Path Similarity based Spurious Retransmission Minimization over Flooding based Routing in UWSN

Yeongjoon Bae, Sungwon Lee, Yonghwan Jeong, and Dongkyun Kim
School of Computer Science and Engineering
Kyungpook National University
Daegu, Korea
{ yjbae, swlee, yhjeong }@monet.knu.ac.kr, dongkyun@knu.ac.kr

***Abstract*— In Underwater Wireless Sensor Networks, we faced a lot of challenges such as long propagation delay, and high packet loss. However, a flooding based routing protocol can address these challenges with its multipath characteristics. If the network uses a flooding based routing protocol, not only DATA but also ACK messages are transmitted through multiple paths. Although the messages are transmitted through multiple paths, we cannot avoid the packet loss completely. So, in order to provide high reliability of delivering messages, we need a retransmission mechanism. Though, if the network uses conventional transport layer protocol such as TCP, it will cause a spurious retransmission problem because it was not designed for the multipath environment. To overcome this problem, we suggest Path Similarity based Spurious Retransmission Minimization over Flooding based Routing in UWSN. Through our simulations, we verified that our scheme achieves the performance improvements of 10% ~ 45% in terms of retransmission ratio and 20% ~ 36% in terms of throughput compared to the Jacobson's RTO algorithm.**

***Keywords*— Spurious Retransmission Problem; Retransmission Timeout; Path Similarity; Flooding based Routing Protocol; Underwater Wireless Sensor Networks;**

I. INTRODUCTION

Since the propagation speed of the sound wave in the water is slower than the radio wave in the ground, the UWSN communication has a slower transmission rate than the terrestrial communication. Therefore, in order to reduce the time consumed in routing, underwater communication mainly uses a flooding-based routing protocol that does not require route searching procedure. In addition, underwater acoustic communication is characterized by a high error rate because it is easily interfered by multiple factors, i.e., surrounding creatures, ship engine sounds, etc. However, flooding-based routing protocols provide packet transmission reliability [1].

UWSNs have enabled researchers from both academia and industry to design various applications such as water temperature, water quality change surveillance, and coastal threat monitoring. In some applications, it may be necessary to transfer files between the sensor node and the sink node to update the sensor operation algorithm. File transfer requires high reliability as an error can result in the failure of the entire file transfer. However, due to the characteristics of UWSN with a high error rate, it is impossible to prevent the loss of the transmitted packet completely, even if the reliability of the packet transmission is increased through the multipath transmission. Therefore, it is imperative for the successful operation of the application that a function is designed that provides packet loss detection and retransmission.

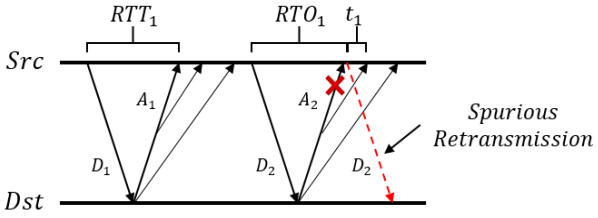

Figure 1. Spurious Retransmission Problem

In order to detect the packet loss, the destination node that receives the DATA transmits an ACK message, and the source node can use the retransmission technique if the ACK does not arrive before the RTO expires. However, in a network with a flooding-based routing protocol, not only DATA but also ACK messages are transmitted through multiple paths. Therefore, in order to use the retransmission technique in a network with a flooding-based routing protocol, it is necessary to consider that ACK can be received through multiple paths. However, according to the conventional operation of the transport layer retransmission protocol [2], the source node calculates the RTO by measuring the RTT as soon as one ACK is received. This can lead to the spurious retransmission problems. Figure 1 illustrates this scenario, where the ACK (A_1, A_2) transmitted from the destination node (hereinafter referred to as Dst) traverse three paths to reach the source node (hereinafter referred to as Src). Currently, Src measures RTT_1 through D_1 and A_1 and calculates RTO_1. Then, in the transmission of D_2, D_2 successfully arrives at Dst, but one of A_2 transmitted from Dst is lost. And since RTO_1 has expired, the source node retransmits D_2. However, if RTO_1 was only as long as t_1, it would have prevented the spurious retransmission problem of unnecessarily retransmitting D_2.

