Congestion Performance Improvement in Wireless Sensor Networks

Junjie Xiong [1] **Michael R. Lyu** [1,2] **Kam-Wing Ng**[1]

[1]**Department of Computer Science & Engineering**
The Chinese University of Hong Kong
Shatin, Hong Kong
[2]**School of Computer Science**
National University of Defense Technology
Changsha, China.
{**jjxiong, lyu, kwng**}**@cse.cuhk.edu.hk**

***Abstract*— Wireless sensor networks (WSNs) have expanded their monitoring and tracking applications into wide areas, such as civil, military, and aerospace fields. Despite their important role, their performance under harsh conditions still remains to be improved. Specifically, when a WSN suffers congestion, the base station (BS) can hardly receive any data from the far-away sensor nodes while it still gets a moderate amount of data from the near-by nodes. Since the goal of WSN applications is to monitor the whole designated area, such unfairness is not acceptable. In addition, the average latency during congestion is intolerably long, failing the data freshness requirement of WSN applications. Although the fairness and latency performance of congested WSNs is very crucial for WSN applications, their degradation during congestion is usually ignored by most current congestion control methods which focus more on avoiding or recovering from congestion rather than on the congestion process itself. To improve the performance during congestion, we propose a multi-queue-LIFO (Last-In, First-Out) approach. Instead of the intuitively and frequently employed FIFO (First-In, First-Out), we are the first to use LIFO to improve delay and fairness in congested WSNs. To further enhance the fairness performance, we divide the single queue in each node into multiple weighted sub-queues logically, and forward packets in each sub-queue based on its weight. This method balances the data reception from other nodes at the BS. Both theoretical analysis and extensive experiments verify the performance improvement of our approach.**

TABLE OF CONTENTS

1. INTRODUCTION

Wireless sensor networks (WSNs) are composed of many low-cost wireless sensor nodes. These sensor nodes are distributed in the designated region and self-organize into a wireless network. They are used in many applications for monitoring or tracking events of interest in the regions [1]. WSNs can also facilitate the aerospace applications, such as space exploration [2]. For example, it is difficult to monitor the moon where human cannot normally act. The current methods are using remote sensing satellites and mobile robots [2]. Nevertheless, these methods are expensive and cannot retrieve detailed data about a large region. Hence, deploying WSNs on the moon to collect the environmental information, such as temperature, sunlight intensity, and seismic status, is more efficient.

The data generated by each sensor node that record the physical events of interest on the region are required to be promptly transmitted to the BS (base station) [1]. The applications fail if the BS receives less data generated from the far-away sensor nodes than those from the near-by sensor nodes in a required period, or if the data cannot arrive at the BS in time. As the experiments in [3] show, the congested sensor network mostly delivers data from nodes one hop away from the sink, hence it is very unfair for far-away sensor nodes. In addition, the average delay of transmitting the data to the BS becomes very long for WSN applications. Although, under normal conditions, the failure will not occur, it happens when congestion takes place. Nevertheless, congestion is inevitable in WSNs, due to the unstable environment of wireless communications and the traffic dynamics (traffic burst and many-to-one multi-hop traffic load) [3]. In aerospace applications, the environment may be more unstable and thus the congestion will be more likely to happen.

Despite that congestion is unavoidable, many current congestion-related methods are still proposed to settle the congestion problem in WSNs [3] [4] [5]. Unfortunately, they usually ignore the performance during congestion. For example, the congestion avoidance and congestion recovery methods either try to avoid or recover from the congestion, hence they can do nothing to the performance degradation but wait for recovery when congestion does take place. Actually, performance degradation during congestion cannot be dismissed in WSN applications. All the data collected in the WSN are real-time and should be delivered in time, and persistent packet drop from the same sensor nodes means blind points in their monitored areas.

In addition, no method considers improving the congestion performance of WSNs in the aspect of the queueing mechanism at the network layer. No LIFO (Last-In, First-Out) queueing mechanism at the network layer is considered for WSNs, while most existing WSNs use FIFO (First-In, First-Out) intuitively [6] [7] [8] [9]. The traditional networks usually enforce FIFO instead of LIFO because they employ

978-1-4577-0557-1/12/$26.00

TCP protocols and thus require proper packet order to indicate packet drop and the corresponding retransmissions [10]. However, WSNs do not have to use FIFO instead of LIFO because they seldom use TCP protocols, and the crucial communication protocols are the medium access control (MAC) and routing protocols [6].

In this paper, we attack the performance degradation challenge during network congestion for WSNs with multi-queue-LIFO approach. Unlike most existing mechanisms that rely heavily on controlling the packet sending rate (flow control) [3] [4] so as to avoid congestion, our approach is triggered once congestion happens. Instead of using the more traditional and intuitive FIFO in WSNs, we are the first to employ LIFO (Last-In, First-Out) queueing mechanism to enhance the delay and fairness among the nodes purely in the network layer. We avoid adjusting the packet sending rate because it is generally determined by the WSN monitoring environment and application parameters, which, once programmed into sensor nodes, require extra efforts and resource to get changed or updated [11]. As a result, our approach is practical regarding all existing WSN applications. In addition, LIFO works better than FIFO for the real-time properties of WSN applications because it achieves shorter delay in congested situations, especially when the customers (packets in WSNs) are limited by a deadline [12]. With LIFO, the newer packets can be served sooner and older customers will more likely be timed out and dropped.

