source sensor node;
        if(packet->key_No = expected_key_No)
            last_pkt = Last_consecutive_key_data(packet->
                        key_No, recv_buf);
            expected_key_No = last_pkt->key_No + 1;
            Numerical_Recover(recv_buf, last_pkt->SeqNo);
            Deliver_to_app(recv_buf, last_pkt->SeqNo);

Last_consecutive_key_data(last_No, recv_buf);
    while(key_pkt whose key_No is last_No+1∈recv_buf)
            last_No = last_No+1;
    return key_pkt;

Numerical_Recover(recv_buf, last_SeqNo);
    prev_pkt = Headof(recv_buf);
    for i = prev_pkt->SeqNo to last_SeqNo
            if(pkt whose SeqNo = i isn't in recv_buf)
                    pkt = prev_pkt;
                    pkt->SeqNo = i;
                    add pkt to recv_buf;
```

Figure 2. The receiving process of *key data* and ordinary data at sink side.

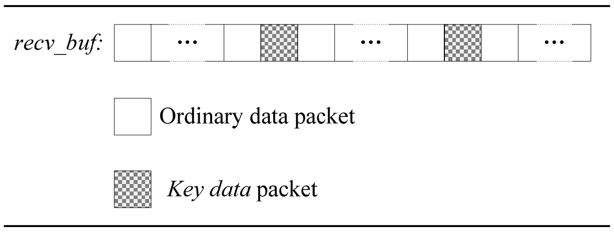

Figure 4. The structure of receive buffer (*recv_buf* in Fig. 2)

encapsulates the sensor data in a DREET packet, then judges if the current sensor data is a *key data* by a given *variation factor* (α) and set the corresponding value to *ACK_flag* ('*00*' for *key data* and '*01*' for ordinary packets). At last, it transmits the packet to the sink. A variable named *last_key_data* is referred in this function. It always registers the latest joined *key data*, so it should be updated each time a new *key data* is added to *key_data_list*, a list recording all the *key data* that has been transmitted but not yet acknowledged.

A sensor node initializes a timer each time it transmits a packet with *key data*. If an ACK to this packet is received in a given time (*timeout_value* referred in Fig. 2), the packet will be removed from *key_data_list* and the timer will be reset and redirected to the first packet in *key_data_list*. Otherwise, if no ACK arrives before the timer times out, the sensor node retransmits the packet and reset the timer at the same time. The function *CallWhenTimeOut()* and *OnReceiveACK()* in Fig. 2 show what DREET does under different cases. DREET adopts a similar mechanism as TCP timeout counter, and sets only one timer for all transmitted *key data*. The timer is always directed to the earliest transmitted packet in *key_data_list*. DREET won't redirect the timer until a corresponding ACK is received. Then the first packet is removed from *key_data_list*, and the timer is redirected to the second packet which is the current earliest transmitted packet in *key_data_list*. The choosing of timeout value has been thoroughly studied in [5].

Fig. 3 shows the receiving process of *key data* and ordinary data at sink side. As depicted in the function *DREET_Receive(),* when the sink receives a DREET packet *packet*, first it adds *packet* to the receive buffer (as depicted in Fig. 4). Then it looks over the *ACK_flag* of *packet*. If this is a packet with *key data*, the sink constructs an ACK packet and replies it to the source. And if this *key data* packet is an expected one (i.e., it's the succeeding *key data* of the last one of the previously delivered packets), we determine which packets should be delivered. We achieve this by the function *Last_consecutive_key_data()*, which returns a packet *last_pkt*, and ensures that all *key data* packets between *packet* and *last_pkt* have arrived, and the succeeding *key data* packets of *last_pkt* hasn't arrived. Next, the lost sensor data between *packet* and *last_pkt* are recovered by function *Numerical_Recover()*. The sink recovers the lost data by numerical method. Here, we just take the previous piece of sensor data, i.e. the one sensed just before the lost one, as the lost sensor data.

DREET requires only the generic functionalities of lower protocols, and it can work in cooperation with most WSN routing protocol supporting communication from sink to sensor node. So it is of great applicability.

IV. Congestion Control

Congestion occurs in WSN when the packet-arrival rate exceeds the packet-service rate, or link-level performance decreases due to contention, interference, and high bit-error rate [1]. Various techniques have been proposed in the wireless and sensor network literature on congestion control. They deal with this problem mainly in three steps: congestion detection, congestion notification, and rate adjustment.

