Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 160541, 9 pages
http://dx.doi.org/10.1155/2014/160541

Research Article

Two Fast Retransmit Techniques in UWSNs with ACK Indiscretion Problem

Sungwon Lee and Dongkyun Kim

Kyungpook National University, Daegu 702-701, Republic of Korea

Correspondence should be addressed to Dongkyun Kim; dongkyun@knu.ac.kr

Received 17 August 2013; Accepted 4 December 2013; Published 13 February 2014

Academic Editor: Wei Wang

In underwater wireless sensor networks (UWSNs), flooding-based routing protocols are preferred due to their capability of reducing the routing overhead in terms of no need of path setup and maintenance. In addition to routing, a transport protocol should be developed to recover lost DATA segments in loss-prone UWSNs. In particular, due to long propagation delay in UWSNS, a fast end-to-end recovery technique needs to be developed. Fortunately, the Fast Retransmit technique well-defined in TCP can be exploited for fast retransmissions of lost DATA segments. However, if it operates over the flooding-based routing protocols, each source node will receive multiple copies of ACK segments which are transmitted along different paths. Therefore, the source node cannot distinguish between these multiple copies of ACK segments and real duplicated ACK segments (defined as an ACK indiscretion problem in this paper), which leads to unnecessary retransmissions. In this paper, we therefore propose two Fast Retransmit techniques to address the ACK indiscretion problem. In our first proposed technique, the sink records a new count number into a header, informing how many duplicated ACK segments have been transmitted from the sink. Since this requires an additional field in the header for the count number, the second technique allows a source node to estimate the number of ACK copies which are expected to be received when the sink transmits an ACK segment, without any dependency on the additional field. From both of our proposed techniques, the source nodes become aware of the accurate number of duplicated ACK segments transmitted from the sink and can perform the Fast Retransmit correctly.

1. Introduction

Recently, researchers have paid much attention to underwater wireless sensor networks (UWSNs) in order to support a variety of applications such as seismic monitoring, oceanographic data collection, and tactical surveillance in the underwater environment.

Different from terrestrial sensor networks, UWSNs have distinctive characteristics such as high propagation delay, limited bandwidth, and high error rate since acoustic signals are used for communications, rather than radio signals [1, 2]. Hence, their communication protocols for UWSNs should be developed to take into account these characteristics.

In particular, many routing protocols have been proposed to enable a packet from each sensor node in underwater to reach the sink on the sea surface [3, 4]. In the terrestrial networks, ad hoc routing protocols such as DYMO [5] are widely used to establish a path from the source to destination. However, these routing protocols require high routing overhead to set up and maintain the established paths. Due to the high propagation delay of the acoustic signal, other routing protocols which can reduce such routing overhead should be devised.

In UWSNs, flooding-based routing protocols are preferred due to their capability of reducing the routing overhead in terms of no need of path setup and maintenance [6–10]. Moreover, these routing protocols can increase the packet delivery ratio by allowing multiple copies of a packet to reach the sink along different paths.

In addition to routing, a transport protocol is still needed to provide the end-to-end reliability of packet delivery. Partially, some military applications such as tactical surveillance require a high level of reliability [11]. In order to support these applications in loss-prone UWSNs, a technique which can detect and recover lost packets should be devised, since the flooding-based routing protocols themselves cannot guarantee the successful packet delivery to the sink.

In order to guarantee the successful packet delivery, an ACK-based retransmission technique is required [11–14]. Particularly, since hop-by-hop ACK-based packet recovery causes much channel contention and lowers the transmission efficiency obtained by link-level broadcasting, an end-to-end ACK-based retransmission technique is preferred. In addition, although other techniques such as FEC [15, 16] which can increase the packet delivery ratio are employed, they require a lot of overhead in UWSNs which have narrow bandwidth.

In most of end-to-end ACK-based retransmission techniques, the absence of an ACK segment during a retransmission timeout requires the source node to retransmit the missing DATA segment. Such a timeout-based retransmission technique causes a long latency to a successful retransmission in UWSNs suffering from long propagation delay.

Fortunately, the Fast Retransmit technique [14] which is well-defined in TCP (transmission control protocol) and widely used in most TCP versions can be exploited to reduce such latency. In the Fast Retransmit technique, the sink transmits a duplicated ACK segment after detecting packet loss. When the source node receives the same four ACK segments (one normal ACK, three duplicated ACKs) from the sink, the source node retransmits the lost DATA segment immediately.

However, when the Fast Retransmit technique is applied to flooding-based routing protocols, the source node will still suffer from an ACK indiscretion problem which is described in more detail in the next section and it will perform unnecessary retransmissions, which may deteriorate network congestion, if there is any.

In this paper, we therefore propose two Fast Retransmit techniques, called FDS (Fast Retransmit technique based on duplicated ACK sequence number) and FDC (Fast Retransmit technique based on Duplicated ACK Count) which can work appropriately with the flooding-based routing protocols. In FDS, the sink records a number of duplicated ACK segments which have been transmitted from the sink into a header. Based on the recorded number, the source node is able to distinguish between the duplicated ACK segments and multiple copies of the ACK segment. However, in FDS, we need an additional field for including the number of duplicated ACK segments into the header. Fortunately, if FDS is implemented in TCP which has the well-defined Fast Retransmit technique, it can record the number of duplicated ACK segments using some of reserved bits of TCP header, which can avoid the need of the additional overhead of bits. However, as the reserved bits of TCP header have already been utilized for other purposes, we need to devise a new method not to depend on those reserved bits and we therefore propose a Fast Retransmit technique, FDC, which utilizes a local variable at a source node. In FDC, the source node calculates an average number of ACK copies which are expected to be received when the sink transmits an ACK segment. Based on the number, the source node estimates how many ACK segments have been transmitted from the sink and it performs the Fast Retransmit correctly.

