Pipeline Scheduling Based on Constructive Interference in Strip Wireless Sensor Networks

Xiangmao Chang[1, 2, *], Xiaoxiang Xu[1] and Deliang Yang[3]

Abstract: Strip Wireless Sensor Networks (SWSNs) have drawn much attention in many applications such as monitoring rivers, highways and coal mines. Packet delivery in SWSN usually requires a large number of multi-hop transmissions which leads to long transmission latency in low-duty-cycle SWSNs. Several pipeline scheduling schemes have been proposed to reduce latency. However, when communication links are unreliable, pipeline scheduling is prone to failure. In this paper, we propose a pipeline scheduling transmission protocol based on constructive interference. The protocol first divides the whole network into multiple partitions and uses a pipelined mechanism to allocate active time slots for each partition. The nodes in the same partition wake up at the same time for concurrent transmission. Multiple identical signals interfere constructively at the receiver node, which enhances received signal strength and improves link quality. Simulations show that the proposed scheme can significantly reduce the transmission latency while maintaining low energy consumption compared with other schemes.

Keywords: Strip wireless sensor network, constructive interference, pipeline scheduling.

1 Introduction

Strip Wireless Sensor Networks (SWSNs) [Chen and Wang (2008); Jawhar, Mohamed and Agrawal (2011); Stoianov, Nachman, Madden et al. (2007)] are usually deployed in a narrow but long strip area for real-time monitoring and data collection. For example, gas monitoring in coal mines needs to deploy sensors on tunnels which are only several meters wide but several kilometers long [Guo, Jiang and Zhang (2012)]. Other applications of SWSN include monitoring highways, pipelines, coastal lines and rivers. The special topology of SWSN makes them more restrictive than ordinary WSNs in terms of node communication and data collection.

Sensor nodes of SWSN generally operate at a low-duty-cycle to save energy. Sensor nodes are dormant for most of the period and active for a short period in each duty cycle. In this case, when a node is going to deliver a packet, it must wait for some period until

[1] Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China.

[2] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China.

[3] Michigan State University, Lansing, USA.

[*] Corresponding Author: Xiangmao Chang. Email: xiangmaoch@nuaa.edu.cn.

Received: 04 August 2019; Accepted: 11 September 2019.

its receiver node is active, which results in transmission latency. As SWSN usually requires a large number of hops to deliver a packet, the transmission latency of each hop can lead to a very serious transmission latency of the whole network. For example, a typical transmission latency of 0.5 s within each hop can lead to a transmission latency of several seconds in SWSNs [Guo and Meratnia (2015)], which is unacceptable in many delay-sensitive applications.

Lots of scheduling schemes have been proposed to reduce the transmission latency caused by low duty cycle in SWSNs [Cao, Abdelzaher, He et al. (2005); Keshavarzian, Lee and Venkatraman (2006); Cao, Guo and He (2012); Guo and Meratnia (2015)]. Pipeline scheduling schemes [Cao, Abdelzaher, He et al. (2005); Keshavarzian, Lee and Venkatraman (2006)] allocate the activation time for each sensor node of the transmission path sequentially, such that each sender node can send a packet to its receiver node directly without any latency. Therefore, pipeline scheduling schemes can achieve the minimum transmission latency in low-duty-cycle SWSNs theoretically. However, in real communication environments, wireless links are usually unreliable to achieve pipeline transmission. When the transmission interrupts due to the instability of the link, the sender node needs to wait until the next cycle to re-transmit the data to the receiver node, which results in longer transmission latency.

To deal with the problem of unreliable links in pipeline scheduling schemes, algorithms such as multi-pipeline scheduling [Cao, Guo and He (2012)] and opportunistic pipeline [Guo and Meratnia (2015)] have been proposed. Multi-pipeline scheduling [Cao, Guo and He (2012)] uses two forwarding paths between the source node and the destination node. When the transmission is interrupted on one path, the transmission can be switched to another forwarding path without waiting for the next cycle. However, when both forwarding paths fail, the transmission still needs to wait for a duty cycle period to retransmit the packet. In opportunistic pipeline scheduling [Guo and Meratnia (2015)], the opportunity nodes outside the forwarding path overhear the transmission of the forwarding path. When the transmission on the forwarding path is interrupted, the opportunistic node can forward the data to the next hop in a relay manner to maintain the pipelines. All these scheduling schemes are based on single wireless links which are unable to solve the problem of link instability fundamentally.