Most papers detect congestion in intermediate nodes by current network metrics such as queue length [11, 16], packet service time [12], etc. In RCRT [13], a centralized congestion control solution is adopted, that is, the sink decides that the network is congested if the time to repair a loss is significantly higher than a round-trip time. After detecting congestion, the congestion information can be transmitted in an explicit way (by special control messages) or an implicit way (by piggybacking congestion information in normal data packets), to notify the involved sensor nodes of congestion.

Upon receiving the congestion information, sensor nodes adjust the transmission rate in different strategy, such as additive increase multiplicative decrease (AIMD) in [14], and some other rate adjustment according to current situation of congestion in WSN [15]. Adaptive Rate Control (ARC) [17] adjusts its transmission rate by overhearing the packet forwarding of its parent node, increases the transmission rate followed by successful forwarding and vice versa, instead of congestion detection and notification.

TABLE I.
Simulation Parameters

Area of sensor field	100x100 m^2
Number of sensor nodes	16~100
Receive sensitivity	-95 *dBm*
Modulation type	PSK
Transmit Power	57.42 *mW*
Receive Power	62 *mW*

However, none of these schemes take account of sensor data, as a result, when the transmitting rate decreases, the key data may be ignored, and accordingly, the event detection rate will be significantly affected.

For a sensor node, the sampling rate is usually a constant value, i.e. the sensor data are produced in an invariable rate, no matter how the transmission rate changes. Thus, when the transmission rate decreases below the sampling rate, the sensor node should do something to ensure the *key data* won't be ignored.

DREET take consideration of the importance of *key data* again in rate adjustment. Assume that, the rate of *key data* is much less than the transmission rate. This is reasonable because *key data* take up only a little proportion of sensor data. In DREET, a sensor node transmits the *key data* immediately once it gets one. When the current transmission rate is larger than given, the sensor node stops the transmission until another piece of *key data* comes, or the given transmission rate is satisfied.

DREET needs the support of intermediate sensor nodes when congestion occurs. Each of the intermediate sensor nodes can identify *key data* packet by *ACK_flag* of the packet it receives ('00' indicates a *key data* packet). Once a *key data* packet arrives at a full receive queue, DREET discards one of the ordinary data packets in the queue to free adequate memory for the *key data* packet. By doing this, the loss of *key data* packets is reduced, and the retransmission decreases accordingly, which, to some extent, alleviates the congestion.

For WSNs using CSMA-like Medium Access Control (MAC) protocols, it is strongly recommended to reduce the maximum number of retransmission of ordinary data packets. If this value is exceeded, discard the packet. The maximum number of retransmission of *key data* packets should be kept unchanged. These modifications can efficiently debase the congestion caused by contention. But cross-layer optimization is involved, which complicates the design of WSN protocols

V. Simulation Results

To evaluate the performance and energy efficiency of DREET, we implemented this protocol and developed an

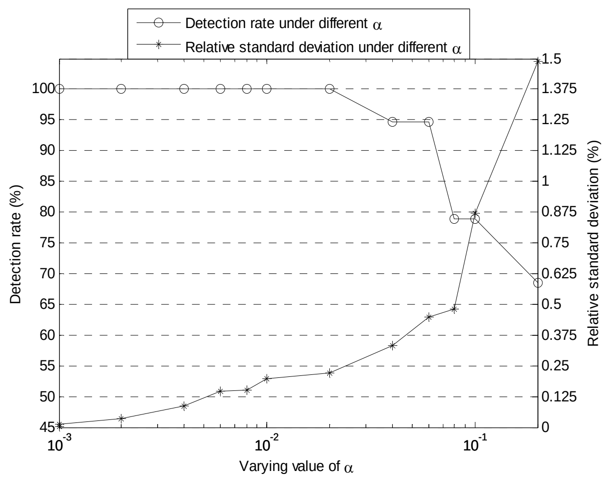

Figure 5. The effect of varying the value of α on detection rate and RSD (relative standard deviation), both denoted in percent.

