# A QoS-Aware Data Aggregation in Wireless Sensor Networks

Jongsoo Jeong†*, Jaeseok Kim†*, Haeyong Kim*, Sangcheol Kim*, and Pyeongsoo Mah*

†*Computer Software and Engineering, University of Science and Technology,*
*113 Gwahak-ro Yuseong-gu Daejeon, Republic of Korea*

**ETRI, 161 Gajeong-dong Yuseong-gu Daejeon, Republic of Korea*

jsjeong@etri.re.kr, jaeseok@etri.re.kr, haekim@etri.re.kr, sheart@etri.re.kr, pmah@etri.re.kr

*Abstract*— **Data aggregation has been one of important key techniques to increase energy efficiency and bandwidth utilization in wireless sensor networks. There were many research works in this area, but most of them have a disadvantage that cannot optimally meet practical requirements, such as the heterogeneity of application data and latency requirements. In this paper, we propose a novel and simple data aggregation protocol, referred to as Lump, which enables to support QoS requirements of applications. For this purpose, it prioritizes packets for differentiated services and facilitates aggregation decision. Its architecture has a cross-layered design that mitigates overheads of in-network processing, and it is completely an independent module residing on between data-link layer and network layer so that it can be applicable to a variety of applications. Our experiments show that, when an intermediate node has four neighbors, Lump can reduce radio traffic up to 30% while satisfying QoS requirements.**

***Keywords*— wireless sensor networks, in-network processing, data aggregation, QoS, MAC protocol**

## I. Introduction

Wireless Sensor Networks (WSNs) have been widely used for various kinds of applications including environmental and habitat monitoring [1], underwater monitoring [2], and surveillance system [3]. Recent advances in low-power design and small micro sensor nodes enabled the exploration of such applications, however the resource constrained nature of sensor nodes is still the most challenging feature. Developers have been trying to maximize energy efficiency so as to increase network lifetime and decrease the cost of deployment. Since each application differs in features and requirements, (e.g., the frequency of data reporting or the number of sensor nodes) the system design of a WSN depends significantly on the application.

Data aggregation [4] has been a key technique that minimizes energy consumption by reducing radio communication while maximizing network utilization by combining data coming from different sources into a single frame. Since the computation cost is much lower than the communication cost [5], it is worth exploring such in-network processing which resorts to tradeoff between energy and latency. The more packets are aggregated, the more energy will be conserved while increasing delay. In this respect, it is quite important to consider how much the tradeoff will impact the Quality of Service (QoS). The characteristics of data are mostly not identical in realistic field. Rather, there are more than at least two types. For example, consider a forest-monitoring application. In this case, there may be three types of packets - network maintaining messages, periodical reports of temperature and humidity values, and notification of fire. Each type can be classified as delay-tolerable, delay-bounded, and delay-critical, respectively. Both application developers and users may expect these properties to be ensured. Thus, in this sense, when data aggregation scheme is used, differentiated services need be provided to users.

To provide the QoS services in WSNs, it is necessary to analyse the application requirements [6]. Since packets are of many types, it is impossible to analyse their QoS requirements individually without the help of the application.

There are some difficulties for supporting QoS in WSNs. 1) Sensor nodes have severe resource constraints, which implies that computation speed should be as fast as possible. This requires that a running protocol on a sensor node must be simply designed. 2) It should be able to support different QoS levels associated with various properties of network: multiple sinks, multiple traffic types, and packet priority. 3) Practical features of WSNs such as dynamic network topology, mobility of nodes, and large scale of deployment increase both complexity and uncertainty. Consequently, it can exacerbate the system performance, which in turn will result degradation of QoS. The entire design of our protocol has been prototyped with careful consideration of the above-mentioned challenges for QoS support.

In this paper, we introduce a QoS-aware data aggregation protocol, which we call Lump. Lump protocol aggregates MAC headers into a single common header and concatenates application-data of each packet. In order to eliminate application specific context, it does not fuse application-data. This lossless aggregation scheme enables Lump to aggregate packets, irrespective of their types. Lump utilizes various properties of packets not only to support QoS, but also to maximize degree of aggregation (DOA). In order to make overall mechanism simple, we take advantage of cross-layer design [8][9] which provides upper layer to specify service requirements on each packet, i.e., Lump protocol gives authority to the application. Therefore, our protocol accurately understands various QoS requirements. It can also keep simplicity since it does not need additional computation to select priority level of packet.