Constructive interference (CI) is an emerging technique which has been proved very effective for data delivery in WSNs [Ferrari, Zimmerling, Thiele et al. (2011); Doddavenkatappa, Chan and Leong (2013); Du, Liando, Zhang et al. (2016); Wang, Liu, He et al. (2014); Wang, He, Mao et al. (2013); Cheng, Mao, Wang et al. (2015)]. CI originates from the physical layer tolerance for multi-path signals. When multiple senders transmit an identical packet simultaneously, multiple signals can interference constructively at a common receiver, rather than cause mutual interference. Moreover, CI can enhance the received signal strength so that the link quality can be improved.

In this paper, we propose a CI-based Pipeline Scheduling (CPS) protocol for SWSN. We first divide the whole network into multiple partitions such that the concurrent transmission of a packet in a partition can be received in the next partition. All partitions are labeled with transport partition and spare partition one by one. Then we schedule each partition such that the pipeline transmissions between transport partitions can be achieved by concurrent transmission of nodes in each partition. Multiple identical signals interfere constructively at the receiver node, which enhances received signal strength and improves link quality. Meanwhile, nodes in spare partitions that overhear the transmitted packet can forward the packet when the following transport partition fails to receive the packet such that the pipeline transmission can be continued. Simulation results show that CPS can significantly reduce transmission latency while maintaining low energy consumption.

The rest of this paper is organized as follows. Section 2 presents the network models and introduces CI. In Section 3, we first present an overview of the CPS by an example, then we elaborate on each part of the CPS. Section 4 presents simulations and analyses of the simulation results. Section 5 concludes the paper.

2 Network model and preliminaries

In this section, we present the models and assumptions used in this work and introduce the constructive interference which will be used later in this paper.

2.1 Network model and assumptions

Suppose N sensor nodes $\{s_1, s_2, \ldots, s_N\}$ are randomly deployed in a strip area where the width is much smaller than the length. The sensor nodes are connected to be an SWSN. There are a source node $a = s_1$ and a destiny node $b = s_N$ in the two endpoints of the SWSN, respectively.

Sensor nodes work in a low-duty-cycle to save energy. Each sensor node alternates active state and dormant state periodically. The node in the active state can transmit or receive data, while the node in the dormant state turns off its radio and only its time module works. Sensor nodes alternate their state according to their work schedules. Each schedule is periodic with the same duty cycle *T*. The duty cycle *T* is divided into several slot units τ, i.e., $T = L \cdot \tau$ where $L \gg 1$. The duty cycle *T* can be fixed according to the application requirements. A timeslot τ is a period of an integral transmission (including data transmission, ACK and a guard interval).

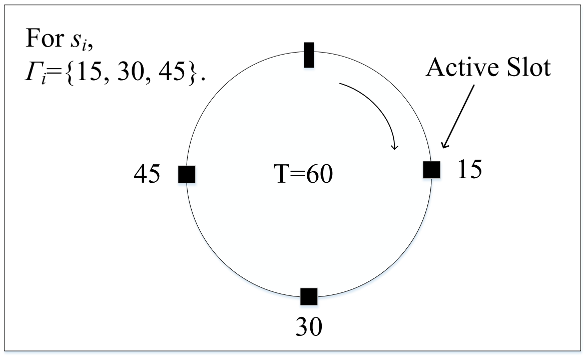

Figure 1: An example of the duty cycle and the active slots of a node

The duty cycle can be represented as a circle, as shown in Fig. 1 [Cao, Guo and He (2012)]. The active time slots of each node are fixed in the schedule. For node s_i, the active schedule Γ_i can be uniquely represented as a set of active slots, $\Gamma_i = \{t_1^i, t_2^i, \ldots, t_m^i\}$, where t_j^i denotes the *j*th active slot of node s_i. For example, in Fig. 1, the duty cycle is *T*=60, a node s_i is allocated a set of active slots: 15, 30, 45, then the active schedule of the node s_i can be uniquely represented as $\Gamma_i = \{15, 30, 45\}$. Each node maintains a local schedule table which records the active time of its neighbors so that the sender can transmit a packet at a proper time slot when the receiver active.

The links of SWSNs are unreliable due to factors such as wireless interferences, noises and channel fading. To simulate the reception probability of the real environment more objective, we use the *SNR-PRR* model [Chang, Huang, Liu et al. (2016)] measured in the real environment to simulate the reception probability of the packet.

2.2 Constructive interference

Constructive interference Ferrari et al. [Ferrari, Zimmerling, Thiele et al. (2011)] originates from the physical layer tolerance for multipath signals: When multiple senders transmit an identical packet simultaneously, concurrent packet transmissions can enhance signal strength and improve the link quality of a common receiver, rather than causing mutual interference.