In addition, LIFO and our multi-queue algorithms improve the fairness performance. It is fairest if the BS can receive equal amount of data from each sensor node. In congested FIFO WSNs, the packets from far-away nodes may time out on the way to the BS because they cannot transmit until the previous packets' transmissions are completed, which may take very long time when congestion happens. In the same scenario, with LIFO, they do not have to wait for the previous packets and have a better chance to arrive at the BS. The multi-queue algorithm in a node divides the physical queue into several logical sub-queues. Each sub-queue is distributed to the node's each child node. Then based on the packet arrival rates from its child nodes, each sub-queue is assigned the corresponding weight. Packets in sub-queues with higher weights are given higher priority in serving, and thus fairness is further improved.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 proposes LIFO queueing mechanism in overloaded WSNs and verifies that it improves the delay and fairness performance during congestion. Section 4 proposes a multi-queue algorithm and proves it fairness advantages over single queue. Simulation results are provided in Section 5. Finally, Section 6 concludes the paper.

2. RELATED WORK

Many mature theoretical results of congested queueing systems have been developed. Doshi and Heffes study the overload performance under a variety of buffering disciplines in star-shaped distributed systems [13]. They also analyze the congested performance of various queueing disciplines for an M/M/l queue, and show that LIFO achieves a much shorter delay than FIFO [12]. Movaghar studies two different queueing systems with m servers [14]. They are either based on a single queueing system or on star-shaped systems, and cannot be applied to study congested WSNs. Jackson networks do not work for congested WSNs as WSNs do not follow speJackson networks [15]. In a Jackson network, the utilization of all of the queues is less than 1 whereas in congested WSNs at least one queue's utilization is 1.

To mitigate the congestion in overloaded WSNs, Wan et al. [4] propose CODA (COngestion Detection and Avoidance) to detect and avoid congestion. Later, Bret et al. [3] propose three congestion control techniques that span transport layer to MAC layer of the traditional protocol stack. Specifically, to handle the congestion caused by the many-to-one multi-hop traffic pattern of WSNs, Wan et al. [5] employ a small number of wireless and multi-radio virtual sinks to scale the heavy traffic loads. Schmitt et al. [16] propose a simple strategy that always gives strict priority to flows that have had a longer path from the BS, and achieve a good balance of response times across the sensor field. While these methods alleviate the congestion, they disregard the real-time requirement on sensor data and ignore the performance degradation during congestion. In addition, no method in respect of queueing-mechanisms is considered for the congestion problems in WSNs. Hence, we design a queueing method that helps improve the performance degradation during congestion.

Although there exists no study on queueing systems in congested WSNs, Mathur and Apte propose an analytical model to study the performance of LIFO and FIFO in overloaded Web services with impatient customers [17]. They show that both throughput and response time of LIFO are better than those of FIFO. However, the customers may feel unfair compared with using FIFO in gaining Web services. This kind of unfairness is not the concern of WSNs in which each customer is a sensor node. As long as it can have its collected data received by the BS within the required deadline, it does not care about LIFO or FIFO. Furthermore, the communication topology in WSNs is different from that in Web services. Therefore, we propose to use LIFO and prove that LIFO suits overloaded WSNs very well.

Since LIFO achieves shorter delay than FIFO in simple congested queueing systems and overloaded Web services, and the above congestion control mechanisms cannot improve the congestion performance, we are inspired to use LIFO instead of FIFO to improve both the delay and fairness performance in congested WSNs.

3. LIFO QUEUEING MECHANISM FOR CONGESTED WSNS

Without loss of generality, we take WSNs that perform monitoring applications as an example. In WSNs, packets coming from the upper layers will be queued at the network layer before being served by the MAC layer. We focus on the first two of the three main factors that affect the queueing: the data generation interval at each node in the application layer (T_i), the deadline of data freshness (T_d) required in WSN applications, and the queue size (Q_s). In resource-limited sensor node, we assume Q_s is a constant value (E.g., 30) for two reasons: the queue size is limited and not liable to be changed or adjusted; the results in [12] indicate that Q_s and T_d exert similar influence in packet drop, so do our experiments. In this section, we analyze LIFO queueing mechanism for congested WSNs and show its improvement in delay and fairness.

Delay analysis

According to the well-established and frequently-used LIFO and FIFO comparison theory in a congested M/M/1 queue

[12], LIFO achieves much better delay performance than that of FIFO, whereas the throughput performance differs little because the congested networks are saturated.

For node i, let L_i be the average delay of all the packets served in the LIFO queue, and F_i be the average delay if node i employs FIFO queue instead. Then $L_i \leq F_i$.

The average delay of interconnected queues in LIFO networks and FIFO networks is

$$T_L = \sum_i L_i \tag{1}$$

$$T_F = \sum_i F_i \tag{2}$$

respectively. Then it follows that $T_L \leq T_F$.

Therefore, LIFO achieves much better delay performance than FIFO in congested WSNs.

Fairness definition

In the traditional notion, LIFO is not fair to all the customers that enter the queue [17]. Later comers could get served much earlier while earlier comers might time out and get refused. However, the fairness in WSNs is considered not in aspect of each customer but in aspect of the application goal, i.e., the BS should receive the same amount of real-time data from each deployed node in time. As long as the BS can receive the expected data to monitor the whole area in time, it does not care which discipline each node is using.

