Lightweight Protocol Suite for Wireless Sensor Networks: Design and Evaluation

Haiming Chen[12], Changcheng Huang[1] and Li Cui[1†]

[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100080, China

[2]Graduate School of the Chinese Academy of Sciences, Beijing, 100049, China

E-mail: {chenhaiming, huangchangcheng, lcui}@ict.ac.cn

[†] Tel: +86-10-6260 0724, Fax: +86-10-6256 2701

Abstract— **Lots of protocols have been proposed to make sensor networks more reliable, efficient and applicable. However, a vitally important problem has been neglected, that is the feasibility to implement the protocol and compatibility with other protocols. In this paper, a layered architecture was designed to organize the sensor network protocols in a light way, with detailed description on the interactive interfaces between layers. In accordance with the architecture, a suite of lightweight protocols were designed and implemented as prototypes. Our proposed protocols are featured by their low requirements on the node resources, and high possibility to be deployed in the real world. All these protocols were implemented and evaluated in a unified simulation environment. Simulation results verified our expectation on the performance of the protocols.**

I. INTRODUCTION

The development of wireless sensor network has given us a chance to reveal previously unobservable phenomena [1]. Forecasting its wide application in both civilian and industrial fields, more and more researchers are making their efforts on various aspects of this promising area, most of which are on the networking issues and energy consumption problems. To date, lots of efficient protocols have been proposed, and some of them have also been implemented in laboratory test beds. The common characteristic of the implemented protocols is *lightweight*. Due to the resource constraints on energy, storage and bandwidth in the sensor network, the implemented protocols cannot be very complicated. They have to be efficient in computation and communication.

In this article, we depict a suite of lightweight protocols for wireless sensor networks in accordance with an interactive layered framework. Our framework inherits the merits of *function decomposition* of the TCP/IP architecture, and prunes some complicated functions from the TCP/IP architecture making it more suitable for the resource-restricted sensor networks. In addition, to verify the *feasibility and efficiency* to implement our proposed framework and lightweight protocols, we implement the protocols as prototypes in a *unified* simulation environment, and evaluate their performance by extensive simulation experiments. The preliminary results demonstrate the feasibility to transplant the proposed framework and protocols to real sensor nodes.

The main contributions of this paper are embodied in the following aspects.

1. The protocol architecture proposed in the paper is adapted from the TCP/IP and compatible with TinyOS [2]. So any protocols designed in accordance with the framework can be easily transplanted to the popular hardware platform.

2. The proposed lightweight protocols conforming to our unified sensor network framework have been implemented in a unified simulation platform, and their performances are widely evaluated. Our motivation to establish a unified simulation platform to verify our proposed framework and protocols can be attributed to following reasons. Firstly, the existing common-purpose network simulation platforms, like NS2 and GloMoSim [3], are specifically for traditional wired and wireless networks, and are built upon the tradition layered network model, they can not be easily extended to simulate cross-layer interaction which is required for the sensor network. Secondly, the specific-purpose emulators, like TOSSIM [4] and EmStar [5], excel in evaluating the performance of implemented protocols in code level, but have fatal problem in scalability.

The rest of the paper is organized as follows. We begin with the description of the protocol architecture designed for the sensor network. Next, details of designing a lightweight contention-based media access control protocol is presented in Session III, followed by elaboration on designing a lightweight flooding protocol for the sensor network in Session IV. And then, we present the procedure of implementing the lightweight protocols in the application of temperature monitoring, and evaluating their performances in the metrics of delivery rate and packet delay in Session V. In addition, we point out some problems and future work in Session VI. At last, we make a brief conclusion in Session VII.