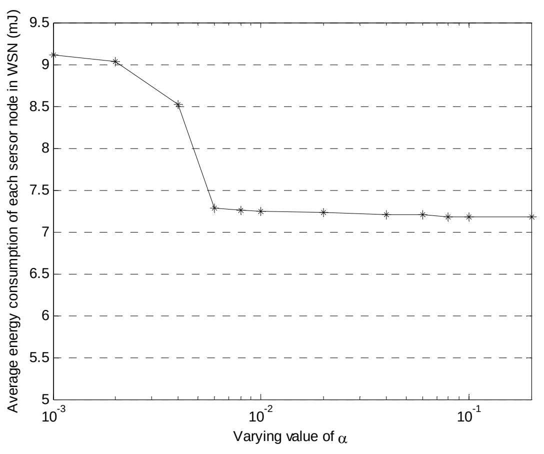

Figure 6. Average energy consumption of each sensor node for varying α.

evaluation environment using Castalia-3.2 [8], a simulator for WSN based on the OMNET++ platform [9].

A. Simulation Setup

As listed in Table 1, we simulated networks with 16~100 sensor nodes randomly distributed in a 100m X 100m square sensor field. To transmit packets from sensor nodes to sink, Multi-path Rings Routing and T-MAC modules are provided by the simulator [10]. Simulations were run for 600 simulator seconds. And during simulating, sensor nodes in the network transmitted a packet every simulator second to sink to report sensor data.

B. Detection Rate and RSD for Varying Variation Factor

We fixed the number of sensor nodes to 100, and performed simulations for varying α (from 0.001 to 0.2) in the environment detailed in 4.1. *Detection rate* and *RSD* (Relative standard deviation) for varying α is depicted in Fig. 5. We find that *RSD* is increasing when α increases. However, it is still less than 1.5% even when α = 0.2. Although *detection rate* keeps to 100% when α < 0.02, it decrease rapidly after α exceeds 0.02, and it reaches 68% when we take 0.2 as the value of α.

Fig. 6 shows the average energy consumption of each node for varying α. We see that the average energy keeps around 7.2mJ when α > 0.006, but the average energy increases remarkably when α falls below 0.006, and it goes up to 9.113mJ when α = 0.001.

From these simulations, we find that the suitable value of α ranges from 0.006 to 0.02, for the sensor data we used in our simulations. It is a little more energy-efficient when α = 0.02.

Fig. 7 compares the sink data when α selects 0.02 and ∞, which stand for the optimal performance of DREET and no transport protocol used, respectively (when α = ∞ , there will be no *key data* and, no ACK or retransmission consequently). We see that DREET (the curve in the middle) preserved all the 19 events occurred at the sensor node (the curve at the top). However, in the simulation with no transport protocol (the curve at the

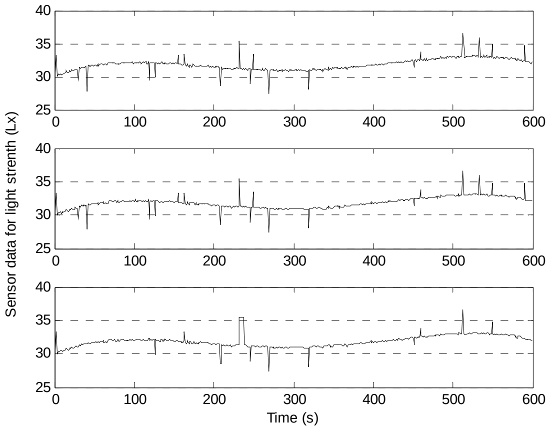

Figure 7. Sensor data for light strength (in Lux) during 600 seconds. From top to bottom, the three curves represent: data sensed by sensor node; sink data with α = 0.02; sink data with α = ∞, which means no transport protocol employed. (For the last two curves, the lost part of sink data, caused by packet loss, has bee n recovered by numerical method)

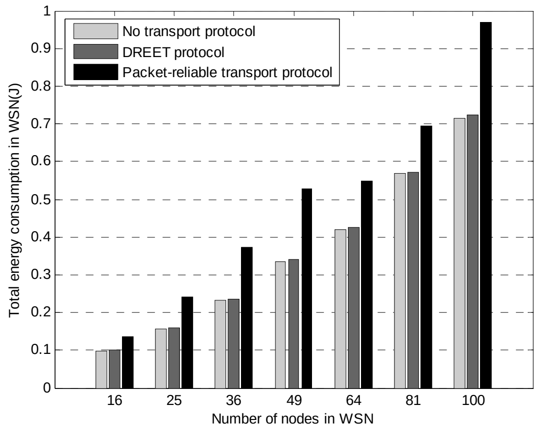

Figure 8. Total energy consumption for different transport protocol and varying network size.