In a time interval T, suppose the BS receives x_i packets from the i-th node, $i \in \{1, \cdots, N\}$, and N is the number of nodes in the WSN. Then the *fairness index* [18] of the WSN is defined as:

$$f(x) = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N \sum_{i=1}^{N} {x_i}^2}, \quad x_i \geq 0. \tag{3}$$

If x_i is equal to each other, then the fairness index is 1 (fairest). Otherwise, the larger the difference among them, the smaller is the fairness index. When $\sum_{i=1}^{N} x_i$ is a fixed number, it is very unfair if some nodes have no packets received by the BS. This is the case we should avoid, but it is very likely to happen in congested FIFO WSNs.

Fairness analysis

In congested WSNs, FIFO is unfair to nodes that are far away from the BS. Packets generated by these nodes may never arrive at the BS at all. LIFO is fairer because they are able to arrive at the BS with a higher probability (not 0). Next we prove this.

Let R be a multi-hop path to the BS. In the path R node m is m hops away from the BS, $\forall m \in \{1, \cdots, H\}$, and H is the maximum hops away from the BS. We assume that each node follows M/M/1 mechanism, i.e. the packets arrivals are a Poisson process (the mean arrival interval is T_i), the service times are exponentially distributed (the mean service time is T_s), there is one server, the length of queue in which arriving packets wait before being served is infinite, and the population of packets available to join the system is infinite.

The queue utilization of node m is $\rho_m = T_{sm}/T_{im}$. In congested WSNs, $\rho_m \geq 1, \forall m \in \{1, \cdots, H\}$.

Node H is farthest from the BS, and we calculate the congestion condition that its first packet P_{H1} can never arrive at the BS if we employ FIFO.

Suppose node H's first packets takes time t_H (It could be T_{sH}) to be served, then before it arrives at node $(H-1)$ to be forwarded (served), node $(H-1)$ has already generated $P_{(H-1)} = t_H/T_{i(H-1)}$ packets.

After packet P_{H1} arrives at node $(H-1)$, it takes node $(H-1)$ extra time $t_{H-1} = T_{s(H-1)}(P_{(H-1)}+1) - t_H$ before packet P_{H1} arrives at node $(H-2)$.

By this time, node $(H-2)$ has generated $P_{(H-2)} = (t_H + t_{H-1})/T_{i(H-2)}$ packets, and it takes node $(H-2)$ extra time $t_{H-2} = T_{s(H-2)}(P_{(H-2)} + P_{(H-1)} + 1) - t_H - t_{H-1}$ before packet P_{H1} arrives at node $(H-3)$.

Finally, node 1 has generated $P_1 = (\sum_{m=2}^{H} t_m)/T_{i1}$ packets by the time packet P_{H1} arrives at it, and it takes node 1 extra time

$$t_1 = T_{s1}(\sum_{m=1}^{H-1} P_m + 1) - \sum_{m=2}^{H} t_m. \tag{4}$$

In total, packet P_{H1} has to wait time

$$T = \sum_{m=1}^{H} t_m \tag{5}$$

until it arrives at the BS. When ρ_m is very high and T_d is small, $T > T_d$ holds, i.e., packet P_{H1} has stayed in the network for so long a period that it fails the deadline and is dropped before arrival at the BS. As a result, the first packet generated by the farthest node cannot arrive at the BS, let alone the packets afterwards because the network is getting more congested.

On the other hand, if we employ LIFO, no matter what ρ_m and T_d is, unlike FIFO situation, the probability that the packet P_{H1} arrives at the BS is not 0. Since each time packet P_{H1} arrives at the forwarding node m, it is enqueued as the first one in the queue and gains the highest priority. Although it has to wait a certain period of time ($T_{sm}/2$ in average) for node m to finish sending the current packet, the probability that it will not be preempted by the new arrivals in this period is about

$$2T_{im}/T_{sm}. \tag{6}$$

Finally, the probability that it will always be served at each forwarding node without being preempted and thus arrive at the BS is

$$\prod_{m=1}^{H-1} \{2T_{im}/T_{sm}\}. \tag{7}$$

It is not 0, and thus LIFO is fairer than FIFO for the far-away nodes.

Fairness case study

Specifically, we take a simple WSN shown in Figure. 1 as an example. Considering the case that the WSN employs FIFO,

Figure 1. Three-node topology of a WSN.

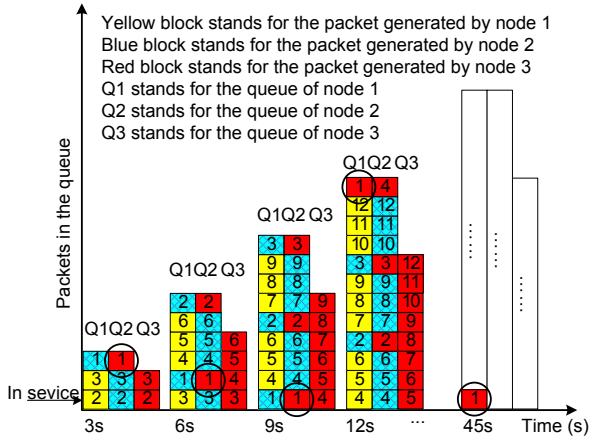

Figure 2. Queue status in the overloaded FIFO WSN with topology shown in Figure. 1.