II. A UNIFIED FRAMEWORK FOR SENSOR NETWORKS

A. Principle and concepts

Neither the traditional network reference model proposed by ISO, nor that of IEEE 802, is fit for sensor networks, due to the specific characteristics which make the sensor network essentially differ from the traditional wired and wireless net-

This paper is supported in part by the National Basic Research Program of China (973 Program) under Grant No. 2006CB303000, and National High Technology Research and Development Program of China (863 Program) under Grant No. 2006AA01Z215, and NSFC project under Grant No. 60572060.

work. Since the traditional wired networks are usually composed of high performance computers or routers, their primary goal is to provide high-speed service for the customers, so the existing protocols for the wired networks are mostly based on the principle of achieving high throughputs and maintaining reasonable flexibility. As for the protocol architecture of the wireless local area network or ad hoc network, although there are some resemblance between those networks and sensor networks, the former ones do have looser restrictions on resources of computation, communication, and power supply. Taken into consideration the strict constraints on these resources in sensor networks, some simplified protocol architectures have been proposed. But so far there has been no consensus on the architecture which is really suitable and feasible for sensor networks. So the authors of [6] called on a unified architecture of sensor networks for both simulation environments and hardware platforms. The advantage of unifying the network architecture is to allow the protocols developed in different environments and platforms be easily integrated together, more reusable and more compatible.

We believe a unified protocol architecture is essential for the further development of sensor networks, but it can never make sense to cast the architecture of the traditional networks to the sensor network without much adaptation. Taking into accounts the merits of the TCP/IP model on function decomposition, we adopt the idea of layering and adapt it to the sensor network.

B. A unified protocol architecture for the sensor network

The traditional TCP/IP architecture categorized the protocols into five layers, of which the networking layer and transport layer are in the core. As was mentioned above, the primary goal of the traditional networks is to provide high-speed services for the customers. At present, protocols in the networking layer of the traditional wired or wireless networks have high requirements on the capacity of computation and communication. To guarantee the reliability of end-to-end transmission, the transport layer is becoming more complex and computation-concentrated. These heavyweight protocols are necessary for the traditional networks to provide acceptable services, but for sensor networks it is too expensive to provide such services. So we designed a unified protocol architecture for the sensor networks by pruning the transport layer from the TCP/IP suits and simplifying the interfaces between layers.

The adapted protocol architecture is shown as Fig. 1. From top to bottom, they are application, routing, media access control and physical layers. Data flow and interaction between the layers are accomplished by the interfaces provided between the adjacent layers. However, interfaces defined in our proposed architecture for the sensor network is much more light-weighted than that in the traditional TCP/IP suite.

C. Application layer

Nodes in a sensor network all have communication modules, but not all of them are supposed to be equipped with sensing modules. Only the nodes embedded in the physical environment, namely *sensor nodes*, are responsible for sensing and generating data, which can be driven by time, event or command. The generated traffic is characterized with both spatial and temporal correlation, which makes the application protocols of sensor network different from that of the traditional one. Other kinds of nodes, namely *network nodes* and *sink nodes*, take the responsibilities of relaying and collecting data.

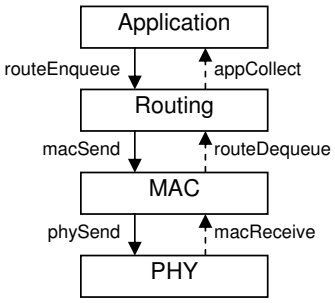

Fig. 1. Unified protocol architecture for the sensor network

From the perspective of traffic flow, the sensor node and sink node are the source and destination respectively, while the network node is the intermediate. So sensor nodes need an application protocol to perceive the environment and inject the results to the network, and the sink node need an application protocol to collect data from the network. In general, for the application layer of the nodes in the sensor network, following two interfaces should be provided.

1. One is provided by the sensor node to interact with the sensing module so that readings can be injected into the network, which is named as *appSense*;

2. The other is provided by the sink node to collect the data from the network, which is referred to as *appReceive*.

D. Routing Layer

Routing layer is assigned to deliver the generated data from the sensor nodes to the sink node, or disseminate query command from the sink node to the sensor nodes, by establishing a multi-hop path. The traditional routing schemes, such as shortest path first or energy efficient, are still suitable for sensor networks, but are not optimal. It is suggested that the routing protocols should be data centric [7] to provide support for data aggregation and fusion. When designing the architecture, we only take into consideration the basic function that the routing protocol is supposed to provide, but not the exact processing procedures. For us, the basic function of the routing protocol is to establish routes rooted at the sink node according to specific metrics, and route the data along the established path. So the indispensable interfaces for the routing layer are listed below.

