An efficient multi-channel communications scheme for wireless sensor network

Md. Shariful Islam, Muhammad Mahbub Alam, and Choong Seon Hong
Networking Lab, Department of Computer Engineering, Kyung Hee University, Korea 446-701
Email: {mahbub, sharif}@networking.khu.ac.kr, cshong@khu.ac.kr

***Abstract*—This paper presents a multi-channel communications system for wireless sensor networks (WSNs), referred to as Load-adaptive practical multi-channel communications (LPMC). LPMC estimates the active load of a channel at the sink, and dynamically adds or removes channel based on the estimated load. The nodes in a path use the same channel; and therefore, they do not need to switch channels to receive or forward packets. Furthermore, LPMC updates the routing path to balance the loads of the channels. LPMC is evaluated by simulation in NS-2, and the results demonstrate that it can effectively increase the channel utilization and network throughput, and decrease the delay.**

I. Introduction

In recent year, a significant number of sensor node prototypes have been designed that provide communications in multiple channels. This multi-channel feature can be effectively exploited to increase the ovarall capacity and performance of wireless sensor networks (WSNs). Withe multiple channel, a single adapter can use different channels at different times. If nearby nodes use orthogonal channels, more than one node can transmit simultaneously and increase the network capacity. Multi-channel communications can provide the required data delivery without adding extra resources. The use of a single channel is therefore not only an under-utilization of the limited resources of WSNs but also it might hinder the fidelity of the application.

To improve the network capacity, many multi-channel MAC protocols have been proposed. These protocols mainly assign (as part of the network setup) orthogonal channels to the nodes (either to the senders or the receivers) in a two-hop neighborhood [1], [2]. Data transmissions among the neighbors therefore require channel switching, and a sophisticated MAC scheme to find a rendezvous time for the sender and receiver. As a result, such protocols require fine-grained time synchronization among the nodes.

To minimize the channel switching, and to use multiple channels when it is necessary, a recent paper proposes a dynamic channel allocation policy based on control theory (hereafter referred to as DM-MAC) [3]. Because the nodes in DM-MAC change their channels in a distributed manner, for multihop communications, nodes still need to switch channels. To completely avoid the channel switching, a static channel allocation policy is proposed in TMCP [4]. TMCP divides the network into a number of sink-rooted sub-trees, and each subtree uses an orthogonal channel. However, the sub-tree creation requires a costly initialization phase.

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by MEST (No. 2009-0083838). Dr. C. S. Hong is the corresponding author

In this paper, we focus to design a multi-channel communications system for WSNs. LPMC dynamically adds or removes channels based on the active network load, and uses multiple channels whenever (when the network load is higher than the capacity) and wherever (part of the network where the load is high) it is necessary.

The main contributions of the paper are as follows. i) We propose a multi-channel communication systems for WSNs keeping as much of the protocol functionalities out of the sensor nodes. ii) LPMC dynamically identifies network load, and adds channel to the mostly loaded part. No initialization steps are required for LPMC, and the overheads for channel assignment is minimum. Nodes do not need to switch channel to receive or forward packets. iii) LPMC adds paths dynamically with non-interfering channels to a set of nodes to meet the traffic demands. iv) The performance of LPMC is evaluated by simulation in NS-2.

The rest of the paper is organized as follows. We present the proposed mechanism in Section II. Section III demonstrates the performance of LPMC. Finally, we conclude in Section IV.

II. Proposed Mechanism

A. LPMC Overview

To describe LPMC, we introduce the following notation and terminology. We define the base station (i.e., sink) as an entity that collects data from the sensor nodes (sources). We consider that the sensor network is mainly used for data collection. The data collection scheme builds a tree connecting the sink and the nodes. Each node forwards the data along the tree. We also assume that the sink is equipped with multiple transceivers each of which works in a different channel. The data of a particular node uses a single path. We assume there are K orthogonal channels available to use for the WSN.

LPMC aims to use minimum number of channels to deliver the data. If a single channel is sufficient, it uses one channel. If the generated data require more capacity, channels are added. In contrast, when the traffic do not need the added channels, channels are removed and eventually nodes use only one channel. All the nodes in a single path use the same channel. Therefore, the intermediate nodes do not need channel switching to receive and forward packets.