bottom), the sink data preserved only 12 events, and the rest 7 events were lost due to packet loss.

C. Energy Spent for Different Transport Protocol and Varying Network Size

We set up simulations of different network sizes and different transport protocol to analyze the energy efficiency of DREET. Seven WSNs of different sizes (16~100 sensor nodes) were simulated, and for each WSN, we performed 3 simulations with different transport protocol (no transport protocol, DREET and packet-reliable transport protocol). For DREET, we select 0.02 as the value of α. A general trend of total energy consumption increase with network size is evident from our preliminary studies in Fig. 8. An inspiring finding is that the introduction of DREET (α = 0.02) only brings a tiny increase on energy consumption, compared to WSN with no transport protocol. The largest increase among all scenarios is only 1.97%, which occurs in the 25-node WSN.

VI. CONCLUSIONS

The notion of *data reliability* is necessary for reliable detection of events in WSN. This is due to the fact that the sink is only interested in the sensor data received from sensor nodes, and the sensor data vary very slightly in a relatively short time, which makes it possible to provide reliable transport just for a proportion of sensor data (referred as *key data* in this paper) instead of for all of them. Based on this notion, we propose a data-reliable energy-efficient transport layer protocol (DREET).

DREET is a novel transport layer solution for reliable event detection in WSN with minimum energy expenditure. To the best of our knowledge, it is the first study of reliable transport in WSN from the *data reliability* perspective. DREET also give an efficient solution for congestion control, which can reduce the loss of *key data* even when congestion occurs, and alleviate the congestion in current network.

Simulation results show that DREET provides an optimal event detection rate and relative standard deviation between the data sensed by sensor node and that delivered to applications. However, DREET achieves these advantages by introducing only a tiny increase on energy consumption, compared to WSN with no transport protocol. We plan to investigate the congestion problems in WSN, and accomplish congestion control functionalities in future work.

REFERENCES

[1] C Wang, K Sohraby, B Li, M Daneshmand and Y Hu, “A Survey of Transport Protocols for Wireless Sensor Networks”, IEEE Network, Vol. 20, pp.34-40, 2006.

[2] C Wang, K Sohraby, Y Hu, B Li and W Tang, “Issues of transport control protocols for wireless sensor networks” Proceedings of Communications, Circuits and System, pp.422-426, 2005.

[3] S Ganesh and R Amutha, “Energy Efficient Transport Protocol for Wireless Sensor Networks”, Proceedings of Computer Science and Information Technology, Beijing, China, pp.464-468, 2009.

[4] C Wan, A Campbell and L Krishnamurthy, “PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks”, Proceedings of Wireless sensor networks and applications, Atlanta, GA, pp.1-11, 2002.

[5] Y Iyer, S Gandham and S Venkatesan, “STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks”, Proceedings of Computer Communications and Networks, San Diego, CA, pp.449-454, 2005.

[6] Y. Sankarasubramaniam, O.B. Akan, and I.F. Akyilidiz, “ESRT: EventtoSink Reliable Transport in Wireless Sensor Networks”, Proceedings of the MobiHoc, Annapolis, MD, pp.177-188, 2003.

[7] A Cerpa, J Elson, D Estrin, L Girod M. Hamilton, and J Zhao, “Habitat monitoring: Application driver for wireless communications technology”, in ACM SIGCOMM Workshop, Latin America and the Caribbean, pp.20-41, 2001.

[8] Castalia, http://castalia.npc.nicta.com.au/.

[9] OMNeT++, http://www.omnetpp.org/.

[10] Castalia User’s Manual, Edited by Athanassios Boulis, NICTA, 2011.

[11] B. Hull, K. Jamieson, and H. Balakrishnan, “Mitigating Congestion in Wireless Sensor Networks”, Proceedings of the 2nd international conference on Embedded networked sensor systems (ACM SenSys’04), MD, pp.134-147, 2004.

