On the Use of Conversation Detection to Improve the Security of Underwater Acoustic Networks

Alberto Signori[‡], Filippo Campagnaro[‡], Kim-Fabian Wachlin[♯], Ivor Nissen[*♯], Michele Zorzi[‡]

[‡] Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy
and Wireless and More, s.r.l., via della Croce Rossa 112, 35129 Padova, Italy

[♯] Department of Computer Science, Kiel University, Hermann-Rodewald-Straße 3, 24118 Kiel, Germany,

[*] Bundeswehr Technical Center for Ships and Naval Weapons Maritime Technology and Research,
Klausdorfer Weg 2–24, 24118 Kiel, Germany,

[‡]{signoria,campagn1,zorzi}@dei.unipd.it,
[♯]kfw@informatik.uni-kiel.de,
[*]ivornissen@bundeswehr.org

***Abstract*—Security is one of the key aspects of underwater acoustic networks, due to the critical importance of the scenarios in which these networks can be employed. For example, attacks performed to military underwater networks or to assets deployed for tsunami prevention can lead to disastrous consequences. Nevertheless, countermeasures to possible network attacks have not been widely investigated so far. One way to identify possible attackers is by using reputation, where a node gains trust each time it exhibits a good behavior, and loses trust each time it behaves in a suspicious way. The first step for analyzing if a node is behaving in a good way is to inspect the network traffic, by detecting all conversations. This paper proposes both centralized and decentralized algorithms for performing this operation, either from the network or from the node perspective. While the former can be applied only in post processing, the latter can also be used in real time by each node, and so can be used for creating the trust value. To evaluate the algorithms, we used real experimental data acquired during the EDA RACUN project (Robust Underwater Communication in Underwater Networks).**

I. Introduction and Related Works

Security aspects in underwater wireless networks have not been widely investigated so far, despite the critical importance of the scenarios in which these networks can be employed [1], [2]. For example, an attack to a military underwater network for enemy targeting or identification can lead to serious consequences. Similarly, Mine Countermeasure (MCM) missions performed by Autonomous Underwater Vehicles (AUVs) coordinated through an acoustic network are also critical from a public safety point of view. In addition, underwater networks can be used for environmental monitoring, such as monitoring for tsunami risk mitigation [3], where an attack and a consequent network failure could become dangerous for human life.

One of the possible ways to identify a malicious node that tries to perform a Denial Of Service (DoS) attack, such as jamming [4], [5] or resource exhaustion [6], is by analyzing the traffic in the network, to isolate nodes with a suspicious behavior. To isolate a node that deviates from a natural behavior, i.e., from following the protocol rules, a trustworthiness index can be used, based on the reputation of the node [7], [8]. Ideally, each action that deviates from the protocol rules causes a decrease in the node's reputation, and conversely the reputation of a node acting as expected should increase. In this work, we define reputation as the opinion that the nodes of the network have about a specific node, based on its past behavior, and we define trust as the belief that a node is safe and reliable, i.e., it behaves according to the network protocol rules, without trying to attack the network. In terrestrial networks the basic element of each trust mechanism is the so called watchdog system [9], i.e., the capability of a node to overhear packets transmitted by its neighbors by exploiting the broadcast nature of the channel. The information carried by the overheard packets can then be used to understand whether or not a node is acting properly. In the underwater environment this operation becomes more difficult due to high variability of the channel conditions [10], that can lead the channel to be in a bad state for a long time, making it more difficult to detect packets from neighbors.

In order to detect whether a node is behaving according to the network protocols, it is essential to first identify all the conversations in the network, and then classify the traffic types of those conversations, by observing the packets transmitted in the network. In this paper we refer to a conversation as a communication exchange between two or more nodes with the aim of conveying the data of a certain application. In this view, two nodes can establish multiple conversations with each other simultaneously. For instance, if node A requires a status update from node B while node B is transmitting to A the value acquired with a temperature sensor every T_1 seconds, and A is transmitting to node B data acquired by a sonar every T_2 seconds, we account for three separate conversations, namely a request-reply (REQ/REP) and two periodic applications with period T_1 and T_2, respectively.

Conversations identification and traffic classification operations can be performed either with a global networking view (centralized solution), by observing the packets transmitted by

This work has been supported by the Bundeswehr Technical Center for Ships and Naval Wepons, Maritime Technology and Reseach, GF 630 Underwater Communication, contract 728053 - E/E71S/K1291CF081

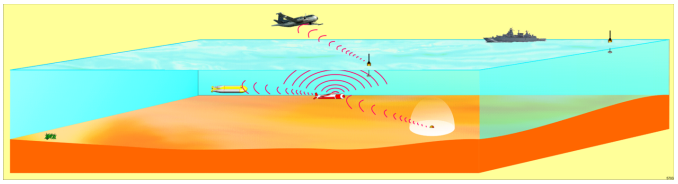

Fig. 1. Typical underwater network scenario composed by static nodes, an underwater vehicle and a surface vehicle. Nodes communicate with acoustic waves