1. *routeEnqueue*: Determine the next hop of the packet generated in the application layer or received from the MAC, and put the routed data into the queue if the current node is not the destination. In view of the limit on the storage resource for the sensor and intermediate nodes, the permitted maximum queue

length of the sensor networks is far less than that of the traditional networks.

2. *routeDequeue*: Provided for the MAC layer to fetch a packet from the queue and put it into the buffer of MAC layer when it is in the idle state.

E. MAC layer

The principal function of the media access control layer is to provide a fair access scheme for all the nodes in the network. Current proposed schemes can be divided into two categories. One is based on contention; the other is based on scheduling. Whatever the scheme is based on, the core procedures of the MAC is the same, that is, fetching a packet from the routing queue, then waiting until the media is available, and then transmitting the packet to the physical layer. So the two interfaces required for the MAC layer are shown as follows.

1. One is provided for the routing layer to send the routed packet to the MAC layer, which is named as *macSend*.

2. The other is provided for the physical layer to deliver the received frames to the MAC layer, which is named as *macReceive*.

As for the procedures of contending or allocating the media to the nodes, they are specific for different protocols. So it cannot be abstracted as an interface.

F. Physical layer

The main functions of the physical layer are transmitting the bits after modulating, and delivering the received error-free bits to the MAC layer. Another important function supposed to be implemented in the physical layer is informing the MAC layer of the current state and state change of the physical radio model to avoid access collision. So the following interfaces should be provided by the physical layer.

1. *phySend*: Called by the MAC layer to send the frame when the physical radio module is idle.

2. *phyReceive*: Triggered by the physical radio model when sensing the arrival of signal.

From the above description of the framework, we can see its compatibility with TinyOS. Besides that, our proposed framework is more general than that implemented in TinyOS, for both simulation environments and hardware platforms.

The following sections will be dedicated to describing the details of designing a suite of lightweight protocols, especially on the MAC layer and routing layer. As for the physical layer protocol, we take it for regarded that it provides the basic services, namely transmitting, receiving and feeding back the states of radio module, to the upper layer.

III. A LIGHTWEIGHT MEDIA ACCESS CONTROL PROTOCOL

The traditional goal of the media access control protocol is to achieve fairness and high utilization among the nodes. But in the sensor network, where traffic is not so heavy and collision is not so severe, the media access control scheme should not be so complicated. Inspired by the radio control model integrated in the TinyOS, we designed a lightweight media access control protocol, named *TinyMAC*. Since the protocol should be refined further, we firstly established a simple model, which does not take the power saving mechanism into consideration. The finite state automaton of the protocol is shown as Fig. 2.

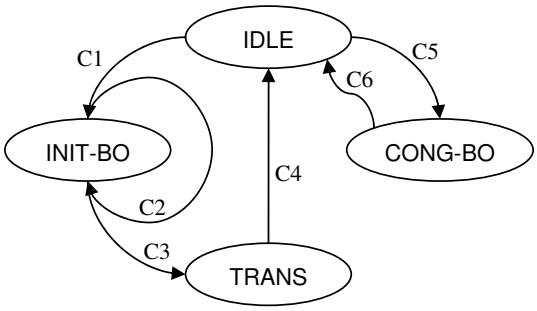

Fig. 2. Finite state automaton of the TinyMAC protocol

In the above automaton, there includes four states, namely idle (IDLE), initial back-offing (INIT-BO), congestion back-offing (CONG-BO) and transmitting (TRANS). The words capitalized with C stand for the conditions to trigger the change of the states. For example, supposing the current state is idle and a packet needs to be sent, if the current state of the physical layer is idle (that is C1), the node changes its state into initial back-offing; otherwise (that is C5), the node starts congestion back-offing. When the congestion back-off ends (that is C6), it returns to idle and repeats the above described process. During initial back-offing, notification of state change (from idle to busy, that is C2) from the physical layer can stop its progress, but cannot make the current state changed. When the initial back-off ends (that is C3), the node starts transmitting the packet in the buffer. The node does not return to idle until the last bit of the packet is sent (this is C4).