We use the set *C* indicates a set of nodes of concurrent transmissions. Then, the received signal strength of the receiver node *j* is:

$$SNR_j = \frac{(\sum_{i \in C} \sqrt{r_{ij}})^2}{\sum_{i \in C} \varepsilon}, \tag{1}$$

where r_{ij} represents the signal strength received by the node *j* from node *i*, ε represents the noise power, the numerator represents the total signal strength received by the node *j*, the denominator represents the total noise [Wang, Liu, He et al. (2014)].

3 CI-based pipeline scheduling

In this section, we first present the overview of the proposed CI-based Pipeline Scheduling (CPS) by an example, then we present each part of CPS in detail.

3.1 Overview of CPS

Fig. 2(a) shows an SWSN with a destiny node *d*. Firstly, we divide the whole network into several disjoint partitions, such that, if any sensors of a partition transmit a packet simultaneously, then at least one sensor in the next partition can receive the packet. The destination node *d* lies in a separate partition. In this example, the SWSN is divided into nine partitions, as shown in Fig. 2(b).

All partitions are numbered sequentially from left to right. We refer the partitions with the odd serial number as transport partitions and refer the partitions with the even serial number as spare partitions. Nodes in the same partition have the same schedules. When a partition is transmitting, a set of nodes inside the partition are selected to transmit the packet simultaneously.

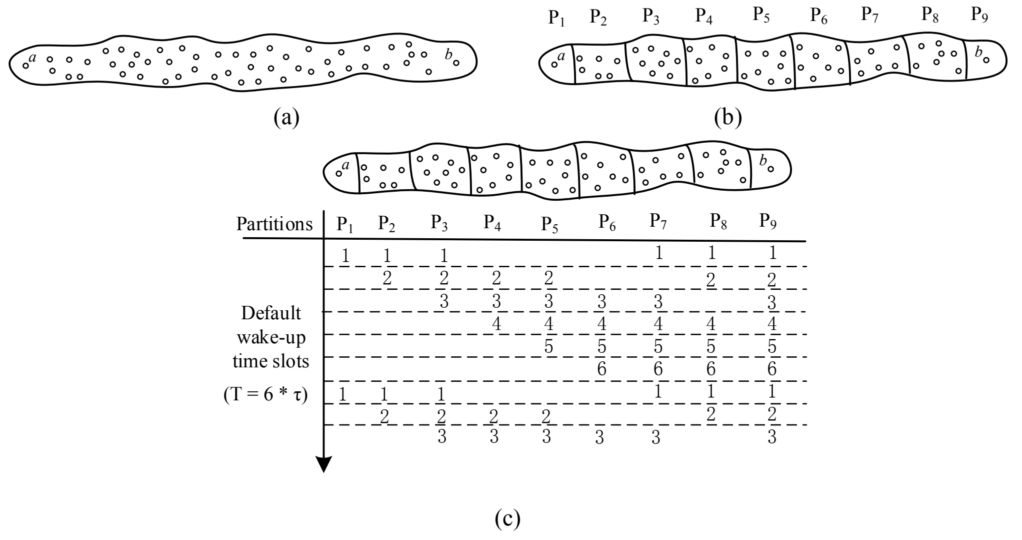

Figure 2: An example of the proposed CPS

The packet is transmitted from the source node to the destination node. The ideal packet transmission process is partition $1 \rightarrow$ partition $3 \rightarrow$ partition $5 \rightarrow$ partition $7 \rightarrow$ partition 9, where the packet is completely transferred in transport partitions, spare partitions are not used. However, due to link instability, transport partition may not be able to successfully forward packet directly to the next transport partition. As an example, we analyze the transmission result when transport partition 3 tries to transmit the packet to transport partition 5, and spare partition 4 listens to the transmission. There are two cases for the transmission:

1) Partitions 4 and 5 successfully receive the packet from partition 3. In this case, partition 5 replies an ACK message to partition 4, and partition 4 replies an ACK message to partition 3. Partitions 3 and 4 go to dormant state immediately after they receive the ACK.
2) Partition 5 fails to receive the packet, while partition 4 successfully overhears the packets transmitted by Partition 3. In this case, the partition 4 replies an ACK message to the partition 3, and the partition 3 goes to the dormant state. Partition 4 is responsible for transmitting the packet to the partition 5 in the next time slot.