The remainder of this paper is structured as follows. We begin a discussion of related work in Section II, then present the design of Lump Protocol in Section III. We describe our prototype implementation and experimental results in Section IV, and future work in Section V. We give conclusions in Section VI.

## II. Related Work

Data aggregation has been a challenge issue in WSNs. There were a lot of efforts in facilitating data aggregation. The aggregation functions relied on manipulating data to some extent, needing semantic information on what the actual aggregation function is. In this sense, these are all application-dependent aggregation operations. The Application-Independent Data Aggregation [7] that isolates aggregation decisions from the application context, and uses dynamic feedback scheme compared with fixed- and on-demand-data aggregation scheme. It is generalized to be utilized over a wide range of applications without incurring the costs of rewriting other components to support application-specific logic. Although AIDA efficiently conserves energy as well as minimizing average end-to-end delay, the feedback scheme is too complex for severely resource constrained sensor node. Specifically, it does not take into account the diversity of packets.

Another researcher considered involving the collection of both real-time and non-real-time data [10]. The suggested schemes have taken into account some applications of energy-efficient delay-constrained data aggregation include real-time target tracking in battle environments and critical relaying of after-shock events in disaster management environments. To balance energy saving and queuing, a scheduling algorithm that is based on weighted fair queuing (WFQ), is employed. However, this algorithm has application-dependency since they manipulate application data while processing aggregation. Moreover, as they strive to derive condition which is denoted as acceptable longest path, for on-time packet delivery over an aggregation tree, it results in longer delivery latency.

Data aggregation is a method to conserve energy by tradeoff between energy and latency. For WSNs, two primary service differentiation parameters are reliability and latency. Wireless links as communication medium of WSNs are more error prone than wired links are. In AFS [11], not only the reaching probability of packet but also latency can be affected by this medium property. In order to satisfy both of them, QoS mechanisms should provide differentiated service such as more acknowledgements, more transmission of packets, and stronger FEC for higher priority packets. However, this end-to-end reliability is a responsibility of transport layer thus beyond our scope in this paper. We assume that the reliability can be supported enough from hop-by-hop error control mechanisms, e.g., CRC, retransmission of data-link layer.

## III. Protocol Design

### A. *Goals of Lump Protocol*

The goals of Lump Protocols are as followings:

- Supporting QoS according to the priority of packet
- Supporting multiple sinks and multiple traffic types
- Maximizing degree of aggregation

We examined some requirements to achieve these goals.

***1) Prioritization***: In order to support QoS, Lump protocol must recognize packet priority. Sensor nodes are too resource constrained to decide packet priority statistically e.g., meter and marker mechanisms of DiffServ [14]. Thus, it is necessary to take simpler approach.

***2) Multiple Sinks:*** Traditional data aggregation protocols are not only limited to support multiple sinks but also expected to decrease DOA, since they operate on top of network layer and do not aggregate packets destined for different nodes. Considering supporting multiple sinks as an essential paradigm, we need Lump layer to aggregate packets that share a route even instantly.

**3)** ***Multiple Traffic Types:*** To support multiple traffic types, aggregation mechanism must not fuse or manipulate application data. Traditional mechanisms perform system-level fusion of the collected data for a certain command. This strategy is only for some possible situation, i.e., it is application dependent, and not able to support multiple traffic types. In other applications, it may be more desirable to deliver application data to the sink node. Thus, we need to perform lossless aggregation.

***4) Maximization of DOA:*** In order to maximize DOA, each sensor node needs to adjust waiting time for aggregation based on its position or role in the network. If a node is a leaf node located at the end of network, it does not relay any packet. Thus, there is no need to wait but to send packet generated by itself immediately. On the other hand, if a node is a router, it must wait for the other packets to aggregate. We need to classify such condition to adjust waiting time. Taking into account the above requirements, we designed Lump protocol as a QoS-aware mechanism.