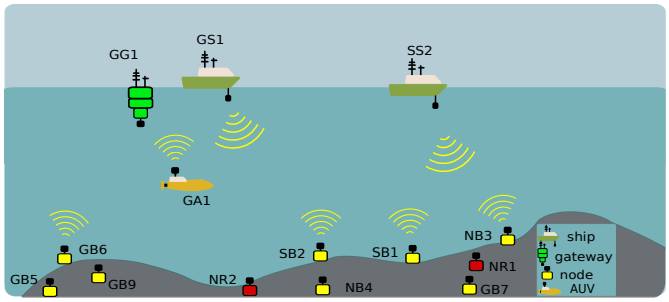

Fig. 2. The network tested during the RACUN project. The test has been performed in near La Spezia, in May 2014.

all nodes, or distributedly, where each node tries to identify the conversations on its own, using only the locally observed packets. While the former approach leads to better results, the latter is more realistic to be implemented in a real-time application, because it does not require full knowledge of all packets transmitted by each node. Still the former solution is of interest, both as a benchmark for the latter approach and to best identify any misbehavior in post processing.

In this paper we analyze and compare both centralized and distributed solutions for conversation identification and traffic analysis in an underwater acoustic network scenario with both mobile and static nodes (Figure 1), and discuss how to apply them in a reputation system used to identify possible attackers in an underwater network. In addition, we assess the accuracy of the conversation detection in a decentralized way when it is performed in a real-time scenario. It is important to understand how accurate the detection is in real time, when some of the packets belonging to the same conversation may not yet have been received. Indeed, a sufficiently accurate detection should allow the assessment of the reputation of the nodes during the mission, and not only at the end of it.

These solutions will be evaluated through a real field dataset acquired during the RACUN project [11], [12]. This dataset has been obtained through the use of the DESERT Underwater Framework [13] integrated with the Gossiping in Underwater Acoustic Mobile Ad-hoc Networks (GUWMANET) protocol and the Generic Underwater Application Language (GUWAL) application layer [14].

This paper is organized as follows. Section II describes the network deployment scenario and the RACUN dataset. Section III presents both the centralized and the distributed conversations detection algorithms, and discusses the use of the latter for a real-time conversation detection. Section IV presents the evaluation of the distributed algorithms compared to the centralized ones, and, finally, Section V draws concluding remarks.

II. Scenario and Protocol Description

The underwater network analyzed in this paper is depicted in Figure 2: the deployment is composed by two ships, one AUV, one gateway buoy, 8 bottom nodes (depicted in yellow) and two bottom receivers (depicted in red). The distance between two adjacent nodes ranges from 600 m to 2.5 km. All nodes were operating in the RACUN frequency band of 4-8 kHz. The maximum range experienced during the sea trial was 4 km. The ship FS Planet (GS1), the gateway buoy (GG1), the AUV (GA1) and four bottom nodes (GB5, GB6, GB7 and GB9) were provided by Germany, two bottom nodes (SB1 and SB2), as well as the network node deployed from the Italian ship Tavolara (SS2), were provided by Sweden, while the remaining two bottom nodes (NB3 and NB4) and the two bottom receivers (NR1 and NR2) were provided by Norway. The ships, the gateway buoy, the German bottom nodes and the AUV were equipped with Develogic acoustic modems, the Swedish nodes with SAAB modems and, finally, the Norwegian nodes were equipped with NILUS units. The experiment was launched directly with GUWMANET in the German bottom nodes, and with the DESERT Underwater framework [13] in all other nodes. In this scenario, all nodes employed the GUWAL application layer and the GUWMANET protocol. More details on the deployment and on the nodes setup can be found in [15].

A. GUWAL and GUWMANET

GUWAL is an application layer specifically designed for underwater networks [14]: it defines four packet types that can be sent through the acoustic nodes. Specifically, the nodes can send a Data Request (DataReq) and wait for receiving Data packets. In addition, GUWAL also supports the transmission of Command and Control (CMD) packets and simple String Text Message (STR) packets. Moreover, the specific message type carried by the packet can be specified inside the packet, such as the type of data sent or requested (e.g., status or position data) or the specific command sent through the network (e.g., move or power control commands). Data packets and STR messages can be sent either upon request or periodically. GUWAL allows to require an acknowledgement (ACK) for the transmitted packets throughout a flag set in the application packet header. The ACK is sent by the destination node in the form of a CMD packet type.

Since GUWAL and GUWMANET are designed to work together with a cross layer approach to reduce the packet header overhead, GUWAL header contains the source and the destination addresses. Each address is 6 bits long, and is divided in two parts: the former is 2 bits long and defines the node type (four different types can be defined), the latter is 4 bits long and defines the node inside the group of nodes of the same type. One address is reserved for broadcast, and

TABLE I
NODES AVAILABLE FOR EACH PART OF THE DATASET

	Subset of nodes
Part I	{NB3, NB4, GB6, GB9, GG1, GS1, GA1}
Part II	{SB1, SB2, NB3, NB4, GB6, GB9, GG1, GS1, SS2, GA1}
Part III	{SB2, NB3, NB4, GB5, GB8, GB9, GS1, SS2, GA1}
Part IV	{SB1, SB2, NB3, NB4, GB5, GB8, GB9, GS1, SS2}

TABLE II
SKETCH OF THE RACUN DATASET

TX\RX	GB7	NB3	SS2	SB1	...
t_1::PCK$_1$	sender	$t1_{NB3}$::RX	not rx	$t1_{SB1}$::RX	...
t_2::PCK$_2$	not rx	sender	rx()	$t2_{SB1}$::RX	...
t_3::PCK$_3$	sender	$t3_{NB3}$::RX	$t3_{SS2}$::RX	not rx	...
t_4::PCK$_4$	$t4_{GB7}$::RX	$t4_{NB3}$::RX	$t4_{SS2}$::RX	sender	...
t_5::PCK$_5$	$t5_{GB7}$::RX	not rx	sender	$t5_{SB1}$::RX	...
...	...	...	...	...	...

four addresses are reserved for multicast, to send packets to all the nodes of the same type.