When a partition (*transport partition* and *spare partition*) transmits a packet, in order to ensure the next partition is active, a time scheduling for all partitions is necessary. In Fig. 2(c), the duty cycle *T* is divided into 6 time slots. In time slot 1, partition 1 tries to forward the packet to the partition 3. At this slot, partitions 1-3 need to be in the active state. If partition 3 successfully receives the packet, it needs to transmit to partition 5 in the next time slot 2. If partition 3 fails to receive the packet and partition 2 successfully overhears the packet, partition 2 needs to forward the packet to partition 3 in time slot 2 and partition 3 needs to forward the packet to partition 5 in time slot 3. Based on this strategy, we can set different activation times for different partitions. In a duty cycle *T*, partition 1 is in the active state in time slot 1 and in the dormant state in other time slots (time slots 2-6), partition 2 is in the active state in time slots 1, 2 and dormant state for the rest of time slots (3-6), and so on. Partition 9 is active for the entire cycle *T*.

Partitions with larger partition number need more time slots in default active state. However, it does not mean that the partition needs to be active during all default active time slots. For example, in the ideal packet transmission process (i.e., partition $1 \rightarrow$ partition $3 \rightarrow$ partition $5 \rightarrow$ partition $7 \rightarrow$ partition 9), although the default active time slots of partition 7 are {1, 3, 4, 5, 6}, it only needs to receive the packet in time slot 3 and transmit the packet in time slot 4, then goes to dormant state when it receives the ACK message before time slot 5. The default active time slots setting is only for the worst case of transmission.

When a partition is forwarding a packet, it is not necessary for all nodes with the packet to forward simultaneously. We use a selective forwarding mechanism to select a part of nodes for forwarding the packet. Each node sets a forwarding probability, which is dynamically adjusted by the received signal strength, the partition size, and the node density in the network.

3.2 Network partition

The purpose of the network partition is to divide the entire long-strip network into multiple partitions and set different scheduling strategies for each partition. During the network partition process, all nodes maintain active. Each node has a number that indicates which partition it belongs to. We refer to the number as partition number and denote the partition number of node s_i as k_i. Each k_i is initialized as *N*.

The partition process starts from the source node s_1. s_1 sets k_1 to 1, then s_1 broadcasts a packet with a partition number $k_1+1=2$. When a node s_i receives a partition number *k*, s_i reads the signal-to-noise ratio (SNR) of the packet and compares it with a threshold *α*. If the SNR of the packet is greater than *α* and $k_i > k$, s_i sets its partition number $k_i = k$ and broadcast a packet with a partition number k_i+1, else, s_i keeps its partition number unchanged. The detailed partition algorithm is shown in algorithm 1.

When the partition process is finished, each sensor node has a partition number. Let the partition with partition number *k* be P_k. We refer P_k as transport partitions when *k* is odd, and refer P_k as spare partitions when *k* is even. Transport partition is responsible for data transmissions, while spare partition monitors whether the packet in transport partition is successfully transmitted. When transport partition fails, the spare partition which overhears the packet is responsible for transferring the packet to the next transport partition.

Given an SWSN, the number of partitions is determined by the partition threshold *α*. A larger partition threshold results in more partitions. A proper threshold is critical to the transmission latency and the energy consumption of the network. More partitions lead to more transmission hops, while fewer partitions lead to more transmissions of spare partitions, both of which lead to more transmission latency. We experimentally explore the optimal setting of the partition threshold in Section 4.

Only when the network topology changes, the partition algorithm needs to re-run to re-determine the partitions of each node in the network. By using partitions for concurrent transmission, the instability of single-hop transmission can be avoided, which enhances link quality and improves transmission efficiency. At the same time, the node does not need to maintain additional routing information.

Algorithm 1: Partition algorithm

```
 1 Initialize $k_1 = 1$, $k_i = N$ ($i > 1$), the duty cycle is T;
 2 $s_1$ distribute $k_1 + 1$ at timeslot 1;
 3 for each sensor $s_i$ and $i > 1$
 4     timeslot=1;
 5     while timeslot < T do
 6         if receive a partition number then
 7             temp=partition number;
 8             if $k_i$<temp do
 9                 $k_i$=temp;
10                 distribute $k_i + 1$;
11             end if
12         end if
13     timeslot=timeslot+1;
14     end while
15 end for
```

3.3 Pipeline scheduling

In this section, we design a scheduling scheme for each partition such that the pipeline transmission can be achieved between partitions.