### B. *Packet Priority*

Lump layer receives packets with their priorities from upper layer directly. This cross-layer design is the simplest approach to recognize packet priority. The priority represents tolerable end-to-end latency of its packet. For example, if a priority value is '5', end-to-end latency of the packet is tolerable during '5' time units. Accordingly, the highest value '0' means that the packet must be sent immediately. The priority is raised toward the highest value as time goes by. On every router node on the path, Lump protocol decides the waiting time of the packet

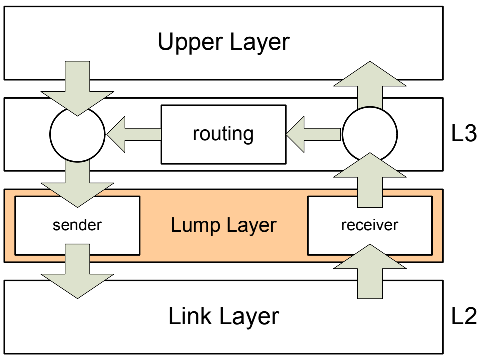

Fig. 1. Architectural Design

corresponding to its priority, which is included in the packet until the arrival at the final destination. With this manner, Lump protocol provides differentiated services for each packet, i.e., it makes delay-tolerable packets to yield network band-width to more delay-critical ones.

## *C. Data Aggregation*

As we previously discussed, supporting QoS is challenging in WSNs, particularly because of various traffic properties and dynamic circumstances of network. Employing data aggregation scheme in such QoS constrained environment has led us to make two decisions.

First, in order to maximize DOA, it is important to decide how long a packet should wait for being aggregated with others. As mentioned above, delay-tolerable packets can be deferred for a certain amount of time. Lump protocol utilizes that time quantum for aggregating packets. This intuitive strategy enables Lump protocol to be lightweight so that little overhead is added to the baseline system.

Second, we designed Lump layer to reside between network layer and data link layer, meaning that the data aggregation scheme is not only independent to the dynamic circumstances of network layer but also able to maintain per-route flow. When packets are being delivered to multiple sinks, routers should maintain per-route flow rather than per-sink flow. If routers aggregate packets based on their sink address, packets destined for different sinks will not be aggregated, which in turn causes reduction of DOA. On the other hand, maintaining per-route flow enables efficient data aggregation even when the packets share the same next-hop in transit. Thus, it is able to support multiple sinks without decreasing DOA.

However, there still exists another problem. Packets destined for different sinks may have different values and different format. Conventional data aggregation mechanisms manipulate application data, using certain operators such as averaging, to reduce data redundancy. This is limited to certain types of data and also not able to aggregate distinct data types. Instead, in order to support multiple traffic types, we design our protocol to aggregate MAC headers and concatenate each application data into MAC payload without data loss. Generally, packets in WSNs contain small sensing data. Relatively, MAC headers consume significant energy as a fixed overhead on every single transmission. Our protocol aggregates their MAC headers into common MAC header so that the transmission rate is decreased.

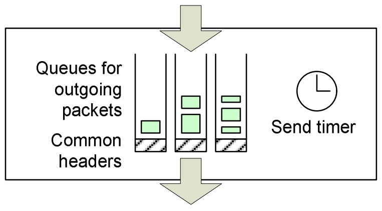

Fig. 2. Sender module

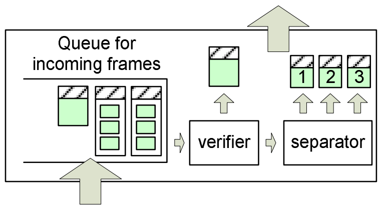

Fig. 3. Receiver module

## *D. Architectural Design*

Fig. 1 shows the architectural design of Lump protocol. Arrows represent the flow of both packets and their priorities. Sender module is designated to intercept all outgoing packets from network layer and responsible for concatenating them into a frame. When the system receives an aggregated frame, receiver module verifies if it contains multiple packets or not, and separates it to individual frames. Then it forwards the frames to network layer. As a result, network layer receives individual frames, including both packets and their priorities, from Lump layer. Then it will either pass the frame to the application or find the next hop addresses where they have to be forwarded to.