The rest of this paper is organized as follows. In Section 2, some related works on flooding-based routing protocols and Fast Retransmit technique are described. In Section 3, the ACK indiscretion problem is described in more detail. In Sections 4 and 5, our proposed Fast Retransmit techniques, FDS and FDC, are presented, followed by performance evaluations in Section 6. Finally, conclusion remarks are given in Section 7.

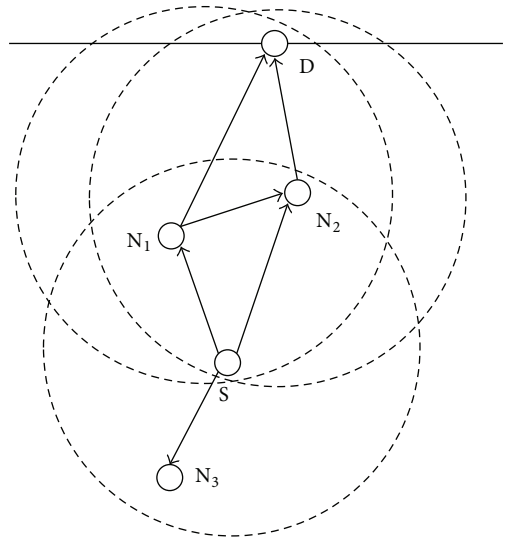

Figure 1: Packet transmissions in DBR.

2. Related Works

In this section, the flooding-based routing protocols preferred for UWSNs and the Fast Retransmit technique well-defined in most of TCP versions are described separately below.

2.1. Flooding-Based Routing Protocol. In loss-prone UWSNs, flooding-based routing protocols have been introduced not only to reduce routing overhead but also to increase the likelihood of packet delivery. These protocols define their different flooding area using location or depth information in order to reduce unnecessary packet transmissions. The packets are flooded along the multiple path in their own flooding area.

DBR [6] which defines its flooding area using depth information is one of the simple flooding-based routing protocols. Since the sink is located on the sea surface in common UWSNs, packets should be transmitted towards the sea surface. Hence, in DBR, all nodes which are located at lower depth from the surface than a current forwarding node can participate in next forwarding. Figure 1 depicts an example of packet transmission in DBR. S is a sender node, D is a sink, and N_1, N_2, and N_3 are intermediate nodes. After receiving packet from S, N_3 drops the received packet sink since it is located in deeper depth from the sea surface. In contrast, N_1 and N_2 broadcast the received packet since they are closer from the sea surface.

Since DBR uses only depth information to define its flooding area, it allows packets to be transmitted in different directions, not towards the sink. Obviously, flooding-based routing protocols such as vector-based flooding protocol (VBF) [8] could avoid unnecessary transmissions not towards

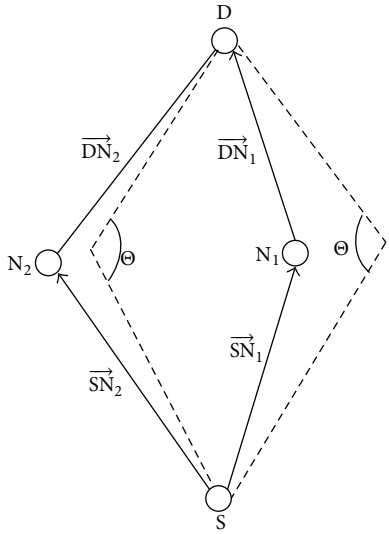

Figure 2: Packet transmissions in DFR.

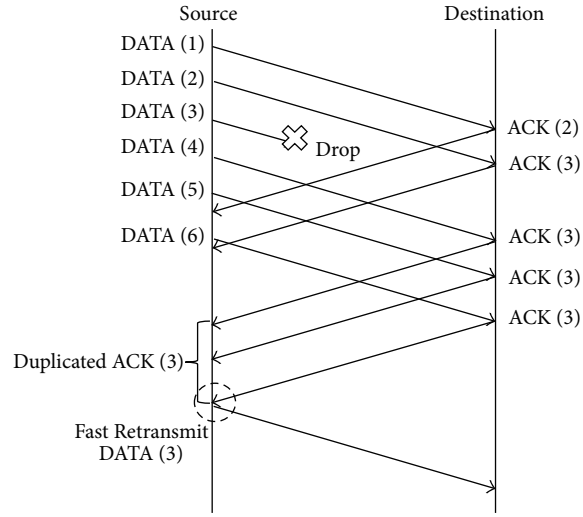

Figure 3: Fast Retransmit technique in TCP.