When a packet starts to transmit from the source to the destination, it transmits from partition 1 in time slot 1 firstly. Thus, partition 1 should be active in time slot 1 (mod *T*) and dormant in other time slots. For partition P_i, $i \geq 2$, we analyze its scheduling in the following two cases:

(1) P_i is a transport partition. In this case, the fastest way for the packet to reach P_i is the packet can be successfully transmitted to P_i only through transport partitions, i.e., $P_1 \to P_3 \to \ldots \to P_i$. This process needs $i - 1/2$ time slots, thus P_i should be active in time slots $i - 1/2 \pmod T$ to receive the packet and active in time slots $i + 1/2 \pmod T$ to transmit the packet. On the other side, the slowest way for the packet to reach P_i is the packet only can be transmitted through all partitions, i.e., $P_1 \to P_2 \to \ldots \to P_i$.

This process needs $i-1$ time slots, thus P_i should be active in time slots $i-1(\bmod T)$ to receive the packet and active in time slots $i(\bmod T)$ to transmit the packet. In summary, the default active time slots for P_i should be from $i-1/2(\bmod T)$ to $i(\bmod T)$ to deal with all possible cases.

(2) P_i is a spare partition. In this case, the fastest way for the packet to reach P_i is the packet can be successfully transmitted to P_{i-1} only through transport partitions, i.e., $P_1 \to P_3 \to \ldots \to P_{i-1} \to P_i$. This process needs $i/2$ time slots, thus P_i should be active in time slots $i/2(\bmod T)$ to receive the packet and active in time slots $i/2+1(\bmod T)$ to transmit the packet if necessary. On the other side, the slowest way for the packet to reach P_i is the packet only can be transmitted through all partitions, i.e., $P_1 \to P_2 \to \ldots \to P_i$. This process needs $i-1$ time slots, thus P_i should be active in time slots $i-1(\bmod T)$ to receive the packet and active in time slots $i(\bmod T)$ to transmit the packet. In summary, the default active time slots for P_i should be from $i/2(\bmod T)$ to $i(\bmod T)$ to deal with all possible cases.

During the default active time slots, partition P_i gets ready to receive packets firstly. Once any sensors in the partition have received a packet in a time slot j, they send an ACK message concurrently within the time slot j. If any sensors receive an ACK message from any following partitions, it means the packet has been received by at least one following partition, then all sensors of P_i turn to dormant from the time slot $j+1$. If no sensors receive ACK message, sensors of P_i which possess the packet send the packet with a probability concurrently in the time slot $j+1$.

The specific scheduling scheme is shown in algorithm 2.

Algorithm 2: The scheduling algorithm for each sensor node

```
1 Start partition algorithm, let the partition of the sensor node be $P_i$;
2 if $i$ is odd
3     ActiveT=[$i-1/2(\bmod T)$, $i(\bmod T)$];
4 otherwise
5     ActiveT=[$i/2(\bmod T)$, $i(\bmod T)$];
6 timeslot=0; flag=FREE;
7 for each timeslot
8     timeslot=timeslot+1;
9     while timeslot $\in$ ActiveT do
```

```
10        if Receive a packet
11           send an ACK message;
12           flag=READY;
13        end if
14        if Receive an ACK then
15           flag=FREE;
16           Go to dormant and break;
17        end if
18        if flag=READY
19            Send the packet;
20        end if
21     end while
22 end for
```

3.4 Selective forwarding

Redundant nodes participating in concurrent transmission cannot improve the effect of CI [Du, Liando, Zhang et al. (2016)]. To save energy, it is not necessary for all sensor nodes in a partition to forward a packet concurrently. Alternative, appropriate set of nodes needs to be selected and transmitted concurrently.

We assign each node with a forwarding probability to select the appropriate forwarding node-set. In a packet transmission, the nodes far away from the sender nodes have low *SNR* values, but when they forward the packet, more nodes that have not received the packet can receive the packet. On the other side, when the network density is high or the partition size is large, more sensor nodes which are unnecessary will take part in the forwarding process. Thus, the forwarding probability should be inversely proportional with the *SNR* value and the network density, while proportional with the partition threshold. According to the above analyses, we set the forwarding probability for each sensor node as follows:

$$p_i = \frac{K \cdot f(\alpha)}{snr_i \cdot \sqrt{\rho}}, \tag{2}$$

where ρ is the node density in the network, α is the partition threshold, snr_i is the signal reception strength of the node s_i, $f(\cdot)$ is the offset function of the sigmoid function, K is a constant for adjustment.

4 Performance evaluation

In this section, we conduct simulations to evaluate the performance of our proposed CPS. We first test the performance in different settings of the partition threshold. Then we test the performance under different network densities. At last, we compare the performance of CPS with the other two existing approaches.

