Priority-based Hybrid Protocol in Wireless Sensor Networks

Hsu-Jung Liu$^1$, Mei-Wen Huang$^2$, Wen-Shyong Hsieh$^3$, Chenhuan Jack Jan$^4$

1Tajen University, PingTung, Taiwan
2Ph.D candidate at the Department of Computer Science and Engineering of NSYSU, Kaohsiung and with Tajen University, PingTung, Taiwan
3Sue-Teh University and NSYSU University, Kaohsiung, Taiwan
4National Pingtung Institute of Commerce, PingTung, Taiwan
1hjliu@mail.tajen.edu.tw
2mwhuang@mail.tajen.edu.tw

Abstract

Most of wireless sensor networks (WSN) research topics consider how to save the energy of the sensor nodes. However, in some applications of WSN, such as in the monitoring of an earthquake or forest wildfire, transmitting emergency data packets to the sink node as soon as possible is much more important than saving power. In this paper, we propose a Priority-based Hybrid Protocol (PHP) WSN model to provide a feasible WSN architecture, which can save the energy of the sensor nodes in normal situation, but will transmit emergency data packets in an efficient manner to the sink node. The simulation analysis shows that PHP obtains superior performance during emergency condition.

Keywords— Wireless Sensor Networks (WSN), energy efficient, Priority-based Hybrid Protocol (PHP)

1. Introduction and Related Works

The Wireless Sensor Networks (WSN) has been implemented in many applications and has been the subject of much research recently. The major restrictions of the WSN include limited energy supply, limited computing power, limited buffer size, and limited bandwidth [1]. Most of the WSN researching topics are focused on how to save the energy of the sensor node (SN). But in some applications of WSN, especially during emergency conditions, delivering data packets to the sink node as soon as possible is much more important than saving power.

Communications in WSN occur in different ways depending on the underlying application or mission of the network. As shown in figure 1, we define that the node D which near the sink node is downstream node and the nodes A, B and C are upstream nodes which are more far to the sink node than the node D. The data packets produced by SNs can be classified into three delivery models: clock-driven, event-driven and query-driven [2]. Clock-driven data packets are gathered by the SNs, and sent to the sink node periodically. In the emergency situation, the sensing data may be over the preset threshold value, the sensing node must send the event-driven data packets to sink node as soon as possible. In query-driven model, the SNs will be queried to reply data packets back to the sink. The event-driven packet and query-driven packet have higher priorities to be delivered than the clock-driven data.

Sensor networks usually operate under light load and become active in response to a detected or monitored event. The energy constraint and the low buffer size are the two important problems in the SN. It is hard to determine the sensing period (τ) required for a SN to sense and transmit data packets to the sink node in the clock-driven model. This is due to the fact that if the value of τ is too short, the SN will produce and send data packets to the sink node more frequently. Consequently, the energy of the SN will be consumed more quickly. On the other hand, the WSN cannot send the real time status to the sink node if the sensing period is too long.

The low buffer size of a SN is an important problem in the design of the WSN routing protocol. In general, the SNs will not send a large amount of data packets to the sink node during normal situation. For example, in the monitoring of forest wildfires, the SNs will send data packets to the sink node periodically (clock-driven), containing information regarding location, humidity and temperature. In the event-driven model, when the measured temperature is higher than the threshold value set in the SN, the triggered SNs will deliver a large amount of emergency packets to the sink node and will easily cause a congestion. Under this condition, the emergency packets that are far away from the sink node may not be delivered successfully.
since they have to compete with clock-driven data packets. Consequently, this causes the sink node unable to respond to all of the WSNs conditions in real time.

The buffer-based congestion avoidance algorithm [3] is efficient to prevent congestion occurring in WSN. As shown in figure 1, the upstream traffic from SNs toward the sink is many-to-one multi-hop convergent. Due to the convergent nature of upstream traffic, congestion more probably appears in the upstream direction. The key for hybrid is to make sure that the sensor nodes (USNi) send data packets to its downstream sensor node D only when node D has the buffer space to hold the packet. Therefore, in case of congestion, the upstream nodes (USni) should reduce the data while forwarding to the downstream node (node D).