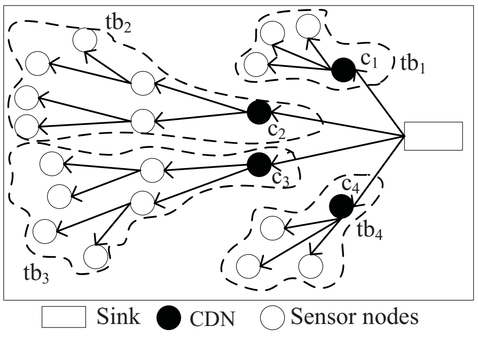

Fig. 1. A typical WSN scenario with 4 TBs rooted at 4 CDNs.

At the sink, LPMC has four distinct components that controls the operation of the whole mechanism. These are *network load detection* (NLD), *channel allocation and deallocation* (CAD), *path update* (PU) and *channel changing* (CC) components.

The basic idea of LPMC is to use the same channel for all nodes in a single path. Nodes do not need to switch their channel to receive and forward packets. Therefore, the channel used by a neighbor of the sink (one-hop away node) is to be used by all nodes that forward their data through this node. We call this node as the *channel deciding node* (CDN). Figure 1 shows a typical WSN environment where a set of nodes send their data to the sink using a tree. There are four CDN nodes in the figure ($c_1 - c_4$). The nodes those forward their data through a CDN creates a *tree-branch* (TB), and all nodes within a TB use the same channel. When a channel assignment takes place, all nodes in a TB change their channel. However, multiple TBs can use one channel. Figure 1 shows four TBs ($tb_1 - tb_4$) rooted at four CDNs.

B. Network Load Detection (NLD)

LPMC assumes that the network is not loaded as long as the application's *reliability* is met. We define the reliability of a WSN application as the ratio of the number of packets received by the sink to the number of packets sent by the sources. Furthermore, LPMC aims at estimating the active load at the sink.

LPMC's load detection mechanism is based on the following intuition: network (or a part of it) is not heavily loaded as long as the packet loss rate is acceptable. This permits the packet losses due to poor wireless link, medium contention, and transient congestion. When the network load increases, the packet loss rate also increases (or the interval between successive losses decreases). LPMC therefore uses the *average loss interval* as an active network load indicator.

The sink maintains a list of flows for each TB. Furthermore, it maintains a per-flow list of missing packets and received packets based on the sequence number of the packets. Because the packets of a flow are forwarded in a single path, the reception of an out-of-order packet indicates a packet loss. Suppose, the sequence number of the packets of the mth and the $(m+1)$th loss events of the ith flow are s_m and s_{m+1}, respectively. Denoting $d_{i,m}$ as the length of the mth loss interval of the ith flow, we have $d_{i,m} = s_{m+1} - s_m$.

There exist many techniques in the literature to measure the average loss intervals. However, we choose the *weighted average loss interval (WALI)* method discussed in [5] over others, because of its robustness in the choice of parameters. Then, for the last n losses, the average loss interval for flow i, $\hat{d}_i$, is calculated as

$$\hat{d}_i(1,n) = \frac{\sum_{m=1}^{n} d_{i,m} w_m}{\sum_{m=1}^{n} w_m}$$

$$\hat{d}_i(0,n-1) = \frac{\sum_{m=0}^{n-1} d_{i,m} w_m}{\sum_{m=1}^{n} w_m}$$

$$\hat{d}_i = \max\left[\hat{d}_i(1,n), \hat{d}_i(0,n-1)\right],$$

where $d_{i,0}$ is the number of successfully received packets since the most recent loss, and w_m is the weight assigned to the mth loss interval. We have used $n = 10$, and $w_m = 1/m$ as our parameters. Note that a smaller value of m indicates a recent loss interval, and thus, the parameters give greater weight to recent loss intervals than distant loss intervals.

The reliability of the ith flow, denoted as r_i, can be calculated as, $r_i = 1 - 1/\hat{d}_i$. If r_i is less than the required reliability of the application, R_{req}, for any flow, we say that the channel used by the ith flow is overloaded.

The NLD also measures the active load of the network (i.e., the number of packets sent per unit time). The sink uses a timer for this. When the timer expires, it finds the sequence number of the most recently received packet of each flow, and restart the timer. If the two most recent sequence numbers of the ith flow are s_1 and s_2, the number of packets sent for the flow is $S_i = s_2 - s_1$. The total load of a tree-branch for F flows is: $\sum_{i=1}^{F} S_i$. The average instantaneous load of a TB is measured using EWMA.