To solve this problem, the RTO must be long enough to receive additional ACKs. However, if the RTO is set long, the retransmission may be delayed and the throughput of the network may be deteriorated. Therefore, the RTO must be calculated to be able to receive the ACK transmitted over multiple paths. However, since additional ACK may also be lost, it is necessary to select an ACK that is less likely to be lost among the ACKs transmitted through the multipath and reflect that ACK in the RTO calculation. In this paper, we propose a scheme that uses path similarity to distinguish ACK with a low probability of loss. Path similarity is defined as the ratio of the same nodes through which the two paths share, it is further explained in Chapter III.

The rest of the paper is as follows. Chapter II describes previous techniques related to the issues covered in this paper. In Chapter III the proposed techniques are explained. In Chapter IV, we verify and explain the proposed scheme through simulation. Finally, Chapter V concludes this paper.

II. RELATED WORK

Previously, there have been several attempts to solve the spurious retransmission problem. In TCP Performance Improvement Considering ACK Loss in Ad Hoc Networks[3], authors proposed TCP-pgy to resolve the spurious retransmission problem caused by ACK loss over reactive routing protocol. And TCP and MPTCP Retransmission Timeout Control for Networks Supporting WLANs[4], authors proposed WLAN RTO(WRTO) algorithm to achieve the less spurious retransmission compared to existing schemes. But these schemes are designed for the terrestrial environment. However, there is a scheme to solve the spurious retransmission problem in the UWSNs that uses the flooding-based routing protocol. ACK Loss-aware RTO Calculation Algorithm over Flooding-based Routing[5] defines ACK Copies Waiting Time ($ACWT$) for waiting for the arrival of additional ACK, and reduces spurious retransmission by waiting for $ACWT$ after RTO expires respectively.

$$k_{cur} = \frac{(RTO - RTT_{cur})}{T_{ACK}} \quad (1)$$

$$k = \gamma \times k_{cur} + (1-\gamma) \times k_{pre} \quad (2)$$

$$ACWT = (k+1) \times T_{ACK} \quad (3)$$

First, Src calculates the initial RTO using the Jacobson [6] method when it receives the first arriving ACK. Src then sends the data to be sent in the next sequence and sets the timeout to RTO calculated previously. If an ACK copy of the previous DATA is received by Src before the set RTO expires, the time difference between the corresponding ACK copy and the immediately preceding ACK is measured. Src calculates the average time difference T_{ACK} between the arriving ACK copies and the immediately preceding ACK by the EWMA method while performing the packet transmission with this operation. If it is determined that a reliable T_{ACK} has been calculated, Src uses T_{ACK} to compute the expected number k of lost ACKs before receiving a successful ACK as in equation (1). Then, by multiplying T_{ACK} by a value obtained by adding 1 to the number of expected loss ACKs as in equation (3), the expected time $ACWT$ for successfully receiving one ACK can be calculated. Then Src increases the timeout by the $ACWT$ calculated after the RTO expires.

III. PROPOSED SCHEME

A. Motivation

The ACK Loss-aware RTO Calculation Algorithm over Flooding-based Routing technique described above equally reflects all ACKs received in calculating $ACWT$. However, if the path through the ACK is not taken into consideration, k_{cur} may not be calculated as intended.

