LETTER

ACK Loss-Aware RTO Calculation Algorithm over Flooding-Based Routing Protocols for UWSNs

Sungwon LEE[†a)], ***Nonmember and*** **Dongkyun KIM**[†b)], ***Member***

SUMMARY In typical end-to-end recovery protocols, an ACK segment is delivered to a source node over a single path. ACK loss requires the source to retransmit the corresponding data packet. However, in underwater wireless sensor networks which prefer flooding-based routing protocols, the source node has redundant chances to receive the ACK segment since multiple copies of the ACK segment can arrive at the source node along multiple paths. Since existing RTO calculation algorithms do not consider inherent features of underlying routing protocols, spurious packet retransmissions are unavoidable. Hence, in this letter, we propose a new ACK loss-aware RTO calculation algorithm, which utilizes statistical ACK arrival times and ACK loss rate, in order to reduce such retransmissions.

key words: *UWSN, flooding based routing, TCP, retransmission, round trip time*

1. Introduction

Recently, much research interest in underwater wireless sensor networks (UWSNs) has increased to support many applications such as pollution sensing and tactical surveillance [1]. In UWSNs, since underwater communications between sensor nodes rely on an acoustic wave, they should cope with many limitations such as high propagation delay, noise, high loss ratio, etc. Among them, since the high loss ratio is a major cause of throughput degradation, many protocols have been designed to provide reliable packet transmissions in various layers such as PHY, MAC, and routing layers [2].

In particular, to route packets between a source node and the sink, flooding-based routing protocols which provide highly reliable packet delivery have been considered most appropriate for UWSNs [3], [4]. In these protocols, each sensor node determines a predefined forwarding area based on geographical information to control the number of forwarding nodes. Only the nodes located within the area participate in forwarding, however, the packets are still flooded along multiple paths in their own flooding area.

In addition to routing, a transport protocol is also needed to provide reliable end-to-end packet delivery, since packet loss is unavoidable while flooding packets. Moreover, some applications such as tactical surveillance in military environments require loss-free communication between the source node and the sink [5]. Packet loss might be recovered in hop-by-hop, but the hop-by-hop approach would lead to much channel contention especially in underwater environments. Therefore, it is desirable to develop an end-to-end (E2E) transport protocol which can recover the lost packets in underwater networks.

In most of the E2E recovery protocols such as TCP in the Internet, a source node sends a data segment and sets a timer called RTO (Retransmission Timeout). Then, it should wait for a corresponding ACK segment from the destination (the sink) [6]. If the RTO expires before the ACK segment arrives, the source concludes that the data segment has been lost in the network and it retransmits the missing data segment. Hence, the retransmission efficiency totally depends on the RTO calculation algorithm selected. Jacobson's RTO calculation algorithm [7] which is described in more detail in Sect. 2 is widely used to perform timely retransmission.

However, the original Jacobson's RTO calculation algorithm does not consider the existence of multiple ACK copies. Thus, if the RTO expires even though some ACK segments expected to arrive early have been lost and others ACK copies can be received later, spurious retransmissions are unavoidable in case that the algorithm might be applied to flooding-based routing in UWSNs. The network is flooded with unnecessarily retransmitted data segments, generating much traffic in the network as well as causing a lot of contention to channel access. In particular, this spurious retransmission (i.e. unnecessary retransmission) prevents the source from transmitting its next data segment until an ACK segment corresponding to the retransmitted data segment is received. Consequently, throughput degradation becomes serious.

Hence, the algorithm of calculating an accurate RTO is required to avoid spurious retransmission. However, since high loss rate in underwater environment makes it almost impossible to obtain the accurate RTO, the development of a loss-aware RTO calculation algorithm has the most importance in UWSNs.

In this letter, we propose an ACK loss-aware RTO calculation algorithm which considers an additional waiting time (called ACK Copies Waiting Time, ACWT) during which the source node awaits any other ACK copies even after the typical RTO expiration. Due to redundant chances to receive other ACK segments during ACWT which are flooded along multiple paths, the source node can avoid spurious retransmissions. However, if a large or small ACWT is taken, late or spurious retransmission is still performed, re-

Manuscript received June 25, 2014.
Manuscript revised July 31, 2014.
Manuscript publicized August 22, 2014.