GUWMANET uses a contention Medium Access Control (MAC) protocol, based on random access. Specifically, each packet is transmitted after a random backoff time and no carrier sensing methods are used. To increase robustness, each packet is automatically repeated after a random time, until either a given number of retransmissions is reached or an implicit ACK is received, as the node can overhear the packets transmitted by the next hop. The number of retransmissions depends on the priority of the transmitted packets, the higher the priority, the higher the number of retransmissions. GUWMANET is also in charge of performing packet routing: at the beginning, a simple flooding mechanism is used to forward packets, then nodes learn routes from packet transmissions and establish a temporary route using information inserted in the packets. This temporary route will be used to directly forward the packets. More details about GUWMANET can be found in [14].

B. Dataset Description

The dataset collected during the RACUN project is divided in four parts, corresponding to four different days of the test described in [15], and each part lasts around 90 minutes. For each part, a different subset of all the available nodes is used, and conversations are detected independently for each one. The subsets of nodes used for each part are reported in Table I.

In the dataset each entry corresponds either to the packet generation time, i.e., the time instant in which the application of the source node generates the packets, or to the packet reception time, i.e., the time instant in which a node correctly received a packet. For each entry the whole packet is stored, therefore information such as source and destination addresses, the GUWAL packet type and, more in general, all the data that the GUWAL and GUWMANET protocols store in the packet can be retrieved.

The entries related to the packet generation times are the only ones used to perform the conversation analysis from a network level perspective, while all entries stored by a node (i.e., all the packets that a node is able to listen to) are used for the conversation analysis from the node level perspective. We want to point out that using only the packets received by a node to detect the conversations poses a further challenge in the conversation detection, since some packets could not be detected by a node due to channel errors, and the timestamp of the received packet accounts for random delays introduced by the protocol stack and the propagation time, making it more difficult to detect periodic conversations.

In addition, this dataset can be used to detect conversation in real time, by analyzing the packets as soon as they are received by a node. Specifically, each time a node receives a packet, the algorithms updates the conversation assignment of all packets received, improving the conversation detection accuracy.

III. CONVERSATIONS IDENTIFICATION ALGORITHMS

The goal of this paper is to describe the Kieler algorithm that identifies the different conversations in the network.

A conversation is defined as a communication exchange between two or more nodes with the aim of conveying the data of a certain application, e.g., if during the sea trial two nodes are exchanging periodic status data, periodic ranging information, and a REQ/REP data, three separated conversations are accounted, i.e., a REQ/REP and two periodic conversations.

To detect the conversation we propose a unique algorithm that can be applied both in a centralized way, i.e., analyzing all packets generated by all the nodes of the network from a global network perspective, and in a distributed way, i.e., analyzing only the packets received by a node. While the former can be performed only in post processing accessing all nodes to extract the transmission logs, the latter can also be applied in real time directly from each node. Although less practical, the centralized version can still be applied in post processing, and provides an important benchmark to the distributed algorithm, as the latter, having only a partial view of the complete network, cannot outperform the former.

The conversation detection algorithms considered in this paper analyze the packets by checking their packet type, source and destination address, time stamp and the GUWAL packet checksum to identify the conversations. Specifically:

- the message type is used to identify DataReq and Data reply pairs or CMDs;
- the source address and the destination address represent the nodes involved in the conversation, including broadcasts and multicasts addresses;
- the timestamp is used to check the chronological order of the messages (e.g., a DataReq should occur before a Data reply), for the recognition of periodic conversations, and to check whether a reply was sent within a predefined time period;
- the checksum is used by GUWAL to reference previous messages of a conversation, a CMD ACK for example takes up the checksum of the CMD again.

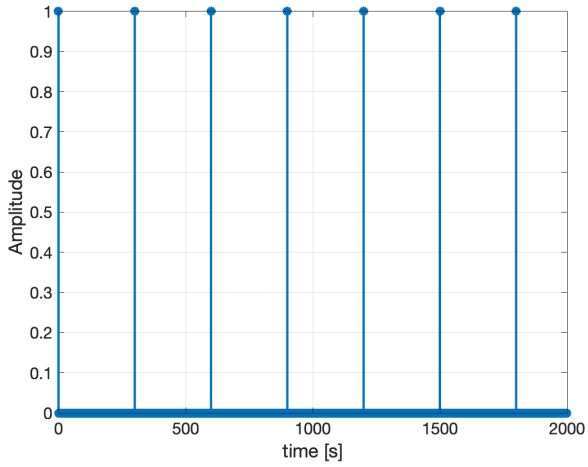

Fig. 3. Example of signal obtained from periodic packets with period equal to 300 s.