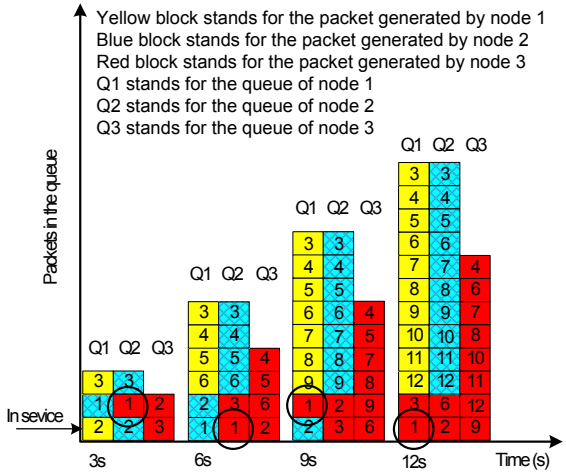

Figure 3. Queue status in the overloaded LIFO WSN with topology shown in Figure. 1.

node 3 that is farthest from the BS may never have any data received by the BS. In the example shown in Figure. 2, T_s = 3s, T_i = 1s. T_d = 30s. The 3-node WSN is overloaded (because $T_s/T_i > 1$) and will congest in a short period. Suppose each parent node finishes transmitting its packet a little moment earlier than receiving the forwarding packets from its child. Then at 3s, node 1 has already generated 3 packets, and its first packet has got served and arrived at the BS, and node 2's first packet is enqueued at the tail of node 1's queue. As time goes by, all the queues are enqueued with more and more packets. Since each forwarded packet is queued at the tail of its parent's queue, packets from far-away nodes will experience more and more packets ahead of them each time they get nearer to the BS. This process is also depicted in Figure. 2. Observing the circled packet, i.e., the first packet generated by node 3, it takes 3s for it to arrive at the queue of node 2, but it takes another 9s for it to arrive at node 1's queue. By the time it arrives at node 1 at time 12s, it is enqueued at the tail of the queue as the 12th packet. Then it has to wait another 36s (i.e. 3×12) to be the served by node 1. However, since the deadline is 30s, it cannot arrive at the BS in time. As for node 3's following packets, they will also fail to arrive at the BS because the network will become more congested by the time of their generation and forwarding.

In contrast, as proved, this unfairness will not happen in congested LIFO WSNs as each time the packet arrives at the forwarding node, it is enqueued at the head of the queue. Figure. 3 shows the queueing process of the previous WSN in Figure. 1 under LIFO. At time 12s the first packet (the circled packet) generated by node 3 arrives at node 1's queue as the first packet, and thus will arrive at the BS at time 15s within the deadline 30s.

Let us consider a more random service scenario for the LIFO WSN in Figure. 1 in which the service time is not a fixed value. In this case if one of node 3's packet get immediately served from node 3 and enters the queue of node 2, it will not necessarily get immediate service from node 2. Although this packet is enqueued to the head of the queue by the time it enters, it might be preempted by other packets' arrival during its waiting for the service. Even if it is immediate served and forwarded to node 1, likewise, it will not necessarily get immediate service from node 1. It is a random process that whether a packet will get immediate service or non-immediate service at each node. Therefore, LIFO in queueing networks does not favor certain nodes, but balances the unfairness of the single-queue LIFO during multi-hop services.

4. Multi-queue Queueing Mechanism for Congested WSNs

Although untraditionally employing the LIFO in WSNs improves the congestion performance greatly, our multiple sub-queue mechanism can further improve the fairness and enhance the the application reliability. Since the real sensor node only has a single queue, the multiple sub-queues are maintained logically and thus is easy to implement without extra hardware requirement.

Fairness analysis

The communication topology of WSNs works like a tree as shown in Figure. 4. In a WSN with 1 BS and N sensor nodes, node n_i has K_i children $C_1, C_2, \cdots, C_{K_i}$, and $i \in \{1, 2, \cdots, N\}$. Rooted at each child node C_z is a subtree also called C_z. The size of the subtree is W_z, and $z \in \{1, 2, \cdots, K_i\}$. As a special case, we include sensor node n_i as its own child corresponding to subtree n_i with size 1. Next, we prove that the network is unfair with single queue mechanism.

Let the reception capacity of the BS be V, and the BS has m_0 1-hop children. Node n_f is f-hop away from the BS. Node n_i is on the path from node n_f to the BS, and it is i-hop away from the BS, $\forall i \in \{, 1, 2, \cdots, f\}$. m_i is the number of 1-hop children of n_i, and $i \in \{1, 2, \cdots, f\}$. Then node n_1 shares V/m_0 reception capacity with the BS's other 1-hop children. Since n_1 has m_1 children, node n_2 is allocated $\frac{V}{m_0 * m_1}$ reception capacity. In this way, we get A_i, the reception capacity allocation of each node n_i, as follows:

$$A_i = \frac{V}{m_0 * m_1 * m_2 * \cdots * m_{i-1}}, \forall i \in \{1, 2, \cdots, f\} \quad (8)$$

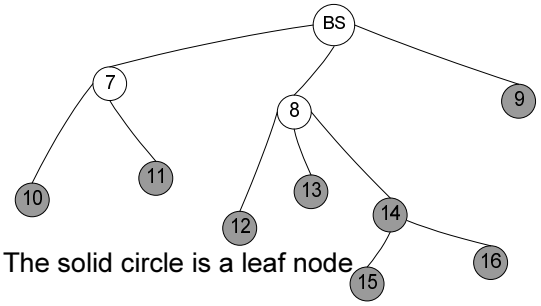

Figure 4. Tree topology of WSNs.