[12] C. T. Ee and R. Bajcsy, “Congestion Control and Fairness for Many-to-One Routing in Sensor Networks”, Proceedings of the 2nd international conference on Embedded networked sensor systems (ACM SenSys’04), Baltimore, MD, pp.148-161, 2004

[13] J. Paek and R. Govindan, “RCRT: rate-controlled reliable transport for wireless sensor networks”, Proceedings of the 5th international conference on Embedded networked sensor systems (ACM SenSys’07), Sydney, Australia, pp.305-319, 2007,

[14] C. Y. Wan, S. B. Eisenman, and A. T. Campbell, “CODA: Congestion Detection and Avoidance in Sensor Networks”, Proceedings of the 1st international conference on Embedded networked sensor systems (ACM SenSys’03), Los Angeles, CA, pp.266-279, 2003

[15] C. Wang, K. Sohraby, V. Lawrence, B. Li and Y. Hu, "Priority-based Congestion Control in Wireless Sensor Networks," Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06), pp.22-31, 2006

[16] C. Y. Wan, S. B. Eisenman, A. T. Campbell and J. Crowcroft, “Siphon: Overload Traffic Management Using Multi-Radio Virtual Sinks in Sensor Networks”, Proceedings of the 3rd international conference on Embedded networked sensor systems (ACM SenSys’05), San Diego, CA, pp.116-129, 2005

[17] A. Woo and D. C. Culler, “A Transmission Control Scheme for Media Access in Sensor Networks”, Proceedings of the 7th annual international conference on Mobile computing and networking (ACM Mobicom’01), Rome, Italy, pp.221-235, 2004.

Gang Han, born in Ji'nan, China, on Jul. 1, 1985, is now a joint Ph.D. student of College of Computer Science, National University of Defense Technology (NUDT), and Cyber Physical System Laboratory (CPSL), McGill University. He received his Bachelor's degree of Engineering in computer science and technology from College of Computer Science, NUDT, Changsha, P. R. China, in 2006. He received his Master's degree of Engineering in computer science and technology in NUDT, in 2008. And during this period, his main research topic is routing and security of mobile ad hoc networks (MANET).

He has published more than 10 papers in journals and conferences. His current research interests include wireless sensor networks, cognitive radio networks, data center, communications in HPC, and etc.

Jia Lu, born in Jiuquan, China, on Jan. 12, 1985, is now pursuing his Ph.D. degree in College of Computer Science, National University of Defense Technology (NUDT). He received his Bachelor's degree of Engineering in computer science and technology from College of Computer Science, NUDT, Changsha, P. R. China, in 2008. He received his Master's degree of Engineering in computer science and technology in NUDT, in 2010. And during this period, his main research is routing on mobile ad hoc networks (MANET). His current research interests include real-time system, wireless networks, network calculus, and so on.

Xinbiao Gan, born in Anlu, China, on Aug. 23, 1982, is now pursuing his Ph.D. degree in College of Computer Science, National University of Defense Technology (NUDT). He received his Bachelor's degree of Engineering in computer Software from College of Computer Science, Wuhan University of Science and Technology, Wuhan, P. R. China, in 2006. He received his Master's degree of Engineering in computer science and technology in NUDT, in 2008. And during this period, his main research is on compiling for general-purpose GPU.

He has more than 10 papers published in journals and conferences, 3 of which are indexed by SCI. His current research interests include parallel architecture, compiling in GPU, data-flow processor, etc.

Huanzhong Li, born in Nanning, China, on Dec. 9, 1984, is now pursuing his Ph.D. degree in College of Computer Science, National University of Defense Technology (NUDT). He received his Bachelor's degree of Engineering in computer science and technology from College of Computer Science, NUDT, Changsha, P. R. China, in 2005. During 2008 to 2009, he studied in McGill University as a visiting scholar, and got great achievement there

He has more than 10 papers published in journals and conferences, including a paper of ICDCS. His current research interests include wireless networks, network calculus, network coding, etc.

Wenhua Dou, born in Pingding, China, in 1946, is a professor in College of Computer Science, National University of Defense Technology (NUDT).

He has more than 90 papers published in journals and conferences. His current research interests include high-performance computing, photonic interconnection and communication, wireless networks, network calculus, network coding, etc