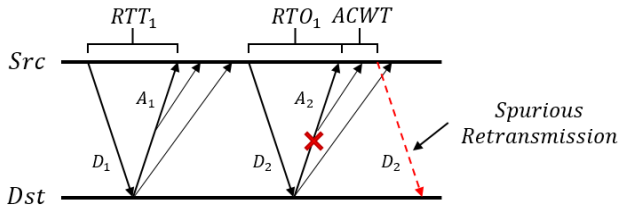

Figure 2. Impact of Path Similarity

Assuming the situation shown in Figure 2, the first, second, and third arriving ACKs for A_1 are defined as $A_{1-1}, A_{1-2}, A_{1-3}$, respectively. In addition, A_{1-1} and A_{1-2}, as shown in the figure, share many nodes. And A_{1-3} is assumed to be transmitted in a different path from these paths. In this case, if A_{2-1} is lost, there is a loss at a certain node on the path through A_{2-1}. Therefore, through similar path like A_{2-2} will also be lost. If all ACKs are equally reflected in the RTO calculation in this situation, spurious retransmission can occur as seen in the situation like A_2 in Figure 2. Therefore, this case also needs to be considered in the RTO calculation.

For this RTO calculation method, it is necessary to determine the path similarity of each packet. The path similarity refers to the ratio of the total node through which a path has been traversed and the same node which both paths share. In order to calculate the path similarity, there is the general technique of storing IDs of all the nodes that have passed through the packet in the header of the packet. However, using this technique causes a problem that the size of the header increases. Therefore, in this paper, we propose a method to determine path similarity using bitmap operation to solve this problem.

B. Assumption and Definition

In this paper, we assume the following situation. (1) The network administrator sets up a specific area in which the node is to be installed. (2) All nodes of the network are randomly installed within a specific area. (3) Divide a specific area into $N \times M$ lattices and assign a unique ID of integer type for each section. (4) Every node knows the information (section location, section ID, N, M, etc.). (5) There may be no node in one section, and one or more nodes may exist. (6) All nodes know the location of the sink (Dst). (7) $N \times M -$ bit Traveled Section (TS) field is added to all routing headers to determine path similarity. (8) When the packet is first generated, the TS field is all zero.

C. Path Similarity Calculation

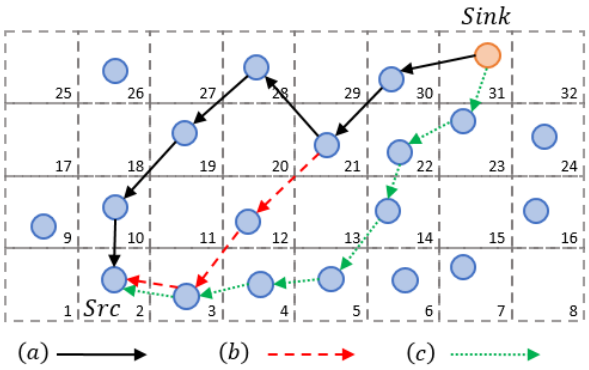

Figure 3. Network Topology and ACK Paths

In this technique, TS field of the packet header and the unique ID assigned to each section are used to measure the path of the packet. The Src that generated the packet broadcasts the packet while the TS field is 0. A node outside the flooding zone drops the packet when receive the packet. However, the node in the flooding zone changes the bits of the TS whose indexes are the same as those of the section ID to which the node belongs, to 1 before sending the packet. (if it is already 1, it does not modify it). If this operation is continued until the packet reached to Dst, Dst can know the path which the packet has been transmitted through the TS field in the header. For example, the TS field for Figure 3 (a) will be filled with 1 bits at 30, 21, 28, 19, and 10th bits in sequence, and will arrive at Dst as shown in Figure 4.

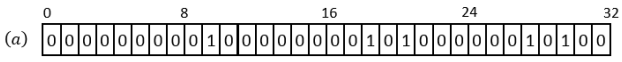

Figure 4. Traveled Section Field of Path (a)

Src that transmitted $DATA_1$ stores the TS field contained in the corresponding packet immediately after receiving the ACK_1 message from Dst. Then calculate the initial RTO RTO_{first} and use it to transmission of $DATA_2$. The initial RTO calculation equation is as follows[6].

$$RTO = RTT_{estimate} + \alpha \times RTTDEV \quad (4)$$

$$RTT_{estimate} = \beta \times RTT_{prev} + (1-\beta) \times RTT_{cur} \quad (5)$$

$$RTTDEV = \sum \left| RTT_i - RTT_{avg} \right| / N \quad (6)$$

In equation (4, 5), α and β are system parameters. Both values are set to 3.4 and 0.2 for the underwater environment [5]. And N is the number of received ACK messages.