In SWSNs, the two most important indicators for performance evaluation of end-to-end transmission are transmission latency and energy consumption. The transmission latency is the total time used to finish transmitting a packet, while energy consumption is the total energy consumption of all nodes during the transmission of a packet. Thus, we use transmission latency and energy consumption as the performance indicators in our simulations. The parameters used in our simulations are set as follows: all nodes are randomly placed in an area of $20m \times 1000m$. The transmission power of a sensor node is fixed at 1 dBm and the noise power is -90 dBm. The energy consumption for sending, receiving and channel listening is 17 *mW*, 15 *mW* and 12 *mW* respectively.

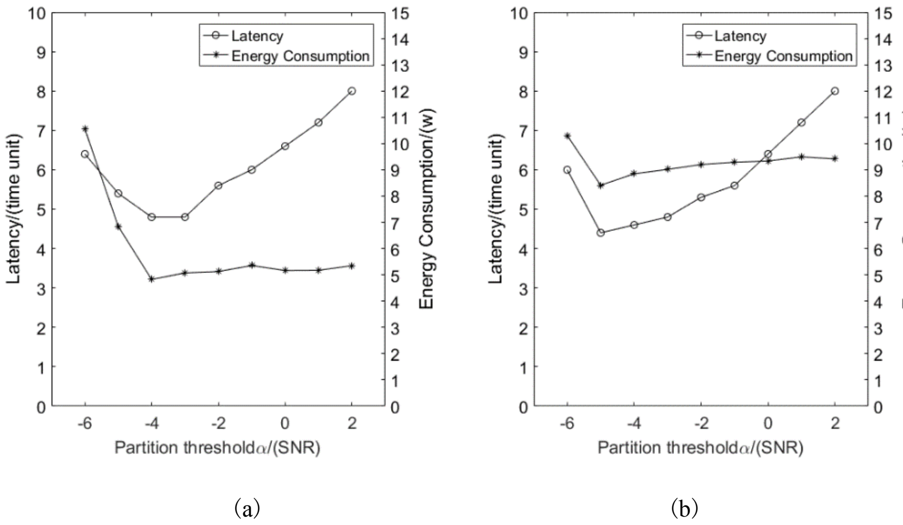

Figure 3: Transmission latency and energy consumption under different partition thresholds

Fig. 3 shows the change of data transmission latency and energy consumption with different partition thresholds. The number of nodes is 400 for Fig. 3(a) and 800 for Fig. 3(b). As shown in Fig. 3, the transmission latency decreases first and then increases with the increase of the partition threshold α, while the energy consumption decreases first and then changes little with the increase of the partition threshold α. We explain this phenomenon as follows. A smaller threshold α leads to a larger partition size and less number of partitions, while a larger threshold α leads to a smaller partition size and more number of partitions. It is difficult for nodes to forward packets from one transport partition to the next transport partition if the partition size is large. Therefore, packets are forwarded through all partitions which increases transmission latency and energy consumption. On the other side, although the packet can be forwarded from one transport partition to the next transport partition if the partition size is small, the transmission latency is increased due to the huge quantity of partitions. The growth of energy consumption is light when the threshold α is large. This is benefited from the scheduling strategy and the selective forwarding strategy. Since the size of each partition is small when α is large, the packet can transmit through most of the transport partitions, which leads to a small period of active time for each partition. Meanwhile, the selective forwarding strategy enables the number of nodes which participant in the forwarding process changes little with the increase of the threshold α .

When the partition threshold is gradually increased to -5 in Fig. 3(a) and -4 in Fig. 3(b), the partition size is rationalized and the transmission latency is minimized. When the partition threshold is too large, although the partition size is small and the packet can be forwarded from one transport partition to the next transport partition, the transmission latency is increased due to the huge quantity of partitions.

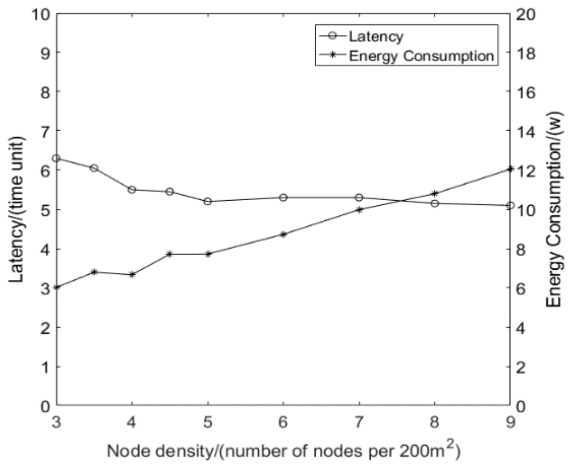

Figure 4: Impact of node density on transmission latency and energy consumption

From Fig. 3, we can also see that, although the network density setting is different between Figs. 3(a) and 3(b), they have the trends of the transmission latency and the energy consumption with the increase of the threshold α . Thus, the choice of the optimal partition threshold α is not related to the network density. The energy consumption in Fig. 3(b) is larger than Fig. 3(a), the reason is that, since the network density in Fig. 3(b) is larger than that in Fig. 3(a), the number of nodes which participant in each forwarding process in Fig. 3(b) is larger than that in Fig. 3(a). In the following simulations, we set the partition threshold α to -5.