the sink, using geographical information [16]. In VBF, each intermediate node determines whether to forward or drop the packet according to the distance between itself and a routing vector which is a vector from the source node to the sink. If the distance between an intermediate node and the routing vector is longer than a predefined distance threshold, the node drops the received packet. Otherwise, the node participates in the next forwarding. However, VBF can still generate unnecessary packet transmissions or provide low reliability since a predefined distance threshold is utilized regardless of link quality. Hence, in our previous work, we proposed a directional flooding-based routing protocol (DFR) [7] which defines its flooding area according to both geographical information and link quality. In DFR, each node N calculates two vectors, $\overrightarrow{SN}$ and $\overrightarrow{ND}$, where S is a source node and D is the sink. If an angle between $\overrightarrow{SN}$ and $\overrightarrow{ND}$ is smaller than a predefined angle θ, node N participates in the next forwarding. Since the defined angle θ is controlled based on average link quality, the number of nodes which participate in the next forwarding depends on link quality which is calculated according to ETX metric [17]. Figure 2 describes an example of packet transmissions in DFR. In this example, the angle which is calculated at node n2 and which is greater than θ drops the received packet. In contrast, node n1 participates in the next forwarding since its calculated angle is smaller than θ. Of course, if link quality becomes too bad, θ is increased and n2 will participate in the next forwarding.

2.2. Fast Retransmit in TCP. In order to provide end-to-end reliability, end-to-end ACK-based retransmission techniques are defined in various transport protocols such as TCP-Vegas [13] and TCP-Reno [14]. In most of end-to-end ACK-based retransmission techniques, when the destination (the sink) receives a DATA segment successfully, it sends an ACK segment to inform the source node of its reception. If the source node cannot receive the ACK segment during a retransmission timeout (RTO) [18], it retransmits the missing DATA segment. In order to calculate the RTO, the Jacobson RTO calculation algorithm [19] is well-defined and widely used to perform the timely retransmission. However, such a timeout-based retransmission technique causes a long latency to a successful retransmission in UWSNs which suffer from long propagation delay.

To reduce the latency, the Fast Retransmit technique [14] which is well-defined in most TCP versions can be employed. According to this technique, the source node detects a loss of DATA segments, based on the reception of duplicated ACK segments. If a DATA segment is lost during transmission, the destination will receive an out-of-order DATA segment. In this situation, the destination transmits a duplicated ACK segment to inform the source node of a next expected sequence number of a DATA segment. Hence, when the duplicated ACK segment is received at the source node, the source node can detect the loss of a DATA segment. However, if the source node transmits the DATA segment immediately after detecting the DATA segment loss, spurious DATA segment retransmissions cannot be avoided when the DATA segment transmission is temporarily delayed [20]. In order to avoid such spurious retransmissions, the Fast Retransmit technique is defined in the TCP standard, using the arrival of 3 duplicated ACKs as an indication that a segment has been lost.

Figure 3 shows an example of the Fast Retransmit technique. In this example, the source node transmits DATA segments (1), (2), (3), (4), (5), and (6). If the DATA segment (3) is lost during transmission, the destination will transmit one normal ACK segment and three duplicated ACK segments. After receiving the third duplicated ACK segment, the source node retransmits DATA segment (3) immediately.

3. ACK Indiscretion Problem with Flooding-Based Routing Protocols

As mentioned before, in order to guarantee the successful packet delivery, the ACK-based retransmission technique is desirable. Moreover, the Fast Retransmit technique can be

exploited in order to reduce a retransmission latency. However, the Fast Retransmit technique faces an ACK indiscretion problem newly defined in this paper when applied to flooding-based routing protocols.

Due to the characteristics of the flooding-based routing protocols, the sink node may receive multiple copies of a DATA segment along different paths. After receiving the same DATA segments, the sink can simply drop the DATA segments which have been already received. Hence, there is no problem in the handling of multiple copies of a DATA segment transmitted from sensor nodes to the sink.

In addition to the DATA segment transmission, the ACK segment transmission should also depend on the underlying flooding-based routing protocols. Hence, source nodes may receive a lot of multiple copies of an ACK segment along different paths. Furthermore, even when the sink transmits a duplicated ACK segment after detecting packet loss, multiple copies of the duplicated ACK segment may still reach the source node. Despite the existence of multiple copies of different types of ACK segment, source nodes cannot distinguish between these ACK segments and they regard them all as duplicated ACK segments.

This phenomenon is defined as the ACK indiscretion problem in this paper. The ACK indiscretion problem can cause the Fast Retransmit to misbehave, requiring the sensor node to perform unnecessary retransmissions of the DATA segment. This happens whenever the source node receives four ACK segments with the same sequence number along different paths, regardless of the received ACK segment type. Hence, this unnecessary retransmission wastes scarce network resource in UWSNs through flooding, which may deteriorate network congestion, if there is any.

Of course, different routing protocols can be applied in order to avoid the ACK indiscretion problem; a flooding-based routing protocol for the DATA transmission from sensor nodes to the sink and a nonflooding-based routing protocol for the ACK segment transmission from the sink. However, it is required that each sensor should have dual routing protocols, which is not desirable in terms of the complexity of implementation.

4. Fast Retransmit Based on Duplicated ACK Sequence Number

In order to address the aforementioned ACK indiscretion problem, FDS (Fast Retransmit based on duplicated ACK sequence) which is one of our proposed techniques enables each source node to correctly distinguish between the duplicated ACK segments that the sink has transmitted and multiple copies of the ACK segment.