Not all packets in the same conversation are always directly related to each other. This is the case if the conversation is performed between multiple nodes: for instance, two replies sent by two different nodes to a multicast request are part of the same conversation,

The first algorithm used to detect conversations, called method 1, works in both centralized and decentralized ways. It performs two steps: *(i)* detection of period conversation, and *(ii)* detection of the remaining (non-periodic) conversations.

The first phase consists in treating the packet arrival time as a signal and trying to detect the period of this signal using Fast Fourier Transform (FFT). More in detail, each packet arrival is considered as a pulse in the signal, therefore the signal given as input to the FFT is 0 everywhere except for the instant when the packet is received or generated, where the signal will be equal to 1. An example of this signal is given in Figure 3.

Since we do not known a priori which packets are periodic and which are not, we first create a list with all packets that are possible candidates to be periodic ones, then we create the signal as discussed above, and use it as input for the FFT. The list of candidate periodic packets is filled based on the source and destination addresses and on the GUWAL packet type. Specifically, all the packets with the same source and destination and with the same packet type are considered to be candidates for the periodic list. Once the FFT is performed, we search for a possible candidate period looking at the frequency response. As last step, the candidate period is used to select, among the list of possible periodic packets, those that fit with the found period. In particular, we look if two packets are at a distance equal to the found period, within a given tolerance that can be differently set for the centralized and the decentralized method.

The second phase consists in the detection of non-periodic conversations. The non-periodic conversations mostly consist of DataReq and Data reply, or of CMD. For each DataReq, we look for subsequent Data packets, selecting those for which the message type and the addresses match. In particular, considering a node sending a request packet P with destination address $dest_P$ we look for all the Data replies which source address matches $dest_P$. Specifically, if $dest_P$ is the broadcast address all the source addresses are considered, if $dest_P$ is a multicast address[1] all the source addresses belonging to the corresponding node type are analyzed, if the address is UNICAST, only that specific address is considered. Similarly, for CMD we search for subsequent packets matching the transmitted command. If the command requires to be acknowledged by the receiver, we look for a subsequent ACK related to the transmitted command.

The same algorithm is used also to perform a real-time conversation detection. Indeed, it is important to assess the performance in a real-time scenario, when at a given point some of the packets belonging to a single conversation may not yet have been received, and therefore the conversation could not be correctly detected with only this partial information. As an example, considering a periodic conversation, at least three packets need to be received before conversations could be identified as periodic: until the third packet is received, the first two will therefore be assigned to one or more wrong conversations. Since the final analysis on the trustworthiness of a node could be performed in a real-time scenario during a mission, it is important to understand how the algorithm behaves in such a scenario.

Another centralized algorithm, named centralized method 2, has been analyzed. The structure of the algorithm is divided into two parts. In the first part the algorithm iterates over all packets and focuses on the first unchecked DataReq, CMD or STR packet. This packet gets a newly created incremental conversation number X and is removed from the list of packets to be checked. In the second iteration, the algorithm compares all remaining packets with the first one, and assigns to them the same conversation identifier when the following constraints are satisfied:

- The packet type must fit, e.g., a DataReply should match a DataReq.
- The address space is consistent, e.g., the destination address of a DataReply should be the source address of a DataReq.
- The amount of time elapsed between two packets of the same conversation is lower than a predefined threshold, e.g., a DataReply should have been sent within a few minutes after a DataReq has been issued.

We underline that two replies sent by two different nodes to the same multicast request are part of the same conversation, even if the two replying nodes do not directly exchange packets between each others.

The second part of the algorithm detects periodic conversations. Since a periodic conversation exists when the same type of packet is sent at approximately the same time interval, called period, the period must be determined for

[1]In GUWAL and GUWMANET a packet can be sent in MULTICAST to all the nodes of the same type, i.e., to all bottom nodes, to all gateway buoys, or to all mobile nodes.

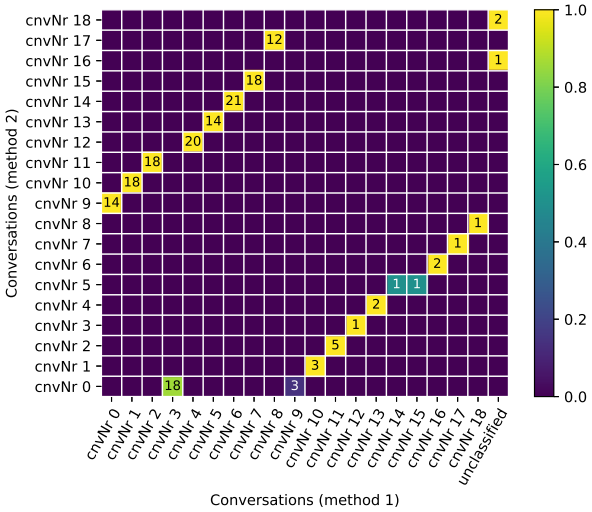

Fig. 4. Comparison of two centralized methods for conversation detection for Part I of the dataset.