Fig. 4 shows the impact of node density on transmission latency and energy consumption. As the node density increases, the transmission latency decreases first and then changes little. The reason is that, a fixed threshold α leads to fixed partitions no matter how node density changes, low node density may result in unsuccessful of transmissions between transport partitions which leads to transmission latency, while high node density may result in successful of transmissions between most of the transport partitions which leads to little changes of the transmission latency. The energy consumption increases with the increase of the node density. This is because high node density leads to more nodes participate in the forwarding process, which consumes more energy.

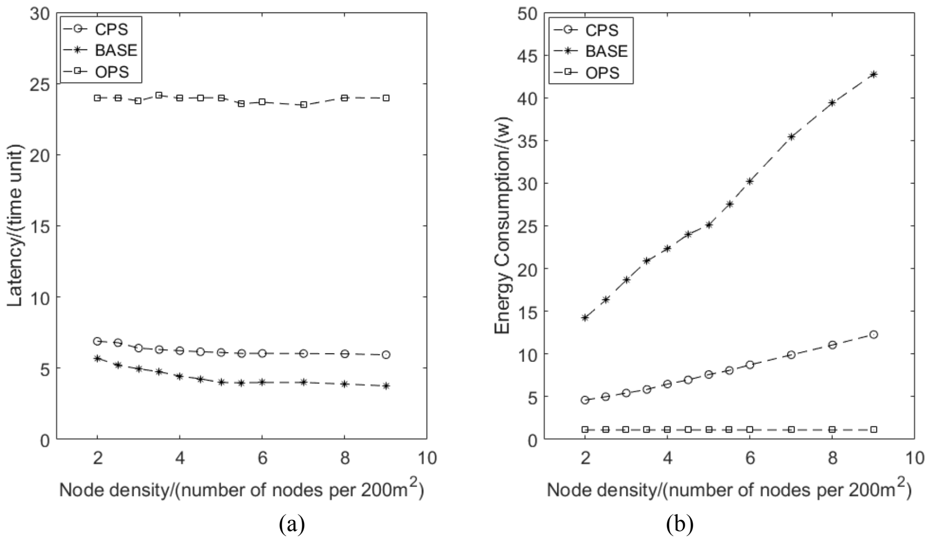

Figure 5: Comparison of transmission latency and energy consumption of three algorithms

We compare CPS with two existing algorithms. One is the benchmark algorithm Baseline [Ferrari, Zimmerling, Thiele et al. (2011)] (BASE). BASE requires that all nodes which receive a packet forward concurrently at a certain probability $P = \frac{K}{snr \cdot \sqrt{\rho}}$, where ρ is the node density of the network, snr is the signal reception strength of the node and K is a constant for adjustment. Another algorithm is the Opportunity Pipeline Scheduling Algorithm [Guo and Meratnia (2015)] (OPS). The main idea of OPS is that when a packet transmits along a forwarding path and the transmission is interrupted, the nodes which are outside the data forwarding path and have received the data are responsible to forward the packet instead of retransmitting in the next cycle. Both BASE and OPS are designed for wireless sensor networks and aim to reduce the transmission latency and energy consumption.

Fig. 5 shows the performance comparison between CPS, BASE and OPS on transmission latency and energy consumption. We can see that both transmission latency and energy consumption of CPS are relatively low. Although the transmission latency of CPS is a little higher than that of the BASE, the energy consumption of CPS is much lower than that of the BASE. Similarly, although the energy consumption of CPS is a little higher than that of OPS, the transmission latency of CPS is much lower than that of OPS. The BASE has the lowest transmission latency but the highest energy consumption, this is because it requires all sensor nodes to be active, as the node density increases, more and more nodes participant in the concurrent forwarding process. OPS has the lowest energy consumption but the highest transmission latency, this is because it is based on point-to-point transmission instead of concurrent transmission, the transmission distance is shorter and the link quality is poorer compared with the concurrent transmission. CPS has a relatively low transmission latency because the concurrent transmission and pipeline scheduling enable the packet can be forwarded immediately once a node receives it. Meanwhile, CPS employs a dormant mechanism and selective forwarding strategy which reduces energy consumption.