In FDS (Figure 8), a new count (DACK_cnt: duplicated ACK count) is defined additionally besides the sequence number (ACK_seq) which includes the next expected sequence number of a DATA segment. DACK_cnt represents how many different duplicated ACK segments have been transmitted from the sink. Since the intermediate sensor nodes are not allowed to set the DACK_cnt, the multiple copies of the ACK segment include the same DACK_cnt. Also, the duplicated ACK segments which are newly transmitted from the sink include different DACK_cnt. Therefore, according to the DACK_cnt, the source node can distinguish the type of ACK segment and retransmit lost DATA segments.

4.1. Operation of Sink. In order to overcome the ACK indiscretion problem at source nodes, the sink should record the number of duplicated ACK segments which have been transmitted from the sink into DACK_cnt field in a duplicated ACK segment. To maintain this number, the sink manages a local value, ACK_send_cnt, which is initially set to 0.

The sink's detailed operation is as follows. When the sink receives a DATA segment, it determines whether the received DATA segment is a DATA segment which is currently expected to receive it or not, based on a sequence number of the DATA segment (DATA seq). If the expected DATA segment has been received, the sink computes the next expected sequence number of DATA segment and resets the ACK_send_cnt to zero. After that, the sink sends an ACK segment whose ACK_seq and DACK_cnt are set to the computed sequence number of DATA segment and ACK_send_cnt, respectively.

Otherwise, if the sink detects DATA segment loss, it increases the ACK_send_cnt by 1 and sends a duplicated ACK segment with the DACK_cnt set to the new computed ACK_send_cnt.

Moreover, if the sink receives multiple copies of a DATA segment, it might send a lot of duplicated ACK segments since these copies are regarded as unexpected DATA segments. Hence, the sink should drop the other multiple copies of the received DATA segment. When the sink receives the DATA segments which include the same DATA seq, it computes the gap between the reception time of the previous one. The sink drops the received DATA segment if the computed gap is less than the minimum length of time for the DATA segment retransmission which is generally defined in ACK-based retransmission techniques because it is a copy of the previously received DATA segment.

4.2. Operation of Source Node. If a source node receives an ACK segment which is not duplicated, it sends a new DATA segment or discards the received ACK segment according to the ACK_seq. Otherwise, if the source node receives an ACK segment which includes the same ACK_seq, it should execute the Fast Retransmit only when the value in the DACK_cnt field is greater than or equal to 3. In addition, in order to address the ACK indiscretion problem, the source node discards the received ACK segments when they are multiple copies of the already received ACK segment.

Here, the detailed operation of each source node is described below. In receiving an ACK segment, a source node compares the ACK_seq included in the received ACK segment with the maximum ACK sequence among previous ACK segments. If the received ACK segment is an ACK segment arriving late, it will be dropped. Otherwise, if the ACK_seq in the received ACK segment is greater than the latest ACK sequence number, the source node will send its new DATA segments [12] (Algorithm 1).

If the ACK_seq in a received ACK segment is the same with the ACK_seq in the previous ACK segment, the source

```
(1) MAX_ACK_seq is the maximum ACK sequence among previous received ACK segments;
(2) Last_FR_seq is a sequence number of the last fast retransmitted DATA segment;
(3) DACK_num is a DACK_cnt of the last received ACK segment;
if (ACK_seq < MAX_ACK_seq) then
   Discard an ACK segment;
end if
if (ACK_seq > MAX_ACK_seq) then
   MAX_ACK_seq ← ACK_seq;
   Transmit a new DATA segment;
end if
if (Last_FR_seq ≠ DATA_seq and DACK_cnt ≥ 3) then
   Retransmit a lost DATA segment;
else
   if (DACK_num = DACK_cnt) then
      Discard an ACK segment;
   end if
   DACK_num ← DACK_cnt;
end if
```

Algorithm 1: Operation of each source node (in FDS).

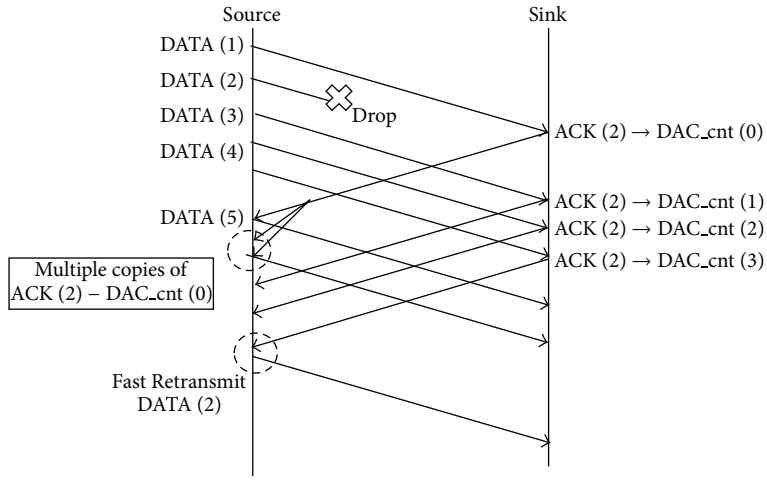

Figure 4: Operation in FDS.

node should operate differently according to the value in the DACK_cnt field, not the number of received duplicated ACK segments. If the value of DACK_cnt is greater than or equal to 3, the source node retransmits the corresponding DATA segment. Moreover, when the corresponding DATA segment has been already retransmitted, the Fast Retransmit should be stopped since the additional retransmissions can incur packet collisions.