In HC WSN model [5], the hybrid scheme but fixed time interval to report data packets from SNs was adopted. The major contribution of the HC WSN model is that, in cluster-based WSN model, which provides different priorities of data packets and delivers the higher priority packets in congestion condition. But HC WSN adopt fixed time interval to report data packets can not prevent buffer congestion well in the SN when the burst emergency event-driven traffic occurs. In ESRT [6], a sensor sets a congestion-notification (CN) bit in the packet header if its buffer is nearly full. The sink recalculates a new reporting data periodically and notices the SNs by the CN. CODA [7] provides a comprehensive discussion on congestion control and proposes an open-loop hop-by-hop backpressure mechanism and a close-loop multisource regulation mechanism. Xu and Christos [8] proposed a dynamic sleep time control in event-driven WSN, which does not provide different priorities traffic services. In this paper, we propose a Priority-based Hybrid protocol, called as PHP WSN model, to provide hybrid packets delivery priorities and to use flexible sensing period according to buffer length of the SN to prolong the lifetime of the SNs and to report event-driven data packets to the sink node as soon as possible.

The remainder of this paper is organized as follows. Section 2 details the algorithm of the proposed PHP delivery model in the WSN. Section 3 shows the performance simulation of the PHP model, and finally, in section 4, we describe our conclusions.

2. Priority-based Hybrid Protocol (PHP) WSN Model

For the computing constraint of the SN, in PHP model, each SN can transmit data packets using the hybrid model. The hybrid model means that the SN delivers data packets using a combination of clock-driven, event-driven and query-driven algorithms. Generally, in clock-driven model, the sensor nodes gather the data packets and send them to sink node periodically (τ) as figure 2-(a). In order to prolong the lifetime of the SNs during normal situation, the PHP extends the value of the clock-driven sensing period as shown in figure 2-(b). However, the derived problem is that the sink node can not receive the information on the condition of the WSN in time to prolong the sensing period of the SN. To solve the problem, there are α times event-driven senses inserted at every τ time.
interval between two clock-driven data packets. The event-driven scheme only consumes sensing energy of a SN in non-event condition. It is evident that the energy of transmitting data packets from a SN to its downstream node is much more than the sensing energy of the SN. We called that the combination of one time clock-driven measurement and \( \alpha \) times event-driven measurements a round. For example, as figure 2-(b) shows that \( \alpha \) equals 4, it means that there are one clock-driven data packet and four event-driven senses in a round. The event-driven data packet would be reported toward the sink only when the preset threshold in the SN was triggered. It is obvious that the SNs can easily save energy in non-event condition.

As for the problem of low buffer size in the SN, it is easy to cause congestion under emergency conditions triggered by the upstream SNs. As figure 3 shows, there are two buffers in the SN; one is for event-driven data packets, and another is for clock-driven data packets. The data packets produced by the SNs are classified into two different priorities. One is the clock-driven data packet, and another is the event-driven data packet or the query-driven data packet. We consider that the event-driven data packet and query-driven data packet have the same priority. The event-driven data packet (or the query-driven data packet) has higher priority than the clock-driven data packet. In the event-driven model, the threshold of the emergency condition is preset in the SNs. If the sink node does not obtain any data packets from the SN which is more interesting than a certain time interval, then the sink node will send a query-driven message to understand the condition of the SN. There are three phases in PHP: first is construction phase of the WSN, which will record the upstream nodes and downstream nodes in each of SN. After constructing the whole sensor network, each SN in the network will notice its upstream node either to stop transmitting packets when detects congestion condition occurs or should report packets in a reasonable rate according to the measured buffer length. Therefore, the congestion avoidance phase in a SN will prevent that burst traffic to cause congestion condition and drops a lot amount of packets. Then, the flexible sensing period adjustment phase will be used in the SN when congestion tendency occurs. The details of the three phases are described as follows.