In Src, if ACK_1 is received again before RTO_{first} expires, Src also determines whether to reflect the ACK in the RTO update based on the path similarity with the first arriving ACK and selectively updates the RTO. The equation for determining the similarity (Path Similarity, PS) between two paths is shown in the following equation (7).

$$PS(TS_{first}, TS_{cur}) = \frac{STSN(TS_{first}, TS_{cur})}{MIN(SC_{first}, SC_{cur})} \quad (7)$$

The SC (Section Count) means the number of intermediate sections through which the packet has been passed. This is calculated by the number of bits marked 1 in the TS field.

Also, the $STSN$ (Same Traveled Section Number) is the number of intermediate sections in which two packets are identically passed through. This is calculated by the result of the AND operation of the two TS fields.

For example, let us calculate the path similarity of (a) and (b) shown in Figure 3 using the above equations.

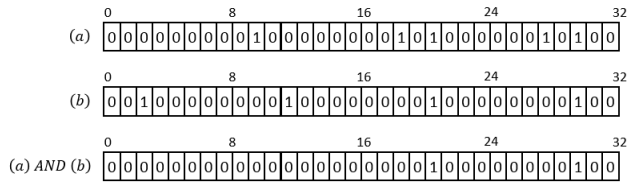

Figure 5. Result of AND operation between (a) and (b)

$$PS(TS_{(a)}, TS_{(b)}) = \frac{STSN(TS_{(a)}, TS_{(b)})}{MIN(SC_{(a)}, SC_{(b)})} = \frac{AND(TS_{(a)}, TS_{(b)})}{MIN(5,4)} = \frac{2}{4} \quad (10)$$

(a) and (b) pass the same two intermediate sections 21 and 30, so $STSN(TS_{(a)}, TS_{(b)})$ becomes 2, and $MIN(SC_{(a)}, SC_{(b)})$ becomes 4 and finally $PS(TS_{(a)}, TS_{(b)})$ is 0.5. In the same way, finding the path similarity of (b) and (c) and the path similarity of (c) and (a) yields a result of 0.25 and 0 in turn. As a result, it can be seen that the PS value becomes larger as the number of the same sections increases.

D. Path Similarity-based RTO Calculation

In Src, after transmitting the DATA, the initial RTO is calculated using the first received ACK and the next DATA transmitted. If an additional ACK for the previous DATA is received before the RTO expires, it is determined whether the corresponding ACK is to be reflected in the RTO calculation using the path similarity determined by the C technique. The proposed RTO calculation algorithm for Src is as follows.

Algorithm 1: The Algorithm of Path Similarity-based RTO Calculation

Current ACK: ACK_{cur}, Current ACK's TS: TS_{cur}, History of Received ACK's TS: $TS_{history}$, RTO calculated by First ACK: RTO_{first}, Maximum # of ACK to choose: ACK_{max}, Path Similarity Threshold: PS_{thresh}

1: **send** DATA
2: $i \leftarrow 0$
3: **if** ACK arrives ($ACK_{cur} \leftarrow$ ACK)
4: **if** $i = 0$
5: $TS_{history}[i] \leftarrow TS_{cur}$
6: $i \leftarrow i + 1$
7: **update** RTT & RTO with ACK_{cur}
8: **else if** $RTO_{first} < T$ **and** $i < ACK_{max}$
9: $decide \leftarrow \boldsymbol{true}$
10: **for**$(j = 0; j < i; j++)$
11: **if** $PS(TS_{history}[j], TS_{cur}) > PS_{thresh}$
12: $decide \leftarrow \boldsymbol{false}$
13: **if** $(decide)$
14: $i \leftarrow i + 1$
15: **update** RTT & RTO with ACK_{cur}
16: **else**
17: **drop** ACK_{cur}