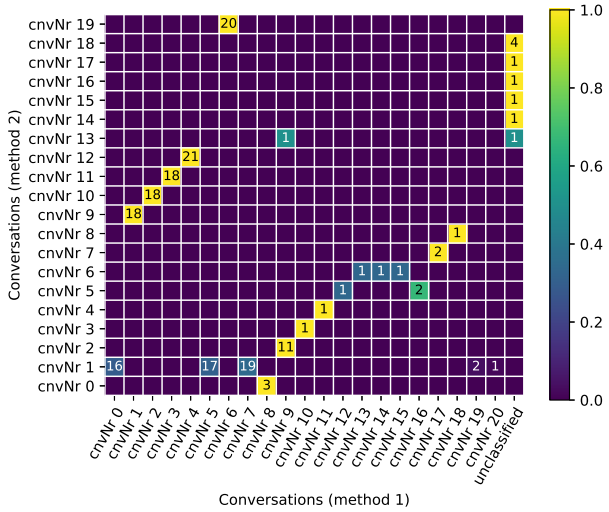

Fig. 5. Comparison of two centralized methods for conversation detection for Part III of the dataset.

the conversations identification. The period is complicated to obtain, especially in the case periodic DataReqs are issued along with other non-periodic DataReqs: in this case a periodic interval is very hard to recognize in the time domain. Therefore, all potentially non-periodic packets (e.g., the CMDs) are removed from the list of packets to be checked, to then proceed with the identification of the periodic conversations. To perform this operation, the algorithm iterates over all the packets again. For an unchecked packet, a new conversation number is created in the same way as in the first part. This packet is now being checked for similarities (i.e., packet type, source and destination address) with other packets that were also unchecked, and all packets where those similarities are matched and that have been transmitted with the same interval are assigned to the same conversation. The period check is based on a previously defined tolerance. Finally, all packets already assigned to non-periodic conversations are checked for this periodic conversation and, if necessary, transferred by changing the conversation number.

IV. Results

In this section we show the results obtained for the conversation detection, firstly comparing the two algorithms described in Section III, then comparing the centralized results with the decentralized ones, and at the end analyzing the performance in a real-time scenario. For the centralized and decentralized comparison and for the real-time analysis only the first algorithm is used.

A. Centralized algorithms comparison

Figures 4 and 5 show the comparison between the two algorithms used for detecting conversations in the centralized scenario, i.e., from a network level perspective. In particular, it shows the results obtained with the two algorithms running on Part I (Figure 4) and Part III (Figure 5) of the dataset. These figures show the fraction of packets that are in common between two conversations found by the two algorithms (yellow squares mean that all the packets assigned to a conversation with method 2 are in common with the corresponding conversation obtained with method 1, purple squares mean that the two corresponding conversations have no packets in common). If the number of packets in common is greater than zero, the actual number of packets that are in common between the two conversations is reported.

B. Centralized vs decentralized results

Figure 6 and Figure 7 compare the results obtained with the decentralized method with those obtained with the centralized method 1, from the perspective of two different nodes. In particular, Figure 6 represents the conversations obtained from the point of view of the node deployed from the Swedish ship SS2, compared with the conversation obtained with a network level perspective for Part IV of the dataset. Similarly Figure 7 shows the results from the perspective of node GS1 when compared with the centralized solution of the same algorithm.

Both figures show that, as expected, an important aspect of the decentralized conversation detection are packets not received by a node due to channel errors. Nevertheless, the algorithm is still able to properly group the received packets in the correct conversation. In Figure 6 most of the packets assigned to a single conversation in the centralized solution are grouped in a single conversation in the decentralized solution, as well. In this case, only conversation number 4 (ConvNr 4) of the centralized solution is split in more than one conversation in the decentralized solution, specifically in conversations number 1, 13, 14. In this specific case, ConvNr 4 is a periodic conversation and the analysis from a network level perspective lead to an easy identification of the period. Indeed, from the node level perspective random delays due

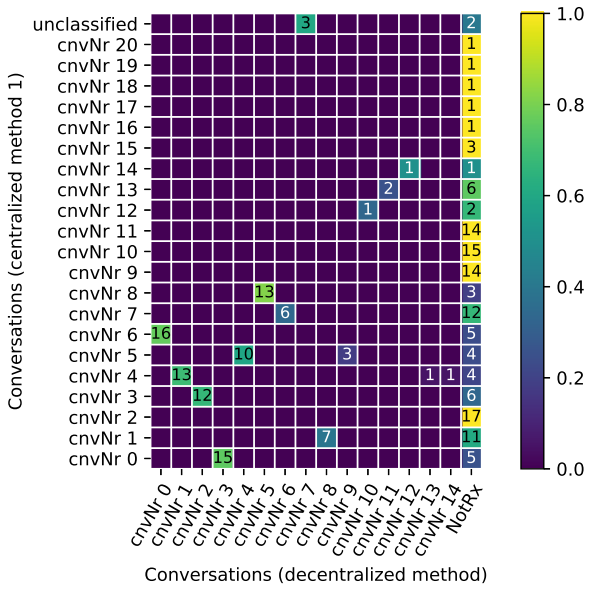

Fig. 6. Comparison between centralized and decentralized solutions obtained from node SS2 for Part IV of the dataset