Sender module is shown in Fig. 2. It consists of common header structures, send timer, and send-queues that are corresponding to the next hop addresses. Sender module deframes and classifies packets coming from network layer by next hop addresses, then enqueues them to the queue corresponding to addresses. Common header structure maintains common information of the queue. On every enqueue of packets, sender module updates common header structure of the queue. In order to aggregate packets, sender module delays them for the amount of time corresponding to their priorities. Send timer manages and updates the priorities of packets in all queues. When it sends a packet, it aggregates all packets in the same queue into a single payload and links common header and the payload. This aggregated frame is processed as a normal frame by MAC layer.

Sender module schedules frame transmission. Conditions of transmission are as followings:

**1) *Size*:** Before sender module enqueues a new packet, it checks the free space of the queue. If there is not enough space in the queue, at first it empties the queue by sending all of the packets, then enqueues new packet. On the other hand, if the free space fits new one perfectly, sender module enqueues it first, then sends all packets with no additional wait. Because there is no more space to aggregate now.

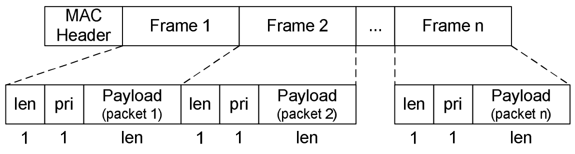

Fig. 4. Frame Format

**2)** ***Position***: Sender module sends packet immediately when the node is not a router, in other words, the node has no more chance to aggregate. In order to recognize its position, sender module distinguishes where packets are generated from. For some packets generated from application layer, sender module tries to finds the queue to store them. If it finds a queue corresponding to the same next hop address, it recognizes that the node is a router. However, if there is no queue to store, sender module recognizes that the node is a leaf node. So it sends the packet immediately. For the other packets coming from network layer, sender module obviously recognizes that the node as router.

**3)** ***Priority***: When a packet is passed to sender module and enqueued to certain queue, sender module checks its priority and decides whether it must be sent now or not. If the priority is the highest, sender module sends it immediately.

**4)** ***Time-out***: Sender module sends the packet when its weighted waiting time in network reaches up to its tolerable end-to-end latency. The send timer checks it on every updates of packet priorities.

Fig. 3 shows the receiver module. It consists of a verifier, a separator, and a queue to store received frames temporarily. When a frame is passed from data-link layer, then receiver module enqueues it at first, and processes frames in the queue sequentially. Verifier dequeues a frame and verifies whether it is aggregated by our protocol or not. If it is, verifier passes it to the separator for next processing, otherwise passes it directly to upper layer. Aggregated frames are separated to individual frames and passed to network layer sequentially by separator.

### *E. Frame Format Details*

An aggregated frame is formed as shown in Fig. 4. Lump protocol aggregates common information of individual packets into one header and concatenates application data with sub headers. Common header contains common information (e.g. source address, destination address) of aggregated frames. Each sub header contains individual information (e.g. length, priority) of each frame.

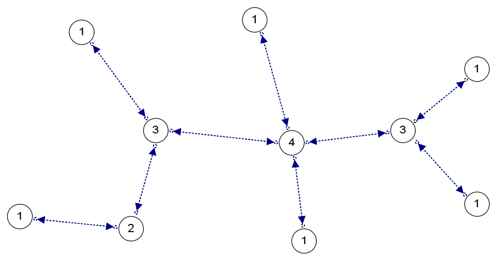

Fig. 5. System model

## IV. Evaluation

We chose to integrate our prototype implementation of Lump protocol in Nano Qplus [12][13] - a multi-threaded, lightweight sensor network operating system. The key design goal of Nano Qplus is ease of use. It has the thread programming model and supports many functions, e.g., message queue, semaphore, user timer and so on. The memory management supports stable thread stack management and dynamic memory allocation scheme in a small sensor node. Nano Qplus has NanoMAC which is a very small sized MAC, but provides most of basic MAC functions, such as data retransmission, acknowledgement and CCA for wireless communication. In order to keep backward compatibility, we retained existing API related to MAC, and add new API related to Lump Protocol. Table I shows the existing and the newly added APIs of Nano Qplus for Lump protocol.