C. Channel Allocation and Deallocation (CAD)

The channel allocation and deallocation component assigns the channels for the TBs. When an overloaded channel is used by more than one TBs, CAD assigns a channel (lightly loaded or unused) to one of the TBs. In contrast, when the network load decreases, it removes channel(s). However, if an overloaded channel is used by only one TB, it cannot allocate another channel to the same TB.

The CAD mechanism maintains two lists. The channel list with fields $<$ *channel_id*, *status*, *max_load*, *cur_load*, *rem_load* $>$, and the TB list with fields $<$ *tb_id*, *avg_load* $>$. The *status* of a channel is either *used* or *unused*. The CAD periodically obtains the average load (*avg_load*) of each TB from NLD. The current load (*cur_load*) of a channel is the sum of the loads of the TBs using this channel. The maximum load (*max_load*) of a channel is the so far mostly supported load by the channel. When a channel is overloaded, the *max_load* is updated by the CAD; if the *cur_load* is higher than the *max_load*, *cur_load* becomes the *max_load* of the channel. The reaming load (*rem_load*) of a channel is the

Algorithm 1 Channel Allocation and Deallocation (CAD)

```
1: Input: $status$[ ], $max\_load$[ ], $current\_load$[ ],
2:      $remaining\_load$[ ], $avg\_load$[ ]
3: ChannelAllocation (Channel $i$)
4: Find the no of TBs, $N$, those use channel $i$.
5: if $N \leq 1$ then Call path update component and return.
6: Find the TB with maximum loss rate, $tb$.
7: for Each used channel $j = 1$ TO $K$, Except channel $i$ do
8:   if $avg\_load[tb] \leq 0.9 \times rem\_load[j]$ then
9:     Assign channel $j$ to $tb$ and return.
10: end for
11: if unused channel available then assign it to $tb$.

12: ChannelDeallocation ()
13: for Each Channel $i = 1$ TO $K$ do
14:   for Each Channel $j = i + 1$ to $K$ do
15:     if $curr\_load[i]$ + $curr\_load[i] \leq max\_load[i]$ then
16:       Assign Channel $i$ to the TBs using Channel $j$
17:       $cur\_load[i]$ += $cur\_load[j]$, $status[j]$ = $unused$
18:     end if
19:   end for
20: end for
```

difference between the max_load and cur_load. Algorithm 1 shows the detailed operation of CAD.

D. Path Update (PU)

The path update module is invoked in a situation when an overloaded channel is used by only one TB. In this case, CAD cannot allocate a new channel to an overloaded TB since LPMC claims that a single TB should be using only one channel. Therefore, the path update module divides this overloaded TB into two TBs and allocates a new channel to the overloaded TB. Note that nodes in the new TB are also assigned a single channel and therefore, do not need to switch between channels while receiving or forwarding a packet

The sink first sends an unicast *path update message* (PUM) to the CDN of the TB. The PUM contains the following fields $< source$, $destination$, $type$, $tb\_ID >$. The $type$ field has the value 1 or 2, and sink sets it to 1. The $source$ is the address of the PUM sender/forwarder, and $destination$ is the address of an upstream node of the sender. LPMC assumes that every forwarding node keeps a list of upstream nodes.

A PUM receiver either forwards it (when $type$ value is 1) to its upstream node(s) or generates a PUM reply (when $type$ value is 2). If a PUM forwarder has more than one upstream nodes, it changes the $type$ value to 2, and forwards the PUM to randomly selected half of the upstream node(s). Because two or more branches join in this node, LPMC creates a new path for one of the branches, and creates a new TB.

If a node receives a PUM with $type$ value 2, it is forced to send a reply. This PUM reply (PUMR) creates a path from the node to the sink and divides the TB. The PUMR has the following fixed fields set by the source of the PUMR: $< source$, $destination$, $tb\_ID >$. The $source$ is the address of PUMR generator, $destination$ is the broadcast address, and tb_ID is copied from the PUM. Every PUMR forwarding node (including the PUMR generator) appends the following fields to the PUMR: $< forward\_node\_addr$, $channel\_ID$, $hop >$, where $forward\_node\_addr$ is the address of the PUMR forwarder, $channel\_ID$ is its current channel, and hop is the hop count of the node from the sink.