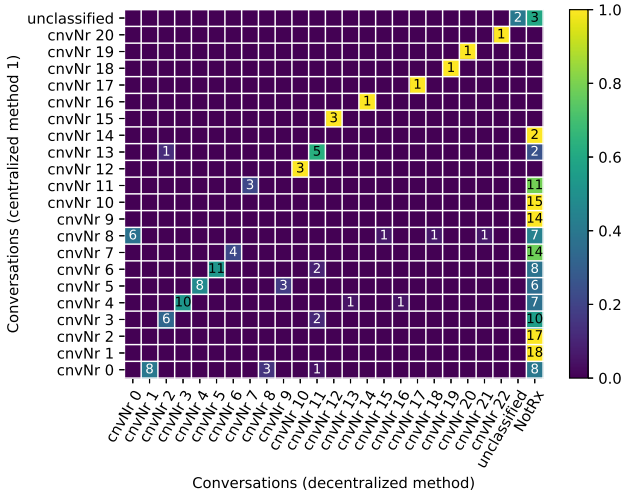

Fig. 7. Comparison between centralized and decentralized solutions obtained from node GS1 for Part IV of the dataset

to the channel propagation and to the protocol stack make it more difficult to detect periodic packets, inducing the wrong detection of two packets (assigned to conversations 13 and 14, instead of conversation 1). Figure 7 confirms that most packets are correctly grouped together, however in this case 6 original conversations are split in more than one in the decentralized solution.

C. Real-time results

In the last step we considered also the performance of the algorithm used to detect conversations in a real-time scenario. We considered only the decentralized method, since it is the only one that can be applied in such case. Indeed, the centralized solution can only be used in post-processing. For

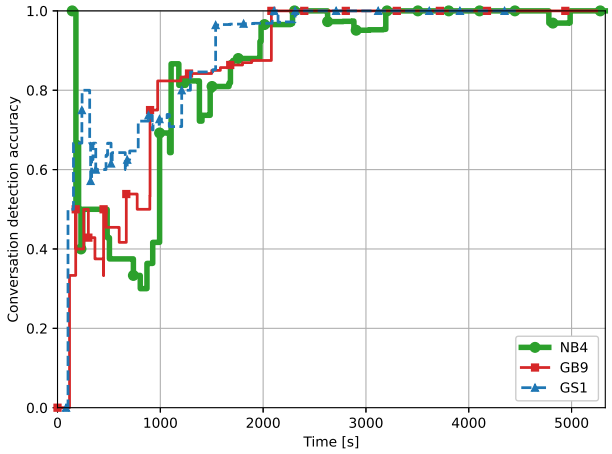

Fig. 8. Real-time performance for Part I of the dataset

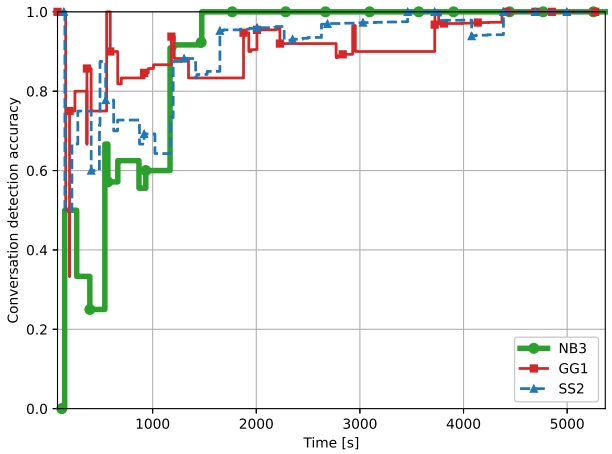

Fig. 9. Real-time performance for Part II of the dataset

each of the four parts of the dataset we show the accuracy of the detection using as a benchmark the conversation detection obtained when all the packets are considered. The accuracy (A), at a given time T, is computed with respect to the number of packets received until T ($N_{rx}(T)$), and not to the whole part of the dataset. To compute the conversation accuracy we match together the real-time conversation and the benchmark conversation that have the maximum number of packets in common with respect to all the other possible combinations between benchmark and real-time conversations. Then the common packets are marked as correct $C(T)$, while the remaining packets in the analyzed real-time conversation (if any) are marked as wrong. Each conversation can not be matched more than one time, and all the remaining packets of the unmatched conversations are marked as wrong. The accuracy is computed as the ratio between the correct packets and all the packets received until time T.