Otherwise, if the value of DAC_cnt is less than 3, the corresponding DATA segment is not retransmitted. However, independent of the DATA segment retransmissions, the reception of duplicated ACK segments can trigger other operations in transport layer such as the part of congestion control. Hence, in order to support such operations, the source node regards the received ACK segment as a duplicated ACK segment when the DACK_cnt in the received ACK segment is different from the DACK_cnt in the previous ACK segments.

Figure 4 shows an example of the FDS. In this example, the source node transmits DATA segments (1), (2), (3), (4), and (5). If the DATA segment (2) is lost during transmission, the sink will transmit 4 ACK segments which include different DAC_cnt. After receiving the ACK segment and the value of DAC_cnt becomes equal to 3, the source node retransmits DATA segment (2) immediately.

5. Fast Retransmit Based on Duplicated ACK Count

As mentioned before, FDS attempts to use some of reserved bits of TCP header to record DACK_cnt for reducing the transmission overhead. However, if all those bits have already been used for other purposes, there exists no room for FDS in the header. Therefore, another proposed Fast Retransmit

technique called FDC (Fast Retransmit based on duplicated ACK count) can address the problem, which does not require any additional header field, but through a local variable (Figure 9).

In FDC, the source node assumes that a new ACK segment is flooded along the same set of multiple paths as the ACK segment transmitted previously and the source node will receive the same number of ACK copies as its previous one. Hence, if a source node receives more ACK copies than the previous one, the source node can detect that some duplicated ACK segments have been transmitted. In this case, if the number of duplicated ACK segment becomes over three according to the basic Fast Retransmit defined in the TCP's Fast Retransmit, the source node performs the Fast Retransmit correctly.

However, in case that a new ACK segment is flooded along different set of multiple paths, FDC does not operate as expected, since the source node cannot estimate the accurate number of duplicated ACK segments which have been transmitted from the sink. Hence, in order to support dynamic changed set of multiple paths, FDC requires a sophisticated path control method which is our future work.

5.1. Operation of Sink. Different from FDS, FDC does not require an additional operation of the sink. The sink performs basic operations which are well-defined by transport protocols such as TCP [12].

When the sink receives a DATA segment, it checks whether the received DATA segment is a DATA segment which is currently expected to receive, based on a sequence number of the DATA segment (DACK_cnt). If the expected DATA segment has been received, the sink sends an ACK segment which includes a newly computed ACK_seq. Otherwise, if the sink receives out-of-order DATA segments, it sends a new duplicated ACK segment which includes the next expected sequence number of a DATA segment.

5.2. Operation of Source Node. In receiving an ACK segment, a source node compares the ACK_seq inside the received ACK segment with the maximum ACK sequence among the sequences of the ACK segments received previously. If the received ACK segment is an ACK segment which arrives late, it is dropped. Otherwise, if the ACK_seq in the received ACK segment is greater than the latest ACK sequence number, the source node will send its new DATA segments and calculate an average number of ACK copies which is expected to be received at the source node when the sink transmits a new ACK segment. The average number of ACK copies is denoted by ADC (average duplicated ACK count) in this paper, which is calculated according to (1). If the amount of ACK segments which include the same ACK_seq is greater than quadruple of ADC, the source node fast-retransmits the lost DATA segment.

$$\mathrm{ADC}_{(t)} = \alpha * \mathrm{ADC}_{(t-1)} + (1-\alpha) * \left(\frac{\mathrm{ACK\_recv}}{\mathrm{Cwnd}}\right). \quad (1)$$

$\mathrm{ADC}_{(t)}$ and $\mathrm{ADC}_{(t-1)}$ are a new ADC value and a previous ADC value, respectively. ACK_recv is the number of received ACK copies which includes the same ACK_seq. Cwnd is a congestion window size which is an amount of DATA segments that a source node can transmit before it has to wait for an ACK segment to proceed. If FDC is employed in non-pipelined transport protocols, Cwnd is set to 1. α is a system parameter.

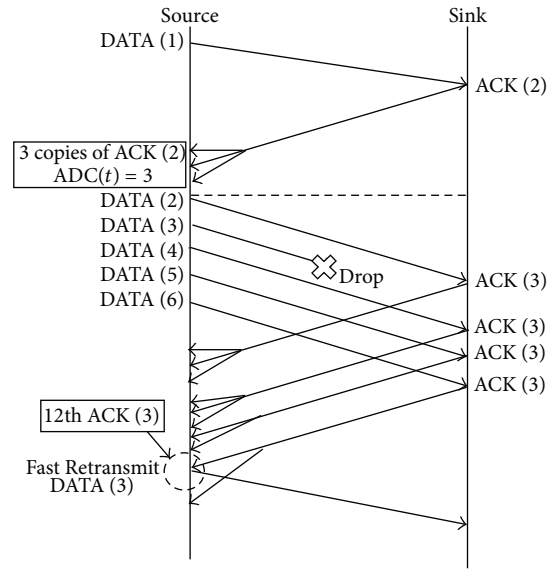

Figure 5: Operation in FDC.

After the new ADC calculation, the source node increases local value (denoted by ACK_cnt) whenever an ACK segment which includes the same ACK_nt as the previous ACK segment is arrived. If ACK_cnt is smaller than the quadruple of ADC, the source node drops the received ACK segment. Otherwise, if ACK_cnt is greater than the quadruple of ADC, the source node fast-retransmits the correspondent DATA segment.