Src transmits the DATA and initializes i (the number of ACKs reflected in RTO calculation) to 0 and waits for ACK reception. If an ACK arrives at this time, the ACK is the first received ACK, so it is selected as the ACK to be used for the RTO calculation independent to ACK_{max} (the maximum number of ACKs to reflect in the RTO calculation). Therefore, Src add TS_{cur} (TS field of the currently received ACK) as an element of the $TS_{history}$ array (TS field of the all ACKs reflected in the RTO calculation). And the RTO is calculated using ACK_{cur} (ACK currently received). Then, if another ACK arrives and i is less than ACK_{max} before RTO_{first} (RTO calculated by the first received ACK) expires, Src calculates PS (Path similarity) of $TS_{history}$ and TS_{cur}. If the calculated PS is less than or equal to PS_{thresh} (Path similarity threshold), add TS_{cur} as an element of the $TS_{history}$ and update the RTO using ACK_{cur}. Also, if RTO_{first} has expired or i has reached ACK_{max}, then all ACKs received afterwards are discarded.

ACK_{max} used in the above operation is the maximum number of ACK to be reflected in the RTO calculation. The smaller ACK_{max} the faster the retransmission occurs and the faster the packet loss can be recovered, and the larger ACK_{max}, the more spurious retransmission can be reduced considering more ACKs to arrive through multipath. In addition, PS_{thresh} is the threshold for the path similarity. As PS_{thresh} is low, only the ACK arriving at the path with low similarity is reflected in the RTO calculation. When the PS_{thresh} is high, the ACK arriving at the path with high similarity is also reflected. In this way, the network characteristics can be changed according to the value of the ACK_{max} and PS_{thresh}, and the network characteristics need to be changed according to the needs of the application. Therefore, it is assumed that ACK_{max} and PS_{thresh} are network parameters and are set according to the requirements of the application.

IV. EVALUATION

To observe the performance of the proposed method, experiments were conducted in Network Simulator (NS-2.34). The simulation environment is as follows. We generate a FTP traffic between source and destination node for simulating the retransmission ratio and throughput. Furthermore, the simulation results were obtained by averaging the results after running the simulations for 20 times. Table I contains the list of remaining simulation parameters.

We used the following quality metrics to evaluate the performance of the proposed scheme,

- **Retransmission ratio** is the $\frac{\#\ of\ Retransmission}{\#\ of\ Packets\ sent\ by\ Src}$
- **Throughput** is the amount of the data per unit time successfully transmitted from Src to Dst

TABLE I. SIMULATION ENVIRONMENT

Parameter	*Value*
Number of Nodes	250
Network Size	2000 * 1000 [m]
Topology Setting	Random Distribution
Transmission Range	100 [m]
Bandwidth	10 [kHz]
Payload Size	128 [KB]
Error Rate	10 [%]
Routing Protocol	DBR[1]
ACK_{max}	2
PS_{thresh}	0.25

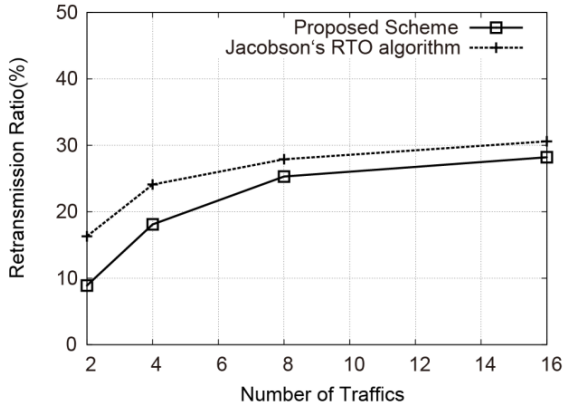

Figure 6. The result of Retransmission Ratio

First, we compared the retransmission ratio of the proposed scheme and conventional Jacobson's RTO algorithm [6] with different number of flows. As shown in Figure 6, proposed scheme produces less retransmission than the Jacobson's RTO algorithm regardless of the number of traffic. In the Jacobson's RTO algorithm, Src retransmits the data when its RTO is expired, even though the Dst successfully received the data and an ACK copy is going to arrive at the Src. On the other hand, in the proposed scheme, the Src minimizes a spurious retransmission by updating RTO selectively.