According to this equation, we find that the farther the node is away from the BS, the lower the probability that its packets are received by the BS. With our multi-queue mechanism, each node is allocated V/N reception capacity and is therefore fairer. Next, we show the algorithm that achieves the very fairness.

Multi-queue-LIFO algorithm

To explain our algorithm, we first take Figure. 4 an example. In this figure, the BS has 3 subtrees rooted at its 3 child nodes, the first subtree is rooted at node 7 with size 3, the second is rooted at node 8 with size 6, and the third is rooted at node 9 with size 1. Larger subtree size means more data to be sent to the parent node. When the network is not in congestion, it is ideal that the child node with larger subtree will have more data received by the BS. However, in congested situation, since the MAC layer allows fair competition for the channel, the BS will receive almost similar amount of packets from its three children. As the reception capacity of the BS is V, then each subtree is allocated $V/3$. Thus, each node in subtree 9 is allocated $V/3$, each node in subtree 7 is allocated $V/9$. In the subtree 8, node 8, 12, and 13 are allocated $V/12$, and node 14, 15, and 16 are allocated $V/36$. This is very unfair to the nodes in subtree 8.

Our queueing mechanism cannot change the underlying unfairness caused by unbalanced deployment and routing. For example, node 8 has a larger subtree than node 7 and 9. However, our multi-queue mechanism can still improve the fairness performance by giving higher capacity to packets from larger subtrees, the probability of forwarding packets from children is in proportion with the sizes of the subtrees rooted at the children. In this way, data generated by each node will obtain an equal chance of transmission at the forwarding nodes. For example, the BS allocates its reception capacity V to its children according to their subtree sizes. As a result, subtree 7 is allocated $3V/10$, subtree 8 is allocated $6V/10$, and subtree 9 is allocated $V/10$. Finally, each node in all the subtrees is allocated $V/10$ capacity. It is much fairer than just relying on LIFO or FIFO disciplines.

Actually, sensor node is equipped with limited resources, and it is impractical to implement real multiple queues at each node for its child nodes. Therefore, we only implement one real queue at each node, but maintain a logical multi-queue for each subtree. In the real queue of node n_i with K_i children, we allocate a sub-queue $SubQ_z$ for packets sent from each subtree C_z, and $z \in \{0, 1, 2, \cdots, K_i\}$. The subtree size is W_z. Each time node n_i is going to send a packet, it will first select a sub-queue to dequeue the packet rather than dequeue directly from the real queue. The probability P_z that node n_i dequeues and forwards packet in sub-queue $SubQ_z$ is in proportion with the corresponding tree size W_z, i.e., $P_z = W_z / \sum_0^{K_i} W_j$. By controlling the number of packets to be forwarded for each sub-queue, the fairness is improved. The multi-queue-LIFO algorithm at node n_i is shown in Algorithm 1.

Algorithm 1 Multi-queue-LIFO algorithm at node n_i.

```
calculate W_z, ∀z ∈ {0, 1, 2, ··· , K_i}.
while true. do
if node n_i finishes the current forwarding. then
remove timeout packets from the queue.
if node n_i's queue is not empty. then
dequeue a packet from the head of SubQ_z with probability P_z, ∀z ∈ {0, 1, 2, ··· , K_i}.
end if
end if
if a packet arrives at node n_i. then
remove timeout packets from the queue.
if the queue is full. then
randomly drop the last packet of a sub-queue.
end if
enqueue the packet to the head of the queue and updates the corresponding sub-queue.
end if
end while
```

5. PERFORMANCE EVALUATION

We use the popular freeware network simulator ns2 as the simulation environment [9]. We conduct the performance evaluation in congested networks of different sizes: from 121-node networks to 400-node networks. Their results are similar, and for the sake of conciseness, we only demonstrate the results of 196-node networks with two different topologies. These network scenarios are generated by the setdest tool of ns2 [9]. Nodes are randomly distributed and connected. In the following, the queue size is fixed as 30. Under the impact of deadline T_d and data generation interval T_i, we compare the delay and fairness performance of four mechanisms: single-queue-FIFO, single-queue-LIFO, multi-queue-FIFO, and multi-queue-LIFO. The former two implement a single queue, and the latter two implement our proposed multiple sub-queues. We do not compare their throughput performances which are similar in saturated networks (both the well-established theory and our experiments confirm this).

Delay

Delay is calculated from the time a data is generated to the time it is received by the BS. We calculated the average delay of all the packets received by the BS in two 196-node network topologies. Figure. 5 and 7 show that when T_d is increasing, the delay of all the four mechanisms is increasing, and the delay of FIFO mechanisms increases much faster. It is obvious because the data are allowed to stay in the network for a longer time and FIFO performs much poorer than LIFO in overloaded situations. In Figure. 6 and Figure. 8, when the T_i rises, the delay of FIFO mechanisms tend to converge with the delay of LIFO mechanisms. It is reasonable because the traffic load becomes lighter when T_i increases. If the T_i increases to a large enough value when no congestion happens, then the throughput is 1, and the delay performance of LIFO and FIFO should be the same. When $T_i = 300s$, the throughput of Topology I and II is very close to 1: about 0.7 and 0.91 respectively according to my experiment results. According to Figure. 10 and Figure. 12, the fairness index