Figure 5 shows an example of the FDC. In this example, the source node transmits a DATA segment (1) and the sink replies with one ACK segment. Due to the flooding, the source node receives the same three ACK segments and it calculates ADC_cnt value to 3. After then, the source node transmits consecutive DATA segments (2), (3), (4), (5), and (6). If the DATA segment (3) is lost during transmission, the sink will transmit four ACK segments (3). Since the ADC_cnt value is 3, the source node retransmits the DATA segment (3) when the 12th ACK segment is received (Algorithm 2).

6. Performance Evaluation

6.1. Simulation Environments. Through the NS-2 simulator, we tested the feasibility of our proposed Fast Retransmit techniques for UWSNs. We conducted their performance comparisons against the Fast Retransmit without the ACK distinction technique. For the rest of operations of a transport protocol, the TCP-Reno was employed in our simulations, since it has already well-defined procedures to provide the end-to-end packet delivery, including congestion control.

We chose the DFR [7] which was proposed in our previous work as a flooding-based routing protocol and we also

```
(1) MAX_ACK_seq is the maximum ACK sequence among previous received ACK segments;
(2) ACK_cnt is a local value;
if (ACK_seq < MAX_ACK_seq) then
   Discard an ACK segment;
end if
if (ACK_seq > MAX_ACK_seq) then
   ADC(t) ← α ∗ ADC(t − 1) + (1 − α) ∗ ACK_recv/Cwnd;
   Transmit a new DATA segment;
end if
ACK_cnt ← 0
if (ADC(t) ∗ 4 ≥ ACK_cnt)then
   Retransmit a lost DATA segment;
   ACK_cnt ← 0
else
   if (ADC(t) ∗ 4 < ACK_cnt) then
      Discard an ACK segment;
      ACK_cnt ← ACK_cnt + 1
   end if
end if
```

Algorithm 2: Operation of each source node (in FDC).

used a broadcasting mode of IEEE 802.11 MAC protocol. We positioned 250 sensor nodes at random locations in the square of 1500 m × 1000 m among which one sink was located in the center top of the square. To emulate high error rate in UWSNs, link error rates of 12% and 3% were set for DATA and ACK segment transmissions, respectively. Each sensor node was set to have a bandwidth of 10 kHz. The maximum transmission range of sensor nodes was set to 300 m. Among the sensor nodes which were located more than 4 hops away from the sink, source nodes were chosen randomly. They generated FTP traffic for 1800 seconds towards the sink. The DATA segment size of the FTP traffic was set to 64 Kbytes. α value of FDC was set to 0.5 (we found out the best α empirically).

6.2. Simulation Results. First, we investigate their average throughput with various numbers of FTP traffic flows (Figure 6). Regardless of the number of FTP traffic flows, TCP-Reno with FDS and TCP-Reno with FDC outperforms the original TCP-Reno in terms of higher throughput.

In the original TCP-Reno, since the source node performs unnecessary Fast Retransmit due to the simple reception of multiple ACK segments without the distinction, a lot of packet retransmissions cause the network resource to be wasted and particularly those retransmitted packets incur packet collisions in the network. In contrast, such throughput degradation can be avoided due to the capability of the ACK distinction in TCP-Reno with our Fast Retransmit techniques. Through this distinction, TCP-Reno with our Fast Retransmit techniques performed the Fast Retransmit only when necessary.

However, as the number of traffic flows increases, the network gets more congested and the number of lost ACK segments increases accordingly. Hence, the number of unnecessary Fast Retransmit misbehaved by source nodes also decreases, which makes their performance gap reduced.

Moreover, TCP-Reno with FDS outperforms TCP-Reno with FDC with higher throughput, regardless of the number

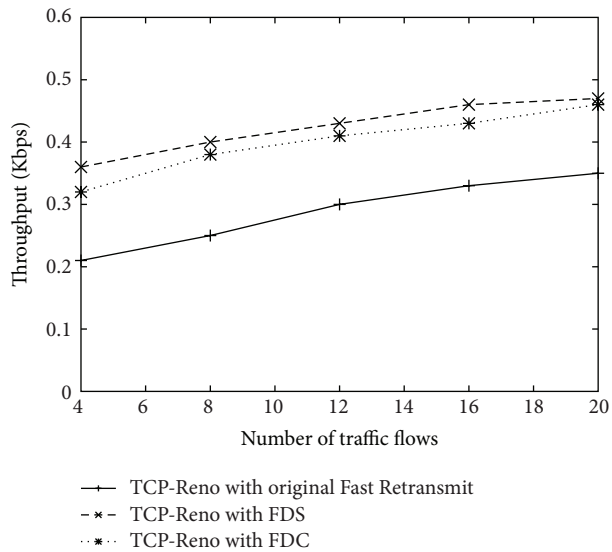

Figure 6: Throughput.

of traffic. In TCP-Reno with FDS, the source node checks whether to fast-retransmit the DATA segment according to DACK_cnt recorded into a header. Hence, in TCP-Reno with FDS, although some ACK segments are lost, the source node can perform Fast Retransmit correctly if DACK_cnt which is included in a received ACK segment is greater than 4. However, in TCP-Reno with FDC, since the source node determines whether to fast-retransmit the DATA segment according to the number of duplicated ACK segments, Fast Retransmit cannot be performed correctly when a lot of ACK segments are dropped due to network congestion.