5 Conclusion

In low-duty-cycle SWSNs, the unreliable wireless links make the pipeline schedule prone to fail which leads to high transmission latency. In this paper, we propose CPS, a CI-based pipeline scheduling scheme to overcome this problem. CPS reasonably divides the SWSN area into partitions and uses a pipeline mechanism to allocate active time slots for each partition. Nodes in the same partition wake up at the same time for concurrent transmission. Multiple identical signals interfere constructively at the receiver node, which enhances received signal strength and improves link quality. Simulations show that CPS can significantly reduce transmission latency while maintaining low energy consumption compared with other schemes.

Acknowledgment: This work is supported in part by the National Natural Science Foundation of China (Grant No. 61672282) and the Basic Research Program of Jiangsu Province (Grant No. BK20161491).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Cao, Q.; Abdelzaher, T.; He, T.; Stankovic, J. (2005): Towards optimal sleep scheduling in sensor networks for rare-event detection. *IEEE Fourth International Symposium on Information Processing in Sensor Networks*, pp. 20-27.

Cao, Y.; Guo, S.; He, T. (2012): Robust multi-pipeline scheduling in low-duty-cycle wireless sensor networks. *Proceedings IEEE INFOCOM*, pp. 361-369.

Chang, X.; Huang, J.; Liu, S.; Xing, G.; Zhang, H. et al. (2016): Accuracy-aware interference modeling and measurement in wireless sensor networks. *IEEE Transactions on Mobile Computing*, vol. 15, no. 2, pp. 278-291.

Chen, C. W.; Wang, Y. (2008): Chain-type wireless sensor network for monitoring long range infrastructures: architecture and protocols. *International Journal of Distributed Sensor Networks*, vol. 4, no. 4, pp. 287-314.

Cheng, D.; Mao, Y.; Wang, Y.; Wang, X. (2015): Improving energy adaptivity of constructive interference-based flooding for WSN-AF. *International Journal of Distributed Sensor Networks*, vol. 11, no. 6, 538145.

Doddavenkatappa, M.; Chan, M. C.; Leong, B. (2013): Splash: fast data dissemination with constructive interference in wireless sensor networks. *Presented as part of the 10th Symposium on Networked Systems Design and Implementation*, pp. 269-282.

Du, W.; Liando, J. C.; Zhang, H.; Li, M. (2016): Pando: fountain-enabled fast data dissemination with constructive interference. *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 820-833.

Ferrari, F.; Zimmerling, M.; Thiele, L.; Saukh, O. (2011): Efficient network flooding and time synchronization with glossy. *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 73-84.

Guo, P.; Jiang, T.; Zhang, K. (2012): Novel energy-efficient miner monitoring system with duty-cycled wireless sensor networks. *International Journal of Distributed Sensor Networks*, vol. 8, no. 1, 975082.

Guo, P.; Meratnia, N. (2015): OPS: opportunistic pipeline scheduling in long-strip wireless sensor networks with unreliable links. *Wireless Networks*, vol. 21, no. 5, pp. 1669-1682.

Guo, S.; Gu, Y.; Jiang, B.; He, T. (2009): Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. *Proceedings of MobiCom*, pp. 133-144.

Jawhar, I.; Mohamed, N.; Agrawal, D. P. (2011): Linear wireless sensor networks: classification and applications. *Journal of Network and Computer Applications*, vol. 34, no. 5, pp. 1671-1682.

Keshavarzian, A.; Lee, H.; Venkatraman, L. (2006): Wakeup scheduling in wireless sensor networks. *Proceedings of the 7th ACM International Symposium on Mobile Ad-hoc Networking and Computing*, pp. 322-333.

Sun, Y.; Du, S.; Gurewitz, O.; Johnson, D. B. (2008): DWMAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks. *Proceedings of MobiHoc*, pp. 53-62.

Stoianov, I.; Nachman, L.; Madden, S.; Tokmouline, T. (2007): PIPENET: a wireless sensor network for pipeline monitoring. *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pp. 264-273.

Wang, Y.; He, Y.; Mao, X.; Liu, Y.; Li, X. Y. (2013): Exploiting constructive interference for scalable flooding in wireless networks. *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 1880-1889.

Wang, Y.; Liu, Y.; He, Y.; Li, X. Y.; Cheng, D. (2014): Disco: improving packet delivery via deliberate synchronized constructive interference. *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 713-723.