[†]The authors are with the School of Computer Science and Engineering, Kyungpook National University, Korea.
a) E-mail: swlee@monet.knu.ac.kr
b) E-mail: dongkyun@knu.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2014EDL8127

spectively. Our proposed algorithm allows the source node to calculate the best ACWT, based on inter-arrival times of ACK copies and the ACK copy loss rate.

2. Related Work

Over the flooding-based routing, the source node can receive multiple ACK copies. However, the source node can decide that the data packet has been successfully delivered to the sink only if one ACK copy is received. Hence, the source node calculates the RTO based on the ACK copy which first arrives at the source node according to Jacobson's RTO calculation algorithm [7] (see Eq. (1)). Of course, an enhanced RTO calculation algorithm with the judicious consideration of characteristics of UWSNs (rather than Jacobson's RTO) should be developed, but it is out of scope of this work.

$$\begin{aligned} RTO\_typ &= RTT_{est} + \alpha * DEV_{RTT} \\ RTT_{est} &= \beta * RTT_{pre} + (1-\beta) * RTT_{cur} \\ DEV_{RTT} &= \sum |RTT_i - RTT_{avg}|/N \end{aligned} \tag{1}$$

RTO_{typ} denotes a typical RTO which is calculated by Jacobson's algorithm. RTT_{est} is an estimated RTT and DEV_{RTT} is mean deviation of RTTs. RTT_{pre} and RTT_{cur} is a previous estimated RTT and the current RTT, respectively. RTT_{avg} is an average RTT and N is the number of received ACK segments. α and β are system parameters.

3. Proposed Algorithm

As mentioned before, even after Jacobson's RTO [5] expires, the source node will have a chance to receive other ACK segments which are transmitted along different paths according to flooding-based routing. Hence, the source node should wait more time until other ACK copies are received after Jacobson's RTO expiration to avoid the spurious packet retransmissions. However, the source node should not wait for those ACK copies indefinitely. There could be a case that a data packet itself was not delivered to the sink due to packet loss. In addition, in case that the source node waits a very long time, retransmission would be delayed, leading to lower throughput. Hence, the source node should calculate ACK Copies Waiting Time (ACWT) that is an expected arrival time of a next ACK copy and waits more only during ACWT after Jacobson's RTO expiration.

Over most flooding-based routing protocols, some of ACK copies are expected to arrive within a period of time. Hence, based on the arrival time of each ACK copy, the source calculates the average inter-arrival time of ACK copies T_{ACK}. When the source node cannot receive any copies of an ACK packet during Jacobson's RTO expiration due to ACK loss, the next copy of this ACK packet would be arrive at the source node after T_{ACK}. If the ACK copy loss were absent, T_{ACK} would be enough for the source node to await the next ACK copy. However, ACK copies are still loss-prone in UWSNs, and the waiting time should be increased with the number of lost ACK copies. In particular, since the number of lost ACK copies cannot be measured exactly, the source node utilizes the estimated number of ACK copies (denoted by k) which have been lost until one ACK copy is received based on statistics of ACK arrival.

As mentioned before, as the number of lost ACK copies becomes larger, it takes more time for the source node to receive the next ACK copy. In other words, the gap between Jacobson's RTO expiration and the reception time of the ACK copy increases with the number of lost ACK copies. Hence, assuming that ACK copies arrive at the source node every T_{ACK}s, k becomes equal to this gap divided by T_{ACK}. Finally, in order to wait for an ACK copy which is expected not to be lost, the source node derives ACWT from the product of T_{ACK} and $k+1$.

According to most flooding-based routing protocols, T_{ACK} depends on the holding-time [3] which is a conventional technique to reduce packet collisions. In these protocols, forwarding nodes hold their packets during their holding-time which is set based on various metrics such as residual energy and geographical information. Since these metrics are stationary in a short period of time, the source node uses the latest estimated T_{ACK} in calculating the ACWT.