As our preliminary work, the length of the initial back-off or congestion back-off is random, but not adjustable. In the standard of IEEE 802.11, the window size of back-off is suggested to be increased exponentially with the times of collision, which is detected by the sender with the timeout of the arrival of the expected acknowledgment from the receiver. In order to reduce the complexity of the media access protocol in the sensor network, we temporarily exclude the reliable transmission mechanism from the link layer. So it is impossible to adjust the length of back-off like IEEE 802.11. Inspired by the radio control model implemented in TinyOS, we adopt the following equations to determine the length of the initial back-off and congestion back-off.

$$InitBO = \text{randInt}(0,31)+1 \quad (1)$$

$$CongBO = \text{randInt}(0,15)+1 \quad (2)$$

The function of randInt(a, b) is defined to return a random number between a and b. So the maximum length of the initial back-off is 32 bytes, which implies about 13.3 milliseconds when the channel bandwidth is 19.2 Kbps. For the

same reason, the maximum length of congestion back-off is about 6.7 milli-seconds.

From the above description, we can confirm that *TinyMAC* is really light-weighted, but at the cost of losing reliability in the link layer than the traditional wireless networks.

IV. A Lightweight Routing Protocol

Routing protocols are considered specific for the sensor network, since it is suggested to provide support for data aggregation or fusion to reduce the volume of data and energy consumption. In this paper, we don't put our effects to design a data-centric or power-efficient routing protocol for the sensor network. Instead, we concentrate on designing a lightweight routing protocol based on flooding, which is referred to as *TinyFlood*.

Flooding is considered to be the simplest scheme to route data in the multi-hop network. But it is also well known that flooding can lead to inner explosion and overlapping, which can waste a lot of bandwidth and other network resources. To reduce the overhead resulting from the uncontrolled flooding, we make following improvements to avoid flooding loop and infinite re-forwarding of outdated packet.

One way to avoid looping packet in flooding is making the node remember the previously flooded packets. When an intermediate node receives a flooded packet, it firstly checks whether it has been flooded by the current node; if not, it will relay the packet by flooding and keep the packet in the memory; otherwise, it will drop the packet. For nodes in the sensor network, memory is severely constrained, so tracing the previously flooded packet seems impractical.

In *TinyFlood*, we make the packet carry the information of the route through which it has been flooded. When an intermediate node receives a flooded packet from the upstream, it checks whether itself has been included in the trace list of the packet; if not, add itself to the trace list and re-flood the packet to the downstream; otherwise, it will drop the received packet. Supposing the upper bound of the number of nodes in the sensor network is 65536, the length of identity of each node will not exceed 2 bytes. So the maximum length of the extra payload added to the packet depends on the TTL (Time To Live) configured by the user, which can be expressed as equation (3). Through this improvement, the problem of flooding loop is solved without requiring large memory.

$$L_{max\_extra\_payload}=2\times \mathrm{TTL} \tag{3}$$

Considering the limited bandwidth of the sensor network, each packet cannot be so long, or else the network will always in burst state. Supposing the total packet size can not exceed 36 bytes, and the average length of data generated by the sensor node is 24 byes, it can be easily deduced from (3) that the TTL can not be more than 6.

To focus our efforts on the lightweight of the protocol, we don't add any other complicated characteristics to the routing protocol, such as data-centric and energy efficient. But we believe that as our future work goes, we will take these factors into consideration.