Second, we compared retransmission overhead of the three protocols according to the various number of traffic flows (Figure 7). Retransmission overhead is defined as the ratio of the number of DATA segments retransmitted by a source node to the number of DATA segments received by

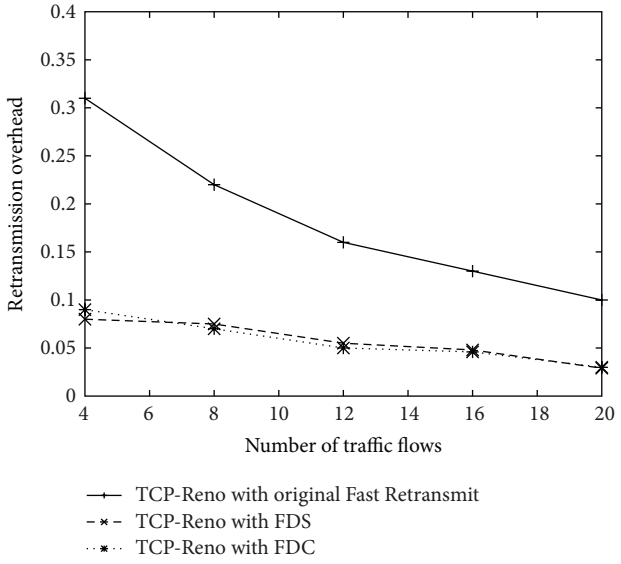

Figure 7: Retransmission overhead.

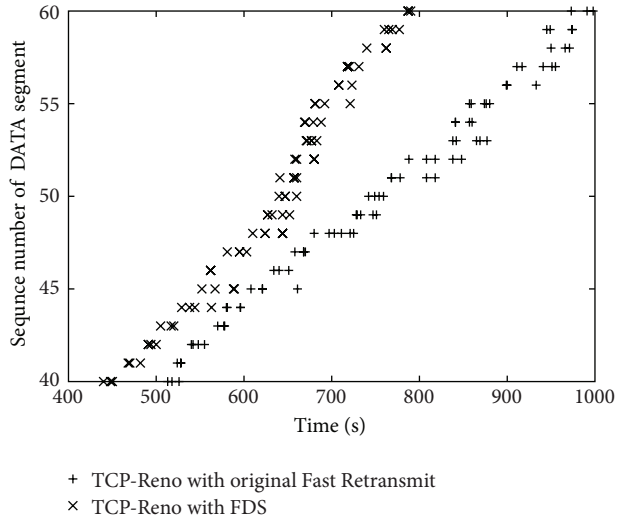

Figure 8: Data segment reception (FDS).

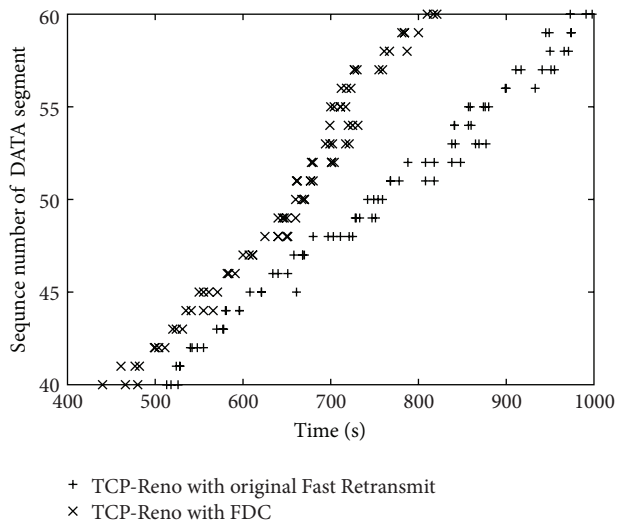

Figure 9: Data segment reception (FDC).

the sink. As observed in the comparison of throughput, the timely execution of the Fast Retransmit in TCP-Reno with our proposed Fast Retransmit techniques leads to less retransmission overhead.

However, as network is more congested, a source node depends on the larger number of timeout-based retransmissions for providing reliable packet delivery, rather than Fast Retransmit. Hence, the gap in their performance also becomes reduced with the number of traffic flows.

Moreover, retransmission overheads of TCP-Reno with FDS and TCP-Reno with FDC are similar regardless of the number of traffic. As mentioned in the previous simulation analysis, TCP-Reno with FDC cannot perform Fast Retransmit correctly when a lot of ACK segments are dropped. However, the lost DATA segments are retransmitted from the source node after RTO timeouts [18]. Therefore, in both of TCP-Reno with FDS and TCP-Reno with FDC, retransmission overheads are still observed as the number of lost DATA segments increases.

Finally, we measured the distribution of arrival time of DATA segments which have reached the sink for the three protocols. In these simulations, one flow was selected among eight flows in the network and we traced the 40th to 60th DATA segments of the flow. Due to the flooding-based routing protocol, it is possible that the sink node received multiple copies of a DATA segment within a short period, regardless of the two implemented Fast Retransmit techniques. However, in TCP-Reno with the original Fast Retransmit, the sink is still receiving a lot of the same DATA segments even after the time elapsed significantly (i.e., a lot of unnecessary retransmissions), unlike TCP-Reno with our proposed Fast Retransmit techniques.