$$A = \frac{C(T)}{N_{rx}(T)} \tag{1}$$

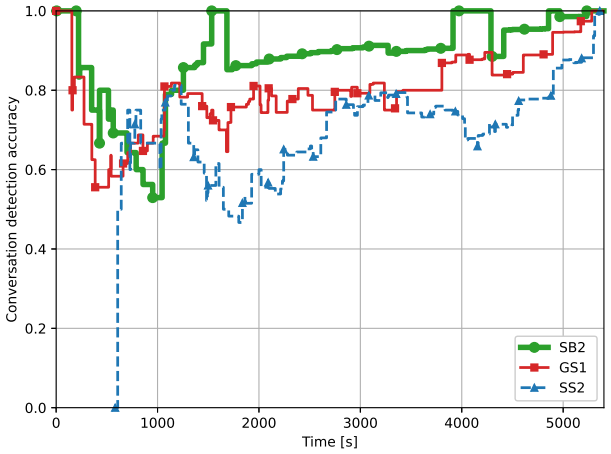

Fig. 10. Real-time performance for Part III of the dataset

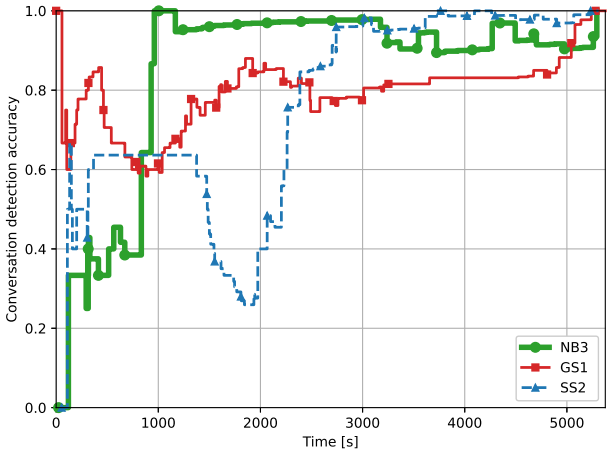

Fig. 11. Real-time performance for Part IV of the dataset

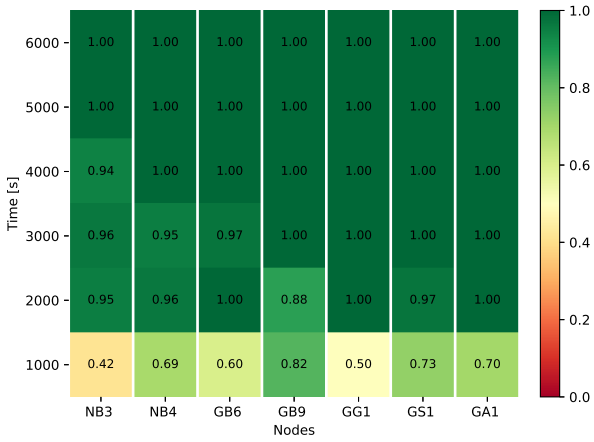

Fig. 12. Real-time performance for Part I of the dataset for all the nodes

Fig. 13. Real-time performance for Part II of the dataset for all the nodes

Figures 8, 9, 10 and 11 show the accuracy evolution, by adding a new incoming packet each time, taking into account 3 nodes for each part of the dataset.

Figure 8 shows that in this scenario after around 1000 s from the transmission of the first packet, the accuracy stays above 0.8 for the 3 nodes almost all the time, and increases up to 0.9 after 2000 s. A similar behavior is reported in Figure 9 for Part II of the dataset

As we can observe from these figures, the behavior is not strictly monotonic as a function of time. At first, adding new packets to the available set can decrease the accuracy of the conversation detection. Indeed, as an example, when a new periodic conversation starts, it takes a while to be recognized as a unique and periodic conversation (as previously discussed in Section III, at least three packets are needed to be able to detect a period, but sometimes more packets are needed). Before the new periodic conversation is detected, the packets can be assigned to wrong conversations leading to a decrease of the accuracy. This behavior can be noticed in Figure 10 and specifically for node SB2, where the accuracy rapidly decreases from 0 to 1000 s because many periodic conversations start in this interval. Once enough packets for each periodic conversation are observed the accuracy increases, as happen from 1000 to 1500 s. A similar evolution can be observed also for node SS2.

In Figure 11 the same behavior is observed for node SS2 from 1500 to 2500 seconds. As before, the beginning of new periodic conversations leads to a decrease in the accuracy which is rapidly recovered once enough packets have been observed.

Figures 12, 13, 14 and 15 show the accuracy for all the nodes available in each part of the dataset. For readability of the figures, the accuracy is reported only for some time instants. In Figures 12 and 13 all the nodes have an accuracy higher than or equal to 0.88 for times bigger than or equal to 2000 s. Different from previous figures, this representation does not keep the insight of the variability due to the beginning of new periodic conversations, but gives an idea of the accuracy considering larger time intervals, such as those that can be used when conversation detection is applied to security in underwater networks.

Figures 14 and 15 depict the accuracy for the nodes in Parts III and IV of the dataset. In this case some nodes take more time to reach a higher level of accuracy, but still all the nodes,