V. Implementation And Evaluation

To evaluate performances of the above described protocols, we implemented them as prototypes in a unified simulation environment.

A. Implementation of TinyMAC

As described in Section II, the protocols in the MAC layer are supposed to provide an interface named *macSend* for the routing layer, and another interface named *macReceive* for the physical layer. For *TinyMAC*, we implemented these two interfaces named *macTinySend* and *macTinyReceive* respectively.

For *macTinySend*, it takes the responsibility of fetching a packet from the queue of the routing layer to the buffer of the MAC layer, and contending for the access to the channel by back-offing. Due to the constraint on storage resource, we assume that there is limited space to hold only one packet in the MAC layer. Table I shows the procedures of the algorithm.

TABLE I
ALGORITHM OF MACTINYSEND

```
1:  Examine the state of the MAC layer macState;
2:  IF macState is idle THEN
3:    Fetch a packet, pktToBeSend, from the queue of the routing layer;
4:    Put pktToBeSend into the buffer of MAC layer;
5:    Examine the state of the physical layer phyState;
6:    IF phyState is idle THEN
7:      Start initial back-offing;
8:    ELSE
9:      Start congestion back-offing;
10:   ENDIF
11: ENDIF
```

As for the *macTinyReceive*, it simply delivers the successfully received packet to the routing layer if it is destined to the current node or it is a broadcast packet. The brief description of the above procedure is presented in Table II.

TABLE II
ALGORITHM OF MACTINYRECEIVE

```
1:  Examine the intermediate address of the received packet interAddr;
2:  IF interAddr is identical to the current node OR
3:    it is a broadcast address THEN
4:    Put the received packet into the queue of the routing layer;
5:  ELSE
6:    Drop the received packet;
7:  ENDIF
```

Due to space limitation, we leave out other details, such as the process of interrupt of the initial back-off by the notification from the physical layer (changing from idle to busy).

B. Implementation of TinyFlood

From the algorithm of *macTinySend*, we can see that an interface should be provided by the protocols in the routing layer, through which packets in the queue can be fetched by

the MAC layer. The principle of the dequeuing algorithm for the sensor network resembles that for the traditional network, in more detail that is scheduling a packet with the highest priority to be sent to the lower layer. Considering the limited storage resource in the sensor nodes, we assume there is only one FIFO queue in the routing layer in each node. So the scheduling mechanism is also very simple in *TinyFlood*, just moving the head packet in the queue of routing layer to the buffer of MAC layer. We implemented the interface as *routeDequeue*, which is consistent with the interface definition in the protocol architecture.

According to the protocol architecture defined above, the other interface supposed to be implemented is *routeEnqueue*. From the algorithm of *macTinyReceive*, we can also see the necessity to implement an interface to put the received packet into the queue of the routing layer. As mentioned above, there is only one queue in the routing layer for storing the packets to be sent to the MAC layer, so the packet should be routed to the next hop before it is put into the queue. Since the processing speed of CPU is sufficiently high compared with the transmission rate in sensor network, it is reasonable to remove the input queue and integrate the process of routing with enqueuing. The space reserved for the queue in our implemented protocol is 3600 KB, which is sufficient to buffer about 100 packets if the maximum packet size is 36 bytes. Table III shows the algorithm of *routeEnqueue* in detail.

TABLE III
ALGORITHM OF ROUTEENQUEUE

```
1:  IF the current node is the destination of the packet THEN
2:      Deliver the received packet to the application layer;
3:  ELSE
4:      Route the received packet to the next hop;
5:      IF the packet has been sucessfully routed to the next hop
6:          and the queue is not full THEN
7:          Put the routed packet to the tail of the queue;
8:      ELSE
9:          Drop the packet;
10:     ENDIF
11: ENDIF
```

As described above, the mechanism for routing the packet to the next hop is based on flooding. The improved algorithm of flooding is presented in Table IV.