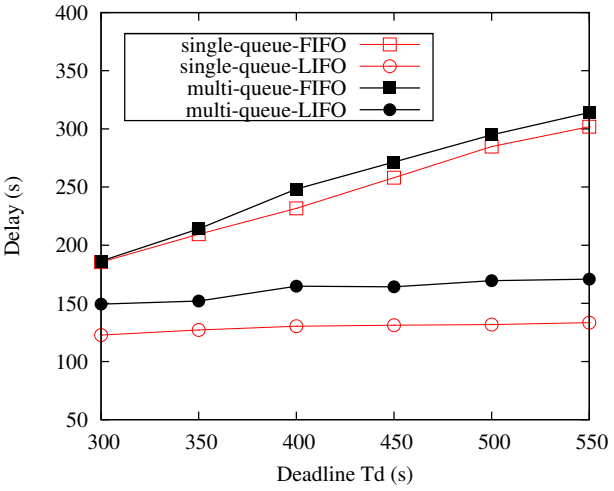

Figure 5. Delay comparison with $T_i = 200s$ in 196-node network topology I.

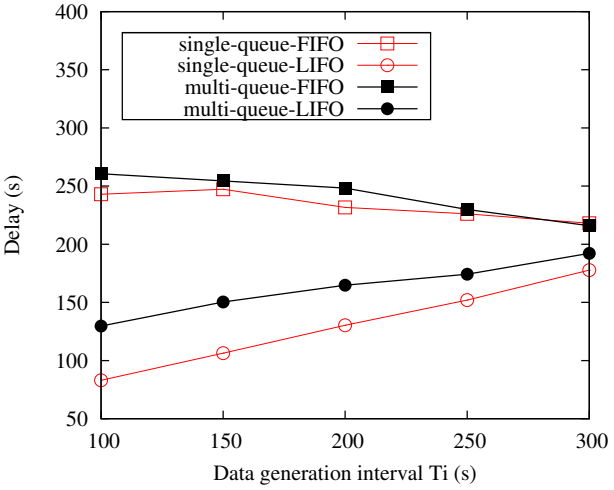

Figure 6. Delay comparison with $T_d = 400s$ in 196-node network topology I.

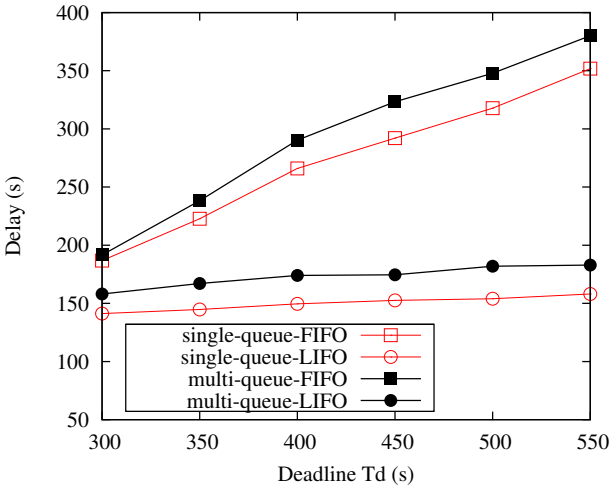

Figure 7. Delay comparison with $T_i = 200s$ in 196-node network topology II.

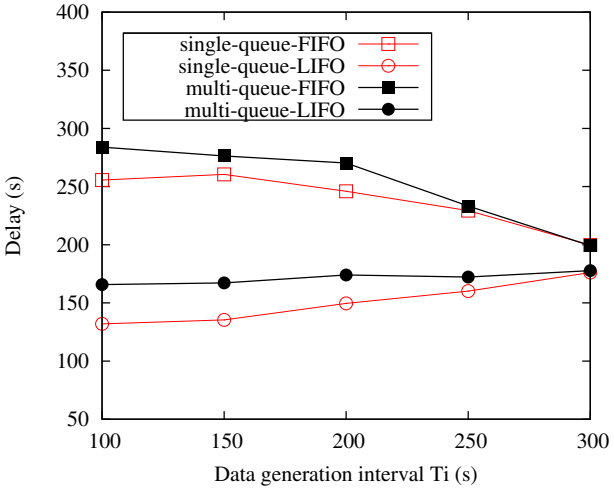

Figure 8. Delay comparison with $T_d = 400s$ in 196-node network topology II.

increases with T_i, which means that more packets from far-away nodes are received, thus the LIFO delay increases. In all these figures, we notice that the delay of single-queue-FIFO and multi-queue-FIFO is much higher than that of single-queue-LIFO and multi-queue-LIFO respectively. In addition, the delay of multi-queue mechanisms is a little higher than that of single queue mechanisms because multi-queue mechanism is fairer and could transmit more data for far-away nodes. This slight delay increase is traded off by multi-queue's fairness improvement.