Leaf nodes can only forward the PUMR. A leaf node can however forward the PUMR, if its hop count to the sink is not greater than the value of the hop field in the last entry of the PUMR. PUMR is broadcasted in all channels one after another to find a leaf node. A node broadcasts the PUMR in a channel, and hears the channel for some period. If no node forwards the PUMR within the period, it broadcasts in another channel. Therefore, nodes first forward the PUMR in the receiving channel to ensure that the upstream node (previous broadcaster) hears it. Eventually, the PUMR is received by the sink. If the sink receives more than one PUMRs, it chooses the shortest path. The reverse path appended in the PUMR creates a new TB, and CC assigns a new channel.

E. Channel Changing (CC)

LPMC uses explicit control message to change the channel. Two types of *channel changing messages (CCM)* are used: i) CCM-1 changes the channel of all nodes in a TB, ii) CCM-2 assigns a new channel in response to a path update.

To change a channel of a TB, the sink sends an unicast CCM-1 to the CDN of the TB. The message has the following fields $< sender$, $receiver$, $new\_channel >$. All nodes broadcast the CCM-1, except the sink. After receiving a CCM-1, every node checks the $sender$ field. If a node receives the CCM-1 from its downstream node, it forwards it; otherwise it discards it. While forwarding the CCM-1, each node replaces the $sender$ field by its own address, and puts the broadcast address in the $receiver$ field. When a forwarder hears at least one of its upstream nodes has forwarded the same message (by snooping), it changes its own channel to the $new\_channel$.

The CCM-2 message is unicasted with the following fields $< sender$, $receiver$, $new\_channel$, $destination >$. The $destination$ is the address of the PUMR generator. CCM-2 also includes the reverse path from the sink to the $destination$, and the channel ID of each intermediate node. Every node forwards the CCM-2, and changes it channel to $new\_channel$. When the $destination$ receives the CCM-2, it converts it to CCM-1 and broadcasts it. Therefore, path update and channel changing happen simultaneously.

III. PERFORMANCE EVALUATIONS

We have performed extensive simulations to evaluate the performance of LPMC in NS-2. A network of area $200m \times 200m$ is used, and 250 nodes are placed in uniform random distribution. The transmission and interference ranges of the nodes are set to 30m and 45m, respectively. Link bandwidth is set to 250 kbps, and 6 orthogonal channels are used.

Figure 2 shows the performance comparison for a varying number of sources from a randomly selected location. Each

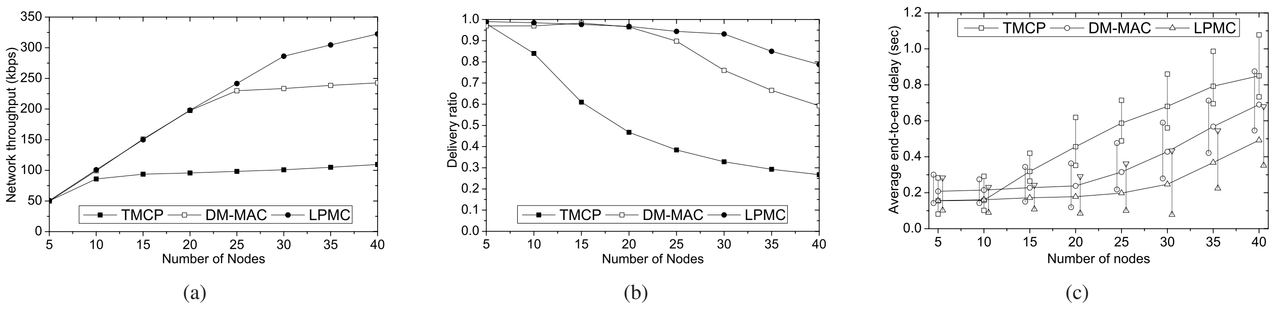

Fig. 2. Performance with varying number of sources from a randomly selected location: (a) average network throughput, (b) delivery ratio, and (c) average end-to-end delay (vertical lines show the max-min variations).