2.1 Construction phase of the PHP WSN

In PHP, we consider the case of multipath routing to the sink and construct the WSN which is similar as Chen and Yang [3] proposed. There are two lists, \( D_x \) and \( U_x \), which are recorded in each of SN \( x \). The list of downstream neighbor sensors (\( D_x \)) can be used to forward packets toward the sink. Each sensor \( x \) will record its upstream SNs in the list of \( U_x \) with \( D_x = \{d_1, d_2, \ldots, d_m\} \) and \( U_x = \{u_1, u_2, \ldots, u_n\} \). The methodology to record the two lists in a SN is described as follows. Firstly, the sink periodically broadcasts a beacon to the neighbors’ SNs, which carries a unique identifier s and a hop counter c initialized to be zero. When a SN \( x \) receives a beacon for the first time (from a neighbor SN \( y \)), it increases the hop counter by one and records this hop count as \( c_x \), then insert node \( y \) to \( D_x \). The SN \( x \) returned the beacon with \( c_x \) to the neighbor node \( y \), and records node \( x \) into the list \( U_y \) in the SN \( y \). The SN
y will recognize its upstream SNs from the list of $U_y$ eventually.

Whether the SN $y$ could be inserted to $D_x$ or not is according to the hop count $c_y$ and $c_x$. When SN $x$ receives a subsequent beacon with $c_y$ from a neighbor SN $y$, it checks if $c_y <= c_x + 1$. If so, the SN $y$ will be inserted into $D_x$ of the SN $x$. The sink increases the broadcast identifier $s$ by one for each subsequent broadcast. Each SN keeps the largest identifier that it has recognized. It resets $D_x$ when receiving a beacon with a larger identifier, and discards all received beacons with smaller identifiers.

### 2.2 Congestion Avoidance Phase

In order to prevent burst emergency data packets to cause congestion occurs, the PHP have to detect the congestion condition in each of the SN. Any SN whose event-driven buffer overflows due to excessive incoming event-driven packets is said to be congested. Let $b_k$ and $b_{k-1}$ be the event-driven buffer lengths which are measured at the end of the $k^{th}$ and $(k-1)^{th}$ time intervals, respectively. The event-driven buffer size is $B$, as shown in figure 4. Let $\Delta b$ be the event-driven buffer length increment observed at the end of the last time period, i.e., $\Delta b = b_k - b_{k-1}$. If the sum of the current event-driven buffer level at the end of the $k$th time interval and the last experienced event-driven buffer length increment exceeds the buffer size, i.e., $b_k + \Delta b > B$, then the SN will indicate that it will experience congestion in the next time interval. Hence, the SN will stop to produce any data packet and send the upstream SNs an implicit ACK [4] including congestion notification (CN) to stop transmitting any data packets in the $(k+1)^{th}$ time interval. After that, the upstream SN will retransmit data packets to the downstream node until it stops receiving the CNs.

![Monitor the event-driven buffer length in the sensor node.](image)

### 2.3 Flexible Sensing Period Adjustment Phase

It is more easily to set the sensing period in a flexible range than a fixed value for a WSN application. After making sure the event-driven buffer of the SN will not overflow described in the congestion detection phase, in the flexible sensing period adjustment phase, the PHP proposes that the value of $\tau$ in next round is a flexible value which ranges from $\tau$ to $\tau'$ according to the measured average event-driven buffer length ($b_{avg}$) in this round as shown in figure 5. The value of $b_{avg} = \frac{\sum (the \ (\alpha+1 \ times \ measurement \ of \ event-driven \ buffer \ lengths \ in \ this \ round))}{(\alpha+1)}$ is shown in figure 2-(b). The value of $\tau_{avg}$ is calculated according to $b_{avg}$, which is described in the following different case conditions.

**Case 1.** $b_{avg} < B_{min}$: The SN will notice its upstream SNs to produce clock-driven or event-driven data packets every $\tau_{avg} - \tau$.

**Case 2.** $B_{min} \leq b_{avg} < B_{max}$: The clock-driven or event-driven data packets will be produced every $\tau_{avg} = \tau + \triangle \tau$, where $\triangle \tau = \frac{b_{avg} - B_{min}}{B_{max} - B_{min}} \ast (\tau' - \tau)$.