Fairness

In Figure. 9, as T_d increases, the fairness index defined in Equation 3 of all the four mechanisms increases because when the data can stay in the network for a longer time. Because LIFO is fairer than FIFO, the fairness index of LIFO mechanisms is about 30% higher than that of FIFO mechanisms when $T_d = 300$, and it about 20% higher when $T_d = 550$. In addition, the increase rate of LIFO mechanisms is lower than that of FIFO mechanisms, which means that LIFO is more robust than FIFO when the deadline changes. Furthermore, the fairness index of multi-queue-FIFO is about 10% higher than that of single-queue-FIFO while the fairness index of multi-queue-LIFO is about 6% higher than that of single-queue-LIFO. This demonstrates that the multi-queue mechanism also improves the fairness performance. When there is no congestion, the fairness index is 1. In Figure. 10, the traffic load becomes lighter with T_i increasing, and thus the fairness index of all the mechanisms improves to approach to 1. Figure. 11 and Figure. 12 show the fairness index of another topology, they also demonstrate that LIFO and multiple sub-queues lead to better fairness. The difference is that the fairness index of the second topology is higher than the first one because the second topology is more balanced and performs better in respect of throughput according to the topology and throughput comparison done in the experiments.

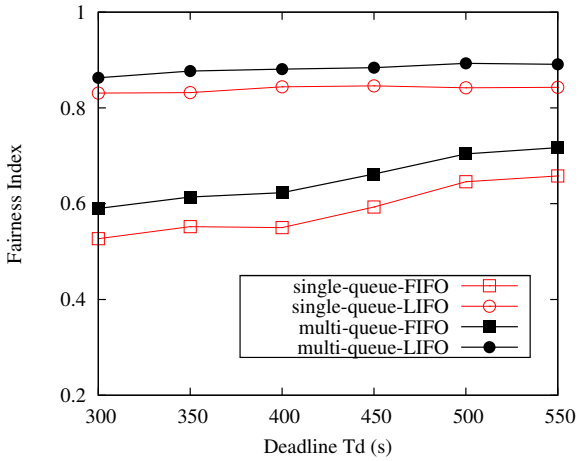

Figure 9. Fairness comparison with $T_i = 200s$ in 196-node network topology I.

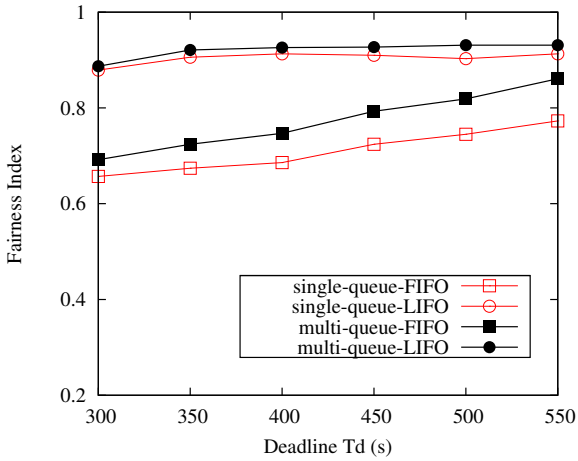

Figure 11. Fairness comparison with $T_i = 200s$ in 196-node network topology II.

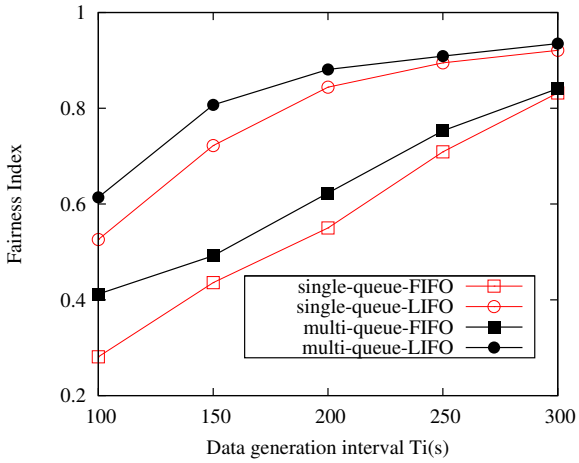

Figure 10. Fairness comparison with $T_d = 400s$ in 196-node network topology I.

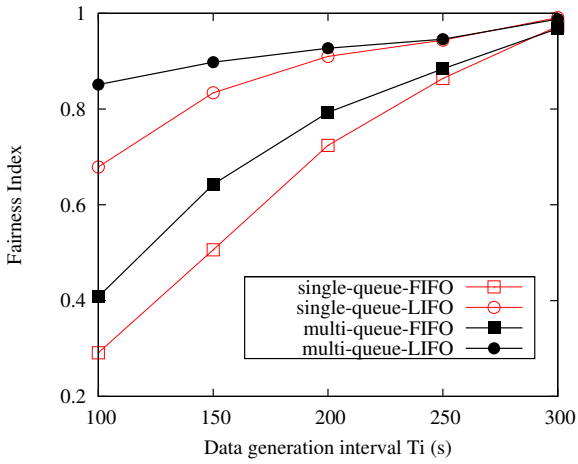

Figure 12. Fairness comparison with $T_d = 400s$ in 196-node network topology II.

6. Conclusion

In this paper, out of the usual FIFO intuition, we improve the delay and fairness performance of congested WSNs by the multi-queue-LIFO mechanism. First we prove that LIFO provides better delay and fairness performance than traditional employed FIFO in congested WSNs. Then to further improve fairness, we implement a multi-queue algorithm by dividing the real queue of a node into multiple sub-queues and adjusting the serving probability for packets from different sub-queues. We compare our multi-queue-LIFO mechanism with the other three mechanisms. The results confirm that multi-queue-LIFO dramatically improves delay and fairness performance while maintaining similar throughput.