Different from T_{ACK}, k depends on channel quality. In UWSNs, since the channel quality is variable, the source node estimates the average k based on the reception history of ACK copies. However, an arithmetic mean is not appropriate to estimate the average k, since it puts the same weight to each of samples. Hence, in order to reflect the current channel quality better, the source node estimates the average k and calculates ACWT according to the EWMA formation and Eq. (2), respectively.

$$\begin{aligned} k_{cur} &= (RTO - RTT_{cur})/T_{ACK} \\ k &= \gamma * k_{cur} + (1-\gamma) * k_{pre} \\ ACWT &= (k+1) * T_{ACK} \end{aligned} \tag{2}$$

RTO and RTT_{cur} are Jacobson's RTO and the current RTT, respectively. T_{ACK} is the latest estimated T_{ACK}. k_{cur} and k_{pre} are the current estimated k and previous estimated k, respectively. γ is given as a system parameter.

4. Performance Evaluation

4.1 Simulation Environments

Through the NS-2 simulator, we implemented our proposed algorithm and Jacobson's algorithm into the stop-and-wait protocol, not pipelined protocols. In the stop-and-wait protocol, a source is allowed to transmit only one data packet at a time before receiving its corresponding ACK packet [8]. On the other hand, the pipelined protocols enable the source to transmit multiple data packets without waiting for an ACK packet (no more than some maximum allowable number of packets in the pipeline). However, the pipeline protocols require a new congestion control mechanism for UWSN to be devised, which is out of scope of this work.

The interest of this work lies in the E2E RTO-based reliable data transfer mechanism.

As our baseline flooding-based routing protocol, the DBR [3] was chosen with the broadcasting mode of IEEE 802.11 MAC protocol. We placed 250 sensor nodes at random locations in the square of 1500m × 1000m among which one sink was located in the center of the sea surface. The source nodes were chosen randomly among the nodes located lower than 800m from the sea surface to generate multiple paths between the sink and source nodes. They generated an FTP traffic for 1800 seconds towards the sink. The data packet size of the traffic was set to 128Kbytes. We repeated our simulation 50 times and measured average performances in terms of retransmission ratio and throughput, respectively.

To emulate high error rate in UWSNs, link error rates of 8% and 3% were set for DATA and ACK segment transmissions, respectively. Since the size of a DATA segment is larger than that of an ACK segment, the link error rate for the DATA segment was assumed to be higher. The maximum transmission range of a sensor node and a bandwidth were set to 300m and 10kHz, respectively.

Moreover, in terrestrial networks such as the Internet, α and β values in Jacobson's algorithm are set to 4 and 0.125. It is known that these values were found using the empirical method. In order to select appropriate values in underwater environments, we also found the best ones for them empirically (α, β and γ value are set to 3.4, 0.2 and 0.25, respectively).

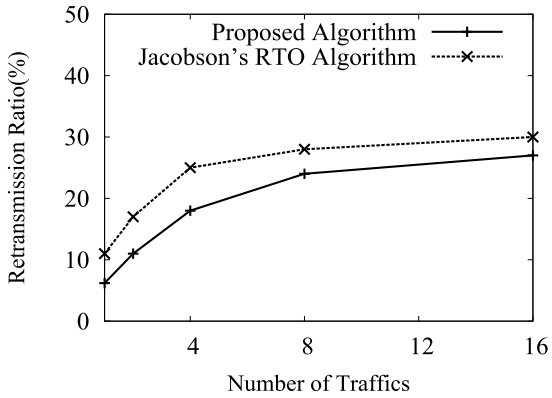

Fig. 1 Retransmission Ratio.

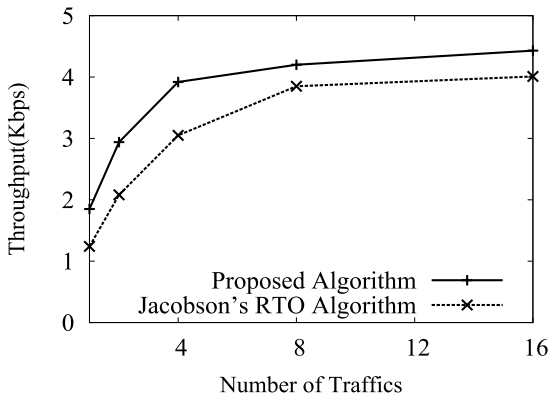

Fig. 2 Throughput.