**Case 3.** $B_{max} \leq b_{avg}$: When the measured event-driven buffer length of the SN is reach or more than $B_{max}$, the SN will send congestion notification (CN) to its upstream SNs, and the upstream SNs will stop produce any data packet.
Fig. 5 Flexible sensing period (τ) adjustment in the sensor node.

Fig. 6 Operation flow chart in a sensor node.

1. Measure the buffer length $b_k$ at the end of the $k$th time intervals.
2. Calculate $Δb = b_k - b_{k-1}$.

If $b_k + Δb > B$
- Yes: The SN will stop to produce any data packet.
- No: The SN sends the upstream SNs an implicit ACK including CN to stop transmitting any data packets in the $(k+1)$th time interval. Wait for $τ_{avg}$ time interval. (detailed in Section 2.2)

End of a round
- Yes: Recalculate the value of $τ_{avg}$ for the next round (detailed in Section 2.3)
- No: Reevaluate the parameter setting.
2.4 Analysis the value of $\tau$ and $\tau'$

Figure 6 shows the operation flow chart of the congestion avoidance phase and flexible sensing period adjustment phase in a sensor node. It is more easily to set a reasonable range of $\tau$ and $\tau'$ than a best value of $\tau$ in any WSN application. In this paper, we define that the transmission rate of the SN x is $R_{SN}$. The number of upstream nodes of SN x is Ux, and SN x maybe produce data packets itself. The number of sensor nodes in the list of Ux is Num(Ux). Then, each of the upstream sensor node ux was dispatched at the rate of $\frac{R_{SN}}{\text{Num}(U_x) + 1}$ to transmit data packets to the SN x. To prevent congestion condition occurs, the value of $\tau'$ can be defined as the value of $\frac{\text{Num}(U_x) + 1}{R_{SN}}$. For the reason that we define the proportion of clock-driven /event-driven equals to $\frac{1}{\alpha}$. The value of $\tau$ can be set as $\frac{\text{Num}(U_x) + 1}{(\alpha + 1) * R_{SN}}$.

We would not try to define the best values of $\tau$ and $\tau'$, it depends on the WSN application and needs more other consideration.

3. Performance Analysis

3.1 Simulation Environment

For the performance analysis of the PHP model we perform extensive simulations to evaluate the proposed hybrid scheme. The simulation tool is the NS-2 simulator [9]. There are many energy consumption parameters: $E_{\text{sensing}}$ is the energy consumed in every sensor node. $E_{\text{amp}}$ is the energy dissipated by the signal power amplification of a sensor node. $E_{\text{elec}}$ is the energy dissipated by the signal processing of a sensor node. $E_{\text{eda}}$ is the energy dissipated by the data aggregation in the SN. $N_{\text{sensing}}$ is the number of sensor nodes having sensing data. $E_{\text{SN}(i)}$ is the energy dissipated by the data aggregation and the data transmission toward the sink of the $i$-th SN. The value of $E_{\text{SN}(i)}$ is the sum of $E_{\text{eda}}$, $E_{\text{amp}}$, and $E_{\text{elec}}$ as shown in Table 1.

Also, there are one hundred sensor nodes randomly located in the 100m*100m network topology for the simulation. Each SN initially has a uniform energy 2 joule. The location of the sink node is at (x=70m, y=120m). In order to restrict the simulation in a reasonable time and evaluate the performance of PHP, the initial value of $\tau$ is set as 0.1 seconds and the value of $\alpha$ equals to 4. It means that each SN reports a clock-driven packet every 0.5 seconds in non-event time period as shown in figure 2-(b). The value of $\tau'$ equals to $(\alpha+1)\tau$ seconds , which means that the event-driven packet will be produced in the range from 0.1 seconds to 0.5 seconds according to the measured average length of event-driven buffer in the SN.