### *A. Evaluation Setup*

To evaluate the performance of Lump protocol, we have constructed a system consists of 10 sensor nodes. The hardware mote used is named Ubi-coin — a Telos B-based hardware platform that has MSP430 microcontroller, CC2420 RF transceiver and a few peripherals. We took a tree-based approach as a maiden work, because it would optimally affect our evaluation due to additional workload. And we assume that the network topology is initially set up and would not change, thus we can exclude unexpected interference such as complicated routing protocol. However, we did not just follow the traditional tree-based data flow. Rather, to support multiple sinks, we take into consideration that variant data from a source may not only flow upwards the sink but also go downwards, i.e., packet can be forwarded to any direction of the next hop in the system. This is possible even when the data flow is fixed upwards a single sink node if we use dynamic routing protocol such as AODV.

TABLE I
API LIST RELATED WITH NANOMAC AND LUMP

| | **Prototypes** | **Description** |
|---|---|---|
| **NMAC** | `void mac_init(UINT8 channel, UINT16 panaddr, UINT16 myaddr);` | Initialize MAC channel and addresses |
| | `void mac_set_rx_cb(void (*func)(void));` | Set callback function to receive frame. |
| | `BOOL mac_tx_noack(MAC_TX_INFO *frame);` | Transmit frame without ACK. |
| | `BOOL mac_tx(MAC_TX_INFO *frame);` | Transmit frame with ACK. |
| | `BOOL mac_rx(MAC_RX_INFO *frame);` | Read frame. |
| **Lump** | `BOOL mac_tx_with_prio(MAC_TX_INFO *frame, PKT_PRIORITY prio);` | Transmit frame with priority. |
| | `BOOL mac_rx_with_prio(MAC_RX_INFO *frame, PKT_PRIORITY *prio_ptr);` | Read frame with priority. |

Fig. 5 shows our simple network structure. Each number on the nodes represents their routing entity, i.e., number of neighbor, meaning that we can expect relative degree of traffic in terms of additional workload forwarding packets between the neighbors. Both sided arrows connecting the nodes represent our considerations, the possible direction of data flow. All nodes have a static routing table and are aware of the network size. Each node is designated to periodically send 10 packets containing application data 20 bytes long every minute. To evaluate overall performance related to the parameters, packets are sent toward randomly selected final destination with randomly generated priority that represents a time unit of a second. Note that, in practical use, application layer is responsible for setting the packet priority thus determine DOA on demand. To make sure to take advantage of Lump protocol, it is important to carefully choose the level of packet priority properly.

## *B. End-to-End Latency*

In our evaluation, we mainly focus on the goal directed to support QoS requirements specified by upper layer. We examined packet transmission, reception, and forwarding of each node respectively, until the pure packet generation (and transmission) exceeds more than 2000 packets. In order to measure both the expected efficiency and overhead precisely, we compared Lump-added MAC protocol with unmodified MAC protocol, NanoMAC, by using the rest of conditions identical. Fig. 6-7 show the change of average end-to-end latency as priority and hop count vary. As both the packet priority and hop counts increase, the latency increases accordingly. However, note that Lump protocol turned out to ensure approximately half the specified priority on average (Fig. 6). We were also able to make certain that not a single packet exceeds its designated priority.

Fig. 8 illustrates, when using Lump protocol with priority 0 for critical message, how the average latency is affected in comparison with baseline of NanoMAC usage. The average latency is more or less the same as NanoMAC. The little difference is mainly due to the additional computation within the Lump layer. The results demonstrate that the tradeoff between latency and energy efficiency are feasible without significantly affecting the system performance in terms of the energy efficiency corresponding to the reduced radio usage.