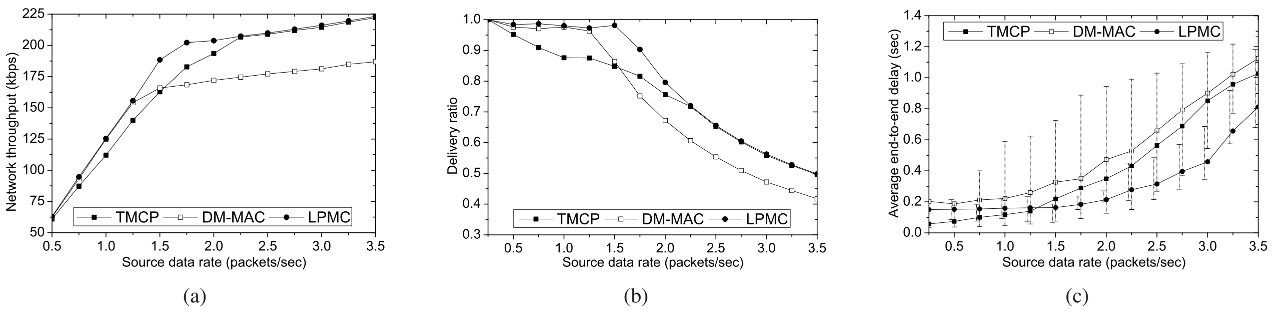

Fig. 3. Impact of channel quality and external interference on the performance: (a) average network throughput, (b) delivery ratio, and (c) average end-to-end delay. All nodes send data to the sink.

source generates data at a rate of 20 packets/sec. Due to the static channel allocation, some of the channels are kept idle in TMCP, and it achieves the minimum throughput with more than 10 sources. DM-MAC requires channel switching when more than one channels are active, and the network throughput is less than LPMC. Fig. 2(b) shows the delivery ratio of different mechanisms, and LPMC achieves the highest delivery ratio. Fig. 2(c) shows the average end-to-end delays of the packets. Because TMCP does not utilizes all the channels, network becomes overloaded when number of nodes increases, and it experiences the maximum delay. The channel switching increases the medium access time in DM-MAC, and thus, the delay is higher than LPMC.

Figure 3 demonstrates the impact of channel quality and external interference. We vary the packet loss rate randomly between 10 to 80 percent. All nodes send their data to the sink. Both LPMC and DM-MAC assign a new channel, if the channel quality degrades; whereas nodes using the interfered (or low quality) channel cannot deliver their data in TMCP. Therefore, LPMC and DM-MAC achieve the required delivery ratio (and so, higher throughput as in Fig. 3(b)) as long as the offered load is less than or equal to the aggregate channel capacities. However, LPMC achieves higher throughput and reliability than DM-MAC, because it does not need channel switching. In contrast, the throughput in TMCP increases for higher source rates. Fig. 3(c) shows the average end-to-end delays. DM-MAC requires the maximum delay due to the channel switching. Because all channels are active in TMCP, the delay is lower than LPMC for low data rates. Though nodes using bad quality channels experience higher delay. The vertical lines show the maximum and minimum average delays among the channels. Furthermore, TMCP experiences higher delay than LPMC for higher source rates, because the queueing and medium access delay are very high for low quality channels.

IV. Conclusions

In this paper, we have designed a load-adaptive multi-channel communications system for WSNs. LPMC controls the channel allocation and deallocation at the sink, and dynamically adds or removes channels. The dynamic channel allocation utilizes the limited channels of WSNs very efficiently, and adds channel only when it is necessary. Finally, simulation results demonstrate that LPMC outperforms the existing mechanisms.

References

[1] G. Zhou, C. Huang, T. Yan, T. He, J. A. Stankovic, and T. F. Abdelzaher, “MMSN: Multi-frequency media access control for wireless sensor networks,” in *INFOCOM 2006*, April 2006, pp. 1–13.

[2] J. Zhang, G. Zhou, C. Huang, S. Son, and J. Stankovic, “TMMAC: An energy efficient multi-channel mac protocol for ad hoc networks,” in *IEEE ICC ’07*, June 2007, pp. 3554–3561.

[3] H. K. Le, D. Henriksson, and T. Abdelzaher, “A practical multi-channel media access control protocol for wireless sensor networks,” in *ACM/IEEE IPSN ’08*, 2008, pp. 70–81.

[4] Y. Wu, J. Stankovic, T. He, and S. Lin, “Realistic and efficient multi-channel communications in wireless sensor networks,” in *INFOCOM 2008*, April 2008, pp. 1193–1201.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation-based congestion control for unicast applications,” in *SIGCOMM*, 2000.