<table>
<thead>
<tr>
<th>Table 1 Simulation Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
</tr>
<tr>
<td>Network size</td>
</tr>
<tr>
<td>Number of sensing nodes ($N_{\text{sn}}$)</td>
</tr>
<tr>
<td>The initial energy of sensor node</td>
</tr>
<tr>
<td>Buffer size (B) in SN</td>
</tr>
<tr>
<td>Processing delay</td>
</tr>
<tr>
<td>Location of Sink</td>
</tr>
<tr>
<td>Data packet size</td>
</tr>
<tr>
<td>Transmission rate ($E_{\text{tg}}$)</td>
</tr>
<tr>
<td>Processing energy ($E_{\text{mp}}$ + $E_{\text{elec}}$)</td>
</tr>
<tr>
<td>$E_{\text{elec}}$</td>
</tr>
<tr>
<td>$E_{\text{esp}}$</td>
</tr>
<tr>
<td>$B_{\text{fin}}$</td>
</tr>
<tr>
<td>$B_{\text{src}}$</td>
</tr>
</tbody>
</table>

3.2 Simulation Analysis

Simulation 1. The delivery ratios of event packets and clock packets

Low packets delivery ratio in WSN means that it is easy to waste the energy of SNs. In this simulation, we compare the packets delivery ratio of PHP scheme with no congestion control (NCC) scheme in the same WSN environment. Figure 7 shows the PHP can delivery event-driven data packets favorably. For the reason that, when a SN i congestion condition detected, PHP will extend the value of $\tau$ of the nodes in U/SN to slow down the production speed of data packets. If the SN has not any congestion control scheme, it would produce packets sustained and attempts to delivery them to sink node. This will cause a large number of packets loss and to decrease the packets delivery ratio.
Simulation 2. Comparison of accumulated packet loss

Accumulated packet loss of a WSN means waste the energy of the SNs. We compare the accumulative packet loss with PHP and backpressure scheme of CODA[7]. The backpressure scheme is that when sensor node i occurs congestion, in order to reduce the forwarding rate, it periodically sends backpressure messages to its neighbors. In the simulation, we set that there are randomly 50% SNs reporting event-driven packets periodically. Figure 8 shows that the PHP scheme seldom drops data packets due to buffer overflow. For the reason that, in the PHP scheme, the adaptive sensing period of $\tau$ will be adjustment well and prevent congestion condition occurring. When congestion occurs, the SN in the backpressure scheme will periodically sends backpressure messages to neighbors, which reduce the forwarding rate to the SN by 50 percent. The accumulative dropped packets of no congestion control (NCC) is more than that in the backpressure scheme and PHP model, which means that NCC will waste more energy than those two Hybrid schemes; and the PHP scheme shows better energy efficiency than backpressure scheme of CODA from the simulation result.

Simulation 3. Compare the number of residual sensor nodes alive

Figure 9 shows the residual SNs alive in the process of simulation. There are 50 percents of SNs randomly reporting event-driven data packets. The SNs periodically send backpressure messages to their neighbors using backpressure scheme, which reduce the forwarding rate to the SN by 50 percent. The simulation results show that the PHP can keep the SNs alive more efficiently for the reason that it can reduce the SN to produce packets when downstream nodes buffer overflow. The alive time of SNs in backpressure scheme is better than that with no congestion control but not as well as the PHP.
Simulation 4. Compare accumulated energy consumption of all sensor nodes in normal situation

To prolong the life time of all sensor nodes in normal (non-event) situation is one of the major purpose in the PHP model. We can prolong the lifetime of the WSNs by extend the value of $\alpha$. As shown in figure 10, the value of $\alpha$ is set as 3, the PHP can reduce the energy consumption of the SNs and extend their life time more well than that of backpressure and NCC schemes.

4. Conclusions

Saving energy and delivering emergency data packets to the sink node as soon as possible are the two important issues in most WSN applications. The PHP propose hybrid and flexible algorithm including different priorities data packets in order to report emergency message as soon as possible when congestion occurs. The simulation results show that the contributions of the proposed PHP WSN model can be described as follows: firstly, in order to prolong the lifetime of the SNs, we use a hybrid reporting data packets to the sink; secondly, in order to reduce the data packets lose, we use the flexible sensing period strategy. Finally, the PHP guarantees that the emergency packets will be transferred to the sink node as soon as possible by way of these two priorities buffers. From the simulation results, the PHP provide an efficient and practicable scheme in WSN.

5. References