7. Conclusion

In this paper, we attempted to exploit the Fast Retransmit well-defined in TCP for the purpose of supporting the reliable end-to-end packet delivery over flooding-based routing protocols in underwater wireless sensor networks. To apply the Fast Retransmit, we revealed the ACK indiscretion problem where each source node cannot distinguish between original duplicated ACK segments and multiple copies of an ACK segment received through flooding, leading to unnecessary retransmissions. To address the ACK indiscretion pro-blem, this paper proposed two Fast Retransmit techniques to be applied to flooding-based routing protocols, namely, FDS (Fast Retransmit technique based on duplicated ACK sequence number) and FDC (Fast Retransmit technique based on duplicated ACK count). In FDS, the sink includes a sequence number indicating the occurrence number of duplicated ACK segments into outgoing ACK segments. From the sequence number, each source node is required to execute the Fast Retransmit only when the sink has transmitted more than three duplicated ACK segments. Since FDS requires an

additional field in the header for the sequence number, we also proposed the second technique, FDC, which allows the source nodes to estimate the number of ACK copies which are expected to be received when the sink transmits an ACK segment, without any dependency on the additional field. In FDC, each source node performs Fast Retransmit only when the number of received ACK segments received at the source node is over four times than the estimated number of ACK segments.

Through NS-2 simulations, we observed that our first proposed Fast Retransmit technique, FDS, achieves 15–60% throughput improvements and 220–286% reduction of retransmission overhead. Also, our second Fast Retransmit technique, FDC, could achieve performance improvements of 13–57% and 218–279% in terms of higher throughput and lower retransmission overhead, respectively. Through traces of the transmitted DATA segments, it was proved that the original Fast Retransmit causes a lot of unnecessary retransmissions.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under Contract no. UD130007DD. This work was supported by the IT R&D Program of MSIP/KEIT (10041145, Self-Organizing Software Platform (SoSp) for Welfare Devices).

References

[1] I. F. Akyildiz, D. Pompili, and T. Melodia, "Challenges for efficient communication in underwater acoustic sensor networks," *ACM SIGBED Review*, vol. 1, 2004.

[2] E. M. Sozer, M. Stojanovic, and J. G. Proakis, "Underwater acoustic networks," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 72–83, 2000.

[3] Y. Bayrakdar, N. Meratnia, and A. Kantarci, "A comparative view of routing protocols for underwater wireless sensor networks," in *Proceedings of the MTS/IEEE OCEANS*, pp. 1–5, June 2011.

[4] R. Otnes and S. Haavik, "Duplicate reduction with adaptive backoff for a flooding-based underwater network protocol," in *Proceedings of the MTS/IEEE OCEANS*, pp. 1–6, June 2013.

[5] I. Chakeres and C. Perkins ", "Dynamic MANET ondemand (DYMO) routing," Internet Draft (Draft-Ietfmanet-Dymo-14), 2008.

[6] H. Yan, Z. Shi, and J. H. Cui, "DBR: depth-based routing for underwater sensor networks," in *Proceedings of the IFIP Networking*, pp. 16–1221, May 2008.

[7] D. Hwang and D. Kim, "DFR: directional flooding-based routing protocol for underwater sensor networks," in *Proceedings of the MTS/IEEE OCEANS*, pp. 1–7, September 2008.

[8] P. Xie, C. J. H. Xie, and L. L. Xie, "VBF: vector-based forwarding protocol for underwater sensor networks," in *Proceedings of the IFIP Networking*, pp. 1216–1221, May 2006.

[9] M. Ayaz and A. Abdullah, "Hop-by-hop dynamic addressing based (H2-DAB) routing protocol for underwater wireless sensor networks," in *Proceedings of the International Conference on Information and Multimedia Technology (ICIMT '09)*, pp. 436–441, December 2009.

[10] X. Xiao, X. PengJi, G. Yang, and Y. Ping Cong, "LE-VBF: lifetime-extended vector-based forwarding routing," in *Proceedings of the International Conference on Computer Science & Service System (CSSS '12)*, pp. 1201–1203, August 2012.

[11] H. Chin, P. Ramanathan, and K. Saluja, "Routing TCP flows in underwater mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 10, pp. 2022–2032, 2011.

[12] J. Postel, "Transmission control protocol," RFC 793, 1981.

[13] L. S. Brakmo, S. W. OMalley, and L. L. Peterson, "TCP vegas: new techniques for congestion detection and avoidance," in *Proceedings of the conference on Communications architectures, protocols and applications (ACMSIGCOMM '94)*, pp. 24–35, October 1994.

[14] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit," RFC 2001, 1997.

[15] P. Xie and J.-H. Cui, "An FEC-based reliable data transport protocol for underwater sensor networks," in *Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN '07)*, pp. 747–753, August 2007.

[16] X. Cheng, H. Shu, Q. Liang, and D. H.-C. Du, "Silent positioning in underwater acoustic sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 3, pp. 1756–1766, 2008.

[17] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom '03)*, pp. 134–146, September 2003.

[18] V. Paxson and M. Allman, "Computing TCP's retransmission timer," RFC 2988, 2000.

[19] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, pp. 314–329.

[20] P. Sarolahti, "Congestion control on spurious TCP retransmission timeouts," in *IEEE Global Telecommunications Conference (GLOBECOM 03)*, vol. 2, pp. 682–686, December 2003.