## *C. Performance*

In our model, we assumed that the number of routing entity would directly affect each node in its degree of traffic. Fig. 9 shows the reduction ratio of radio transmission under the varying degree of traffic. At one extreme degree in our model, at the rightmost vertex in the graph, Lump protocol has reduced 27% of packet transmission, which represents the reduced amount of redundant MAC header. The graph demonstrates that our design has achieved graceful degradation against increasing traffic load. This is a truly remarkable result since, in our model, the packets flow randomly into all possible directions, not the one-way traffic. However, when the number of neighbor is one (meaning that

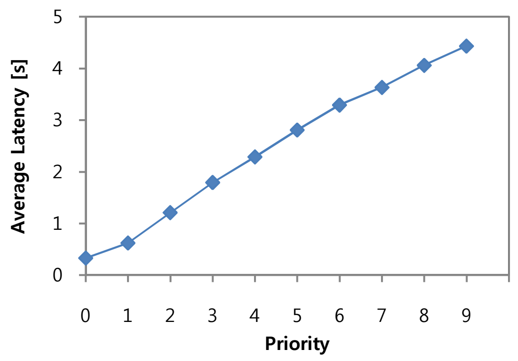

Fig. 6. Average latencies with the order of priority

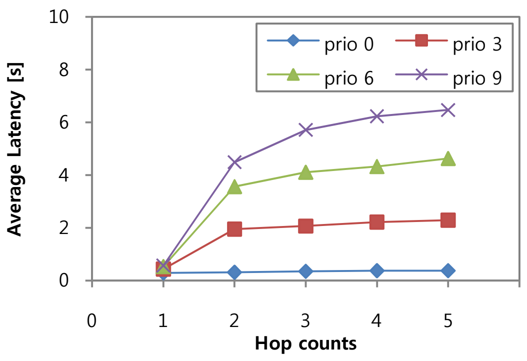

Fig. 7. Average latencies with hop counts

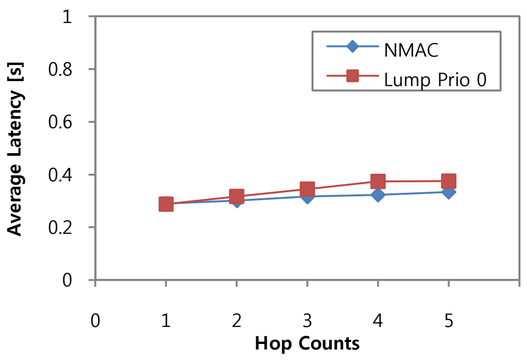

Fig. 8. Latency comparison: NanoMAC vs. Lump with priority 0

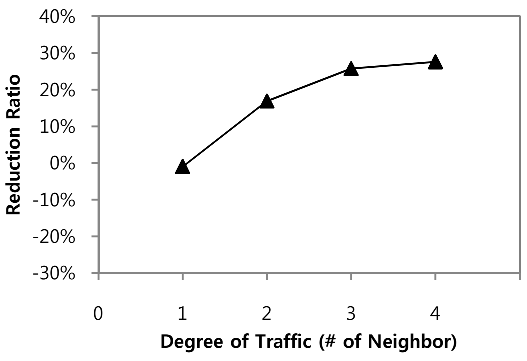

Fig. 9. The number of transmission reduction ratio for three hours

it is a leaf node), the reduction ratio results in 0%. This is because leaf nodes are not engaged in data aggregation mechanism. Overall, Lump protocol does not harm the system performance to serve two goals: Energy efficiency and QoS support.

## V. Future Work

Although our protocol is well-performed for the goal we have targeted, there are some immaturities in which we hope to carry out future work.

First, the larger the packet size becomes, the more congestion would occur due to the limited channel bandwidth. Consequently, Power consumption may increase owing to retransmission trials. Thus it is not always the best plan to maximize the size of packets [5]. However, results have shown that packet retransmission rate of the two protocols are observed as the same, approximately 0.1%. One possible reason is that the designated traffic in the system does not exceed maximum bandwidth capacity. Since we decided to initially minimize unexpected interferences, we started with a moderate traffic model. For now, we plan to expand the evaluations with other traffic models that are dynamic and malicious, in order to enhance stability of Lump protocol.