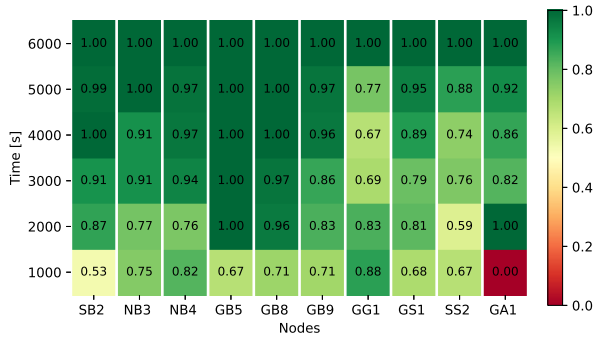

Fig. 14. Real-time performance for Part III of the dataset for all the nodes

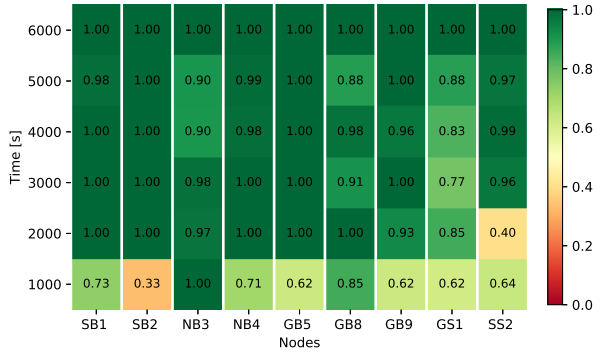

Fig. 15. Real-time performance for Part IV of the dataset for all the nodes

except for node GG1 in Part III of the dataset, have an accuracy higher than or equal to 0.77 for times bigger than or equal to 3000 s.

V. Conclusion and Outlook

In this paper we addressed the problem of conversation detection, proposing an algorithm to detect conversations in both a centralized and a decentralized way. While the former can be used only in post-processing when all the packets generated by the nodes are available, the latter can be used to detect conversations in a real-time scenario. The real-time detection can then be used to compute the reputation of a node based on its behavior and, consequently, give to each node a trustworthiness index, with the goal to detect and possibly isolate misbehaving nodes.

As expected, we observed that decentralized conversation detection is affected by packets losses due to bad channel conditions and by random delays introduced by the protocol stack and the propagation time, making it more difficult to detect conversations, especially periodic ones. Nevertheless, in most cases the algorithm used for conversation detection was found to be accurate and able to group packets in the same conversation as in the centralized version. In addition, the real-time detection turns out to be able to rapidly detect conversations, even periodic ones, promising to be suitable for the computation of the trustworthiness index. For future work, we will build a trust model able to isolate misbehaving nodes, exploiting the conversation detection mechanism presented in this paper.

References

[1] C. Lal, R. Petroccia, M. Conti, and J. Alves, "Secure underwater acoustic networks: Current and future research directions," in *Proc. UComms*, Aug. 2016.

[2] G. Yang, L. Dai, and Z. Wei, "Challenges, threats, security issues and new trends of underwater wireless sensor networks," *Sensors*, vol. 18, no. 11, p. 3907, Nov. 2018.

[3] P. Kumar, P. Kumar, P. Priyadarshini *et al.*, "Underwater acoustic sensor network for early warning generation," in *Proc. MTS/IEEE Oceans*. Hampton Roads, VA, USA: IEEE, Oct. 2012, pp. 1–6.

[4] A. Signori, F. Chiariotti, F. Campagnaro, and M. Zorzi, "A game-theoretic and experimental analysis of energy-depleting underwater jamming attacks," *IEEE Internet of Things Journal*, March 2020, Early Access.

[5] L. Ma, C. Fan, W. Sun, and G. Qiao, "Comparison of jamming methods for underwater acoustic DSSS communication systems," in *IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Mar. 2018.

[6] V. Shakhov and I. Koo, "Depletion-of-battery attack: Specificity, modelling and analysis," *Sensors*, vol. 18, no. 6, pp. 1849–1869, Jun. 2018.

[7] G. Han, J. Jiang, L. Shu, and M. Guizani, "An attack-resistant trust model based on multidimensional trust metrics in underwater acoustic sensor network," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2447–2459, Dec. 2015.

[8] W. Fang, C. Zhang, Z. Shi, Q. Zhao, and L. Shan, "BTRES: Beta-based trust and reputation evaluation system for wireless sensor networks," *Journal of Network and Computer Applications*, vol. 59, pp. 88–94, Jan. 2016.

[9] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 255–265.

[10] B. Tomasi, P. Casari, L. Finesso, G. Zappa, K. McCoy, and M. Zorzi, "On modeling janus packet errors over a shallow water acoustic channel using markov and hidden markov models," in *MILCOM Military Communications Conference*. IEEE, 2010, pp. 2406–2411.

[11] J. Kalwa, "The RACUN project: Robust acoustic communications in underwater networks - an overview," in *IEEE OCEANS*, Santander, Spain, Jun. 2011.

[12] C. Tapparello, P. Casari, G. Toso, I. Calabrese, R. Otnes, P. van Walree, M. Goetz, I. Nissen, and M. Zorzi, "Performance evaluation of forwarding protocols for the RACUN network," in *Proc. ACM WUWNet*, Kaohsiung, Taiwan, Nov. 2013.

[13] F. Campagnaro *et al.*, "The DESERT underwater framework v2: Improved capabilities and extension tools," in *Proc. UComms*, Lerici, Italy, Sep. 2016.

[14] M. Goetz and I. Nissen, "GUWMANET - multicast routing in underwater acoustic networks," in *Proc. MCC*, Warsaw, Poland, Oct. 2012.

[15] M. Goetz, I. Nissen, R. Otnes, and P. van Walree, "Performance analysis of underwater network protocols within international sea trial," in *Proc. MTS/IEEE OCEANS*, Genova, Italy, May 2015.