4.2 Simulation Results

First, we measured retransmission ratio of our proposed algorithm and Jacobson's RTO algorithm with different number of flows (Fig. 1). Regardless of the number of traffic flows, our proposed algorithm outperforms Jacobson's RTO algorithm in terms of lower retransmission ratio. In Jacobson's RTO algorithm, despite the successful reception of data packets at the sink and existence of other ACK copies in the network, the source node would retransmit the data packets if an ACK copy did not arrive before its RTO expiration. In contrast, in our proposed algorithm, the source node avoids spurious retransmissions because it waits for an ACK copy additionally during the ACWT.

However, as the number of traffic flows increases, the network gets more congested and the number of lost data packet increases accordingly. Since both of algorithms should retransmit data packets which are actually lost, their performance gap is reduced. In particular, when the network becomes congested (e.g., the number of traffic flows is larger than 4), high retransmission ratio (almost 30%) is observed, regardless of the implemented algorithms due to bottleneck effect around the sink. The deployment of multiple sinks could avoid the bottleneck effect, which is our future work.

Second, we conducted the comparisons of their total average throughput according to the various number of traffic flows (see Fig. 2). Jacobson's RTO algorithm has lower throughput, since it could not avoid the spurious packet retransmissions, which causes the network resource to be wasted and deteriorate the network congestion. However, in our proposed algorithm, the throughput slightly decreases only if a large waiting time is taken. Still, it is observed that our proposed algorithm outperforms Jacobson's RTO algorithm with higher throughput, regardless of the number of traffic flows. In addition, as the network gets more congested, it is more likely that a data packet itself is not delivered to the sink. In this case, the gap is reduced since the additional ACWT would cause the source node to delay its retransmissions.

5. Conclusion

In this letter, we proposed a new ACK loss-aware algorithm which calculates a retransmission timeout (RTO) value for end-to-end packet recovery in order to reduce spurious retransmissions over the flooding-based routing protocols for UWSNs. In loss-prone UWSNs, a large amount of ACK loss can force the source node to retransmit packets unnecessarily. Hence, in our proposed algorithm, the source node increases the RTO in proportion to the number of ACK copies which are lost before the reception of one ACK copy. Since the number of lost ACK copies cannot be measured accurately, the source node estimates this number based on

inter-arrival times of ACK copies and the reception time of one ACK copy.

Through NS-2 simulations, we verified that our proposed algorithm could achieve performance improvements of 11% ∼ 46% and 14% ∼ 55% in terms of retransmission ratio and throughput, respectively.

Acknowledgments

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD130007DD.

References

[1] E.M. Sozer, M. Stojanovic, and J.G. Proakis, "Underwater acoustic networks," J. Oceanic Engineering, pp.72–83, 2000.
[2] I.F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: Research challenges," Ad Hoc Networks Journal, pp.257–279, 2005.
[3] H. Yan, Z. Shi, and J.H. Cui, "DBR: Depth-based routing for underwater sensor networks," IFIP Networking, pp.16–1221, 2008.
[4] D. Hwang and D. Kim, "DFR: Directional flooding-based routing protocol for underwater sensor networks," Proc. MTS/IEEE OCEANS, pp.1–7, 2008.
[5] H. Chin-Ya, P. Ramanathan, and K. saluja, "Routing TCP flows in underwater mesh networks," IEEE J. Sel. Areas Commun., vol.29, pp.2022–2032, 2011.
[6] V. Paxson and M. Allman, "Computing TCP's retransmission timer," RFC 2988, 2000.
[7] V. Jacobson, "Congestion avoidance and control," ACM SIGCOMM, Computer Communication Review, pp.314–329, 1988.
[8] S. Lin, D. Costello, and M. Miller, "Automatic-repeat-request error-control scheme," IEEE Commun., vol.22, pp.5–17, 1984.