Second, when a priority value is 5, the corresponding packet is supposed to be delivered in 5 seconds. But the average end-to-end latency has resulted in about half the priority values. Indeed, the distribution of the results spread out randomly. The reason is that the aggregation queues are based on the next-hop address so that delay-tolerable packets may be sent earlier with a delay-critical packet when both are aggregated in the same queue. Although the system satisfies application requirements, neither accuracy nor precision has been met. We aim to reconcile the error between application decisions and the results of services so as to attain consistently high DOA.

Finally, since we designed Lump layer to be independent, more optimization will be available by cooperating with the other protocols such as data-centric routing. We will investigate such advantages.

## VI. Conclusion

This paper presented Lump protocol, running independently between network layer and data-link layer, as a way of data aggregation. To the best of our knowledge, the other data aggregation mechanisms are interested in complex algorithms to increase DOA without knowing application demands. This may, in turn, lead to misdirected performance and not fulfill the application's expectations. Our design concept which employed cross-layer manner was straightforward. Therefore we have minimized computation-bound workload and focused on adaptability and reliability.

Lump protocol is mainly targeted to aggregate individual packets sharing the same next-hop address into a single frame to reduce radio bandwidth usage while supporting QoS requirements of applications. The Lump module tries to aggregate packets as long as each the potential waiting time remains. It is designed to be adaptable for both end-nodes and router nodes.

We have demonstrated that our design meets the goals. Our prototype designed on top of Nano Qplus, a well-made WSN operating system, outperformed its baseline. Results have shown that Lump protocol offers sufficient level of QoS support while reducing the redundant MAC headers. Despite the fact that the packets have been randomly destined for the multiple sinks, Lump protocol gracefully reduces radio communication rate with little overhead.

## References

[1] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgessm T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods," in *Proceedings of the 3rd ACM international conference on Embedded networked sensor systems*, 2005.

[2] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proceedings of the 3rd ACM international conference on Embedded networked sensor systems*, 2005.

[3] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. Stankovic, T. Abdelzaher, and B. Krogh, "Lightweight detection and classification for wireless sensor networks in realistic environments," in *Proceedings of the 3rd ACM international conference on Embedded networked sensor systems*, 2005.

[4] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proceedings of the 22nd IEEE international conference on Distributed Computing Systems Workshops*, 2002.

[5] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," in *Communications of the ACM*, 2000.

[6] D. Chen, and P. K. Varshney, "QoS Support in Wireless Sensor Networks: A Survey," in *Proceedings of the International Conference on Wireless Networks*, 2004.

[7] T. He, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "AIDA: Adaptive application-independent data aggregation in wireless sensor networks," in *ACM Transactions on Embedded Computing Systems (TECS)*, vol 3, pp. 426-457, 2004.

[8] S. Shakkottai, T. S. Rapaport, and P. C. Karlsson, "Cross-layer Design for Wireless Networks," in *IEEE Communications Magazine*, vol 41, pp. 74-80, 2003.

[9] V. Srivastava, and M. Motani, "Cross-layer design: a survey and the road ahead," in *IEEE Communications Magazine*, vol. 43, pp. 112–119, 2005.

[10] K. Akkaya, M. Younis, and M. Youssef, "Efficient Aggregation of Delay-Constrained Data in Wireless Sensor Networks," in *Proceedings* of *the 3rd ACS/IEEE Internaltional Conference on Computer Systems and Applications*, 2005.

[11] S. Bhatnagar, B. Deb, and B. Nath, "Service Differentiation in Sensor Networks," in *Proceedings of Wireless Personal Multimedia Communications*, 2001.

[12] S. C. Kim, H. Y. Kim, J. K. Song, M. S. Yu, and P. S. Mah, "Nano Qplus – A Multi-Threaded Operating System with Memory Protection Mechanism for Wireless Sensor Networks," in *1st China-Korea WSN Workshop (CKWSN)*, Chongqing. China, 2008.

[13] Nano Qplus, http://qplus.or.kr/

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," in *RFC 2475, IETF*, 1998.