TABLE IV
ALGORITHM OF ROUTEFLOOD

```
1:  Check the route trace carried in the head of the received packet;
2:  IF the current node has been in the list THEN
3:      Return failure to route the packet;
4:  ELSE IF the length of the route trace is equal to TTL[a] THEN
5:      Return failure to route the packet;
6:  ELSE
7:      Add the current node to the route trace;
8:      Return success in routing the packet;
9:  ENDIF
```

[a] The value of TTL in our implemented prototype is 6.

C. Performance Evaluation

To illustrate the performance of the above described protocols clearly, we set up a typical scenario which is composed of a line of nodes. As shown in Fig. 3, six sensor nodes and one sink node positioned in a horizontal line comprise the typical topology setting. Distance between two adjacent nodes is 100 meters, which is equal to the transmission range of each node.

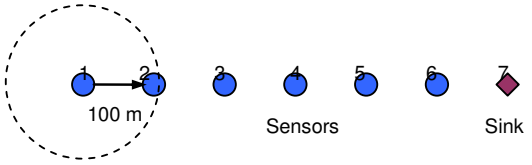

Fig. 3. Topology setting for performance evaluation

Sensor nodes are assigned to read the temperature every 1 second after 20 seconds preparation, and send the readings to the sink node. All the nodes keep working for 1 hour. So the sensor nodes are supposed to send 3580 temperature data to the sink node. Fig. 4 shows that all the packets can be collected by the sink if the bandwidth is 2.4 Kbps. But the delivery rate will decrease as the number of hops increases if the bandwidth is 1.2 Kbps, more specifically about 60% of the data generated by node 1 can be received by the sink, while almost 100% of data generated by node 5 and node 6 can be collected by the sink.

The factor leading to the packet loss can be easily found from Fig. 5. When the 6th node or the 5th node is assigned as the sensor node, collision happened in its adjacent node (in the direction to the sink) is zero, but when the other nodes are to send their readings to the sink, collisions occurred in their adjacent nodes are severe, which arise from the flooding and lack of scheme to solve the problem of hidden terminal in the MAC layer. However, the results are in our expectation, because our protocols are designed by taking the *lightweight* as our main objective.

Fig.6 shows the average delay of packets collected by the sink node. From the results we can see that packet delay from the sensor node to the sink increases linearly with the number of hops, whatever the bandwidth is. The result is also in our expectation, since we do not take priority into consideration when implementing the queue, so that packets sent from the farther nodes are expected to reach the sink later.

The effectiveness of the improvement of the flooding protocol is presented in Fig. 7. When the nodes in the two extreme are the sensor nodes, the total packets dropped by the intermediate nodes are more than ten thousand. The results show that the lightweight mechanism can effectively avoid looping in flooding.

VI. FUTURE WORK

So far, we have evaluated performances of our proposed protocols with typical scenario in a simulation environment.

Although the results have verified our expectations on the protocols, the performance of our proposed lightweight architecture and protocols should be further evaluated in a large scale random environment. In addition, we plan to implement the protocols in a hardware platform.

Besides that, we will add some efficient power saving mechanisms into the above proposed protocols. A suite of lightweight energy efficient protocols will be designed and implemented in the end.