In the future, we are interested in studying the queueing mechanisms in which the deadline of the data is not all the same. We also plan to implement the multi-queue-LIFO mechanism in real WSNs because it requires no extra hardware adjustment or application change.

Acknowledgments

The work described in this paper was significantly supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 415311) and sponsored in part by the National Basic Research Program of China (973) under Grant No. 2011CB302600.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on wireless sensor networks," *IEEE Communications Magazine*, vol. 40, pp. 102–114, 2002.

[2] W. Li and J. Han, "Dynamic wireless sensor network parameters optimization adapting different node mobility," in *Proc. of the IEEE Aerospace Conference*, 2010.

[3] H. Bret, J. Kyle, and B. Hari, "Mitigating congestion in wireless sensor networks," in *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004, pp. 134–147.

[4] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion detection and avoidance in sensor networks," in *Proc. of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003, pp. 266–279.

[5] C. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft, "Overload traffic management using multi-radio virtual sinks in sensor networks," in *Proc. of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005, pp. 116–129.

[6] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*. Springer Verlag, 2004, pp. 115–148.

[7] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proc. of the 29th Annual IEEE International Conference on Local Computer Networks (LCN)*, 2004, pp. 455–462.

[8] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire tinyos applications," in *Proc. of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003, pp. 266–279.

[9] Network simulator NS-2. http://www.isi.edu/nsnam/ns/.

[10] V. G. Cerf and R. E. Kahn, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, vol. 22, pp. 637–648, 1974.

[11] J. Heo, B. Gu, and S. I. Eo, "Energy efficient program updating for sensor nodes with flash memory," in *Proc. of the 2010 ACM Symposium on Applied Computing*, 2010.

[12] B. T. Doshi and H. Heffes, "Overload performance of several processor queueing disciplines for the M/M/1 queue," *IEEE Transactions on Communications*, vol. 34, no. 6, June 1986.

[13] ——, "Comparison of control schemes for a class of distributed systems," in *Proc. of the 21st IEEE Conference on Decision and Control*, Dec 1982, pp. 846–853.

[14] A. Movaghar, "On queueing with customer impatience until the beginning of service," *Queueing Systems Theory and Applications*, vol. 29, pp. 337–350, 1998.

[15] B. R. Haverkort, *Performance of Computer Communication Systems*. J. Wiley, 1998.

[16] J. B. Schmitt, F. A. Zdarsky, and L. Thiele, "A comprehensive worst-case calculus for wireless sensor networks with in-network processing," in *Proc. of the IEEE Real-Time Systems Symposium (RTSS)*, Dec 2007.

[17] V. Mathur and V. Apte, "An overhead and resource contention aware analytical model for overloaded web servers," *Journal of Systems and Software*, vol. 82, no. 1, pp. 39–55, 2009.

[18] R. K. Jain, D.-M. W.Chiu, and W. R.Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," Washington University, Tech. Rep., 1984.

BIOGRAPHY

Junjie Xiong is currently a PhD student in the Computer Science and Engineering Department at the Chinese University of Hong Kong. She received the B.Sc. degree (2005) in electronic and information engineering from Wuhan University, Wuhan, China and the M.S degree (2008) in communication and information system from the University of Science and Technology of China, Hefei, China. Her research interests are in wireless networks and wireless sensor networks, especially their reliability and performance.

Michael R. Lyu received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, China, in 1981, the M.S. degree in computer engineering from University of California, Santa Barbara, in 1985, and the Ph.D. degree in computer science from University of California, Los Angeles, in 1988. He worked at the Jet Propulsion Laboratory, Bellcore, and Bell Labs; and taught at the University of Iowa.
He is currently a Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, wireless communication networks, Web technologies, mobile networks, digital video library, multimedia processing, and machine learning. He has published over 350 refereed journal and conference papers in these areas. He has participated in more than 30 industrial projects in these areas, and helped to develop many commercial systems and software tools. He has been frequently invited as a keynote or tutorial speaker to conferences and workshops in U.S., Europe, and Asia.
He initiated the first International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the Program Chair for ISSRE'96, Program co-Chair for WWW10, General Chair for ISSRE'2001, General co-Chair for PRDC'2005, and has served in program committees for many conferences. He is the editor for two book volumes: Software Fault Tolerance, published by Wiley in 1995 and the Handbook of Software Reliability Engineering, published by IEEE and McGraw-Hill in 1996. These books have received an overwhelming response from both the academia and the industry. He was an Associate Editor of IEEE Transactions on Reliability, IEEE Transactions on Knowledge and Data Engineering, and Journal of Information Science and Engineering. He is currently on the editorial board of Wiley Software Testing, Verification and Reliability Journal. He is an IEEE Fellow and an AAAS Fellow. He was also named Croucher Senior Research Fellow in 2008 and IEEE Reliability Society Engineer of the Year in 2010.

Kam-Wing Ng has been with the Chinese University of Hong Kong since 1978. In between, he had taught at the University of Wollongong (1985-86) and the University of New South Wales (1989-91) in Australia. He was the organizing chairman of the first international computer science conference held in Hong Kong in 1988 (ICSC'88). He was the Head of the Graduate Division for 97-98, 02-05, and was the Director of the MSc in Computer Science programme for 96-98, 00-06. He had been the Chairman of the Department of Computer Science and Engineering from 1 August 97 until 31 July 99. His research interests include computer networks and grid computing.