However, as the number of traffics increases, the network is faced with congestion that causes packet losses. Due to the increment of packet loss, both schemes must retransmit the data which are lost. These retransmissions are not spurious retransmission, so both schemes performance gap is reduced.

Next, in Figure 7, we compared the average throughput of the proposed scheme and Jacobson's RTO algorithm with a different number of flows. Jacobson's RTO algorithm has lower throughput regardless of the number of traffic since it could not minimize the spurious retransmissions.

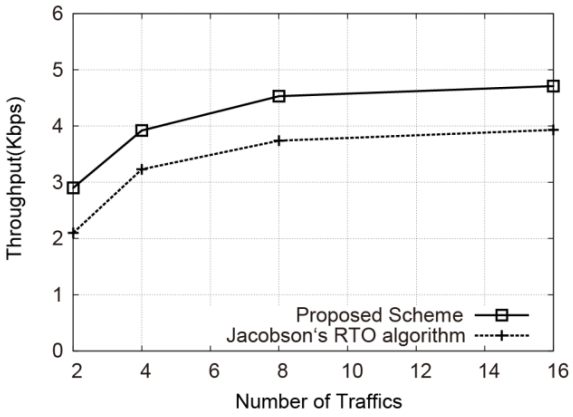

Figure 7. The result of Throughput

In addition, the increased network congestion results in failure to deliver the data. In such a situation, the performance gap between two schemes is reduced due to the longer RTO would cause the delay of retransmission that really needed.

V. CONCLUSION

In this paper, we aim to minimize the spurious retransmission problem over the flooding based routing protocol in UWSNs. We proposed a scheme which updates the RTO selectively by considering a path similarity of ACK among multiple ACK transmitted through different paths. To minimize a spurious retransmission, the RTO must be calculated such that the ACK transmitted over multiple paths can be received. Since additional ACK can be lost, it is necessary to select an ACK that is less likely to be lost among the ACKs transmitted through the multipath and reflect that ACK in the RTO calculation. To distinguish an ACK that is less likely to be lost, we considered a path similarity. Through our simulations, we verified that our proposed scheme achieves the performance improvements of 10%~45% in terms of retransmission ratio and 20% ~ 36% in terms of throughput compared to the Jacobson's RTO algorithm. In the future, we will further investigate the effect of ACK_{max} and PS_{thresh} on the network performance.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2016R1D1A3B01015510).

REFERENCES

[1] H. Yan, Z. Shi, and J. H. Cui, "DBR: Depth-based routing for underwater sensor networks," *IFIP Networking,* pp. 72-86, 2008.

[2] V. Paxson, M. Allman, J. Chu, and M. Sargent, Computing TCP's Retransmission Timer, RFC6298, 2011.

[3] D. Kim and H. Yoo, "TCP performance improvement considering ACK loss in ad hoc networks," in *Journal of Communications and Networks*, vol. 10, no. 1, pp. 98-107, March 2008.

[4] S. Shin, D. Han, H. Cho, J. M. Chung, I. Hwang and D. Ok, "TCP and MPTCP Retransmission Timeout Control for Networks Supporting WLANs," in *IEEE Communications Letters*, vol. 20, no. 5, pp. 994-997, May 2016.

[5] S. Lee, and D. Kim, "ACK Loss-Aware RTO Calculation Algorithm over Flooding-Based Routing Protocols for UWSNs," IEICE TRANSACTIONS on Information and Systems, 2014, pp. 2967-2970.

[6] V. Jacobson, "Congestion avoidance and control," ACM SIGCOMM, Computer Communication Review, pp.314-329, 1988.