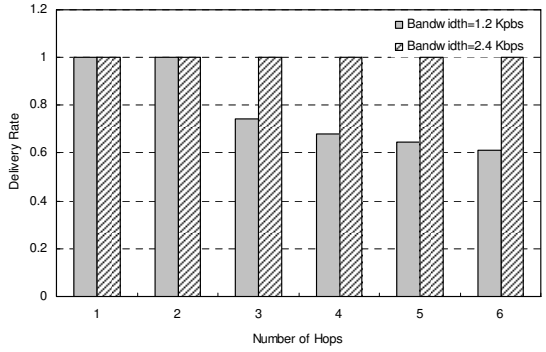

Fig. 4. Delivery rate for each sensor node with different hops to sink

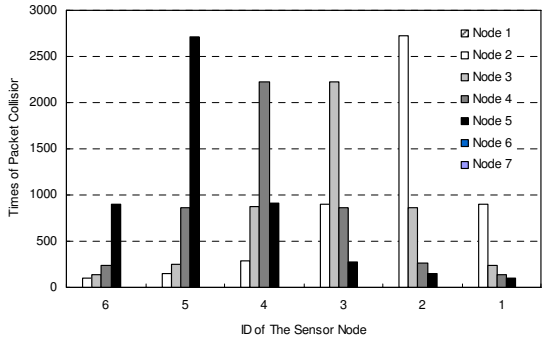

Fig. 5. Number of collision occurred in each node

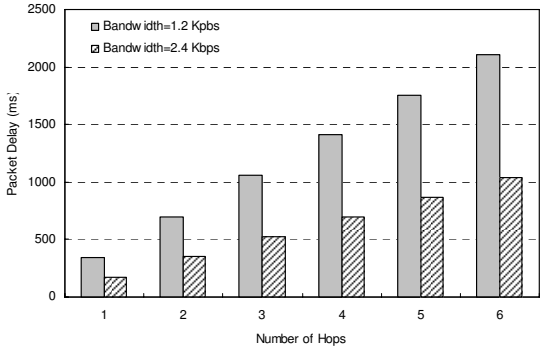

Fig. 6. Average delay of packets collected by the sink

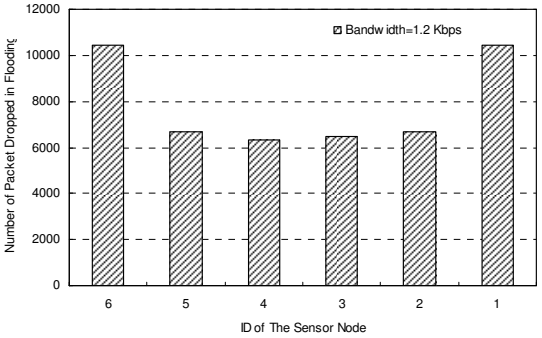

Fig. 7. Total number of packets dropped in flooding

VII. Conclusions

In our opinion, establishing a unified protocol architecture and implementing a suite of lightweight protocols are of vital importance for the further development of sensor networks, especially for the promotion of its applications.

In this paper, we firstly proposed a protocol architecture, which combined the merits of the architecture of TCP/IP and TinyOS; then we designed a suite of lightweight protocols in accordance with the proposed architecture. At last, we evaluated the performance of the above protocols by extensive simulations. Results demonstrated the correctness and effectiveness of our proposed architecture and protocols. Due to page limitation, here we only elaborated the protocols of MAC layer and routing layer, which we referred to as *TinyMAC* and *TinyFlood*. However, in future more work will be done to improve the architecture and the protocols.

References

[1] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, "Connecting the Physical World with Pervasive Networks," *IEEE Pervasive Computing*, vol. 1, pp. 59-69, Jan. 2002.

[2] J Hill, P Bounadonna, D Culler, "Active message communication for tiny network sensors", in *Proceedings of INFOCOM*, 2001.

[3] X. Zeng, R. Barodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large–Scale Wireless Networks," In *Proceedings of Workshop on Parallel and Distributed Simulation*, pp. 154–161, 1998.

[4] P. Levis, N. Lee, M. Welsh, and D.Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," In *Proceedings the 1st ACM Conference on Embedded Networked Sensor Systems(SenSys03)*, pp. 126–137, 2003.

[5] J. Elson, L. Girod, and D. Estrin, "EmStar: Development with High System visibility," *IEEE Wireless Communication*, pp. 70–77, December 2004.

[6] V. Handziski, A. Kopke, H. Karl, and A. Wolisz, "A Common Wireless Sensor Network architecture?" *Technical report of the Telecommunications Networks Group*, Technische Universitat Berlin, 2003.

[7] B. Krishnamachari, D. Estrin, S. Wicker, "Modelling data-centric routing in wireless sensor networks," in *Proceedings of the IEEE INFOCOM*, 2002.