SCIENTIFIC JOURNAL OF POLISH NAVAL ACADEMY

2019 (LX) 3 (218)

sciendo

DOI: 10.2478/sjpna-2019-0021

EMULATOR AND SIMULATOR OF TERMA SCANTER AND ARPA RADAR DATA SERVER

Maciej Sac[1], Sylwester Kaczmarek[2], Marcin Narloch[3]

Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Narutowicza 11/12 Str., 80-233 Gdańsk, Poland; e-mail: msac@eti.pg.edu.pl, kasyl@eti.pg.edu.pl, narloch@eti.pg.edu.pl; ORCID: [1]0000-0002-6734-3046, [2]0000-0003-2932-5610, [3]0000-0002-7640-2941

ABSTRACT

The software solutions presented in this paper generate real-time data compatible with ARPA radar standard as well as Terma SCANTER 2001 radar cooperating with Video Distribution and Tracking (VDT) server. Two different approaches to this problem are considered: emulation based on the data captured from real devices and simulation of objects on the sea. For both of them architecture, implementation details and functional test results are presented. The developed software will be used to test new functionalities of the multimedia surveillance system implemented for the Maritime Division of the Polish Border Guard within the STRADAR project.

Keywords:

radar data server, emulation, simulation, Video Distribution and Tracking, Terma SCANTER 2001, ARPA.

Research article

INTRODUCTION

The aim of the STRADAR system [1], developed for the Maritime Division of the Polish Border Guard (PBG), is to generate tasks for visualization of different types of data and present them in the Event Visualization Post (EVP), which includes a multi-display high-resolution screen. Each visualization task can contain ongoing or archival video from cameras, audio (telephone calls), SMS, files/photos or map data.

Currently, new software modules for the STRADAR system are developed, dedicated to visualization of radar data coming from ARPA radars [2] as well as Terma SCANTER 2001 radars cooperating with Video Distribution and Tracking (VDT) servers [15]. Sources of the above-mentioned radar data are necessary for implementation and testing of these new functionalities. Due to lack of access to physical devices and sources of real radar data streams, there are two software solutions available: emulation and simulation. In the first case, specially developed software radar servers provide (after proper transformations performed in real time) data gathered from real devices and imitate their behavior from functional point of view. The second approach regards a simulator of objects on the sea with radar data generation ability, formed accordingly to particular network protocols of radar data server.

The developed emulation and simulation software (its architecture and implementation details) is described in the next two sections. Functional tests of the implemented software are presented and discussed in section 'Software tests'. The paper is summarized in the last section ('Conclusions').

EMULATOR OF RADAR DATA SERVER

During the performed research three variants of radar emulation software were developed:

- emulator of a Terma SCANTER 2001 radar with a VDT implemented in C/C++ programming language;
- emulator of an ARPA radar implemented in Linux Bash scripting language;
- emulator of a Terma SCANTER 2001 radar with a VDT implemented in Linux Bash scripting language.

In the next part of this section all these three solutions are described.

The role of a Terma VDT server is to process raw radar data (e.g. from the SCANTER 2001 radar) and provide text position information (plots, tracks) as well as binary data (processed radar video) over a TCP/IP network. The VDT can be also used to change configuration parameters of the connected radar.

The idea of radar data server emulators for Terma devices (C/C++ and Bash variants) is to use the information gathered from a VDT and provide it at any time to radar visualization software client (such as SCANTER Radar Service Tool [8]) without any limitations. The following VDT protocols are available in the emulators (all protocols have a text format and use TCP for transport):

- Own Unit Management Protocol, OUMP [11] (provides the information about the unit with the SCANTER 2001 radar, among others position, speed, course; uses TCP port 18000);
- Plot Data Protocol, PDP [12] (provides plot data defined as [15] the group of connected radar cells in which the measured video signal exceeds a defined threshold value and/or fulfils some other discrimination criteria; these data correspond to real objects on the sea but also to waves and radar echoes; uses TCP port 17404);
- Track Control Interface Protocol, TCIP [13] (configures the process of extracting track data from plot data; allows defining positions of Aids to Navigation — AtoNs; uses TCP port 17394);
- Track Data Interface Protocol, TDIP [13] (provides track data [15] with the information about the objects on the sea (among others position, speed, course) identified by analysis of the plot data; uses TCP port 17396);
- Video Control Connection Protocol, VCCP [10] (configures and controls provisioning of processed radar video; uses TCP port 1500).

Moreover, the Video Distribution Protocol (VDP) [10] is supported, which is a binary protocol providing processed radar video.

The C++ variant of radar data server emulator for Terma devices was implemented under Debian Linux 8 operating system and g++ 4.9.2 compiler. For handling network communication the Boost.Asio 1.55 library [3] was used. The source code is contained in the `radar_server.cpp` file, which structure is illustrated in fig. 1.

During software development it was assumed that one instance of the emulator handles one VDT protocol (for a set of VDT protocols multiple instances should to be run). The protocol type is determined by the TCP port number given as the input parameter during start of the emulator (1500 for VCCP, 17394 for TCIP, 17396 for TDIP, 17404 for PDP, 18000 for OUMP). The chosen port number is passed to the `main()`

function of the emulator (as arguments of software process invocation), which creates a `boost::asio::io_service io_service` object (providing basic input/output functions) and calls the `server()` function. This function creates a `tcp::socket sock(io_service)` object, accepts TCP connections at the given port (`unsigned short port`) and handles them, for which the `session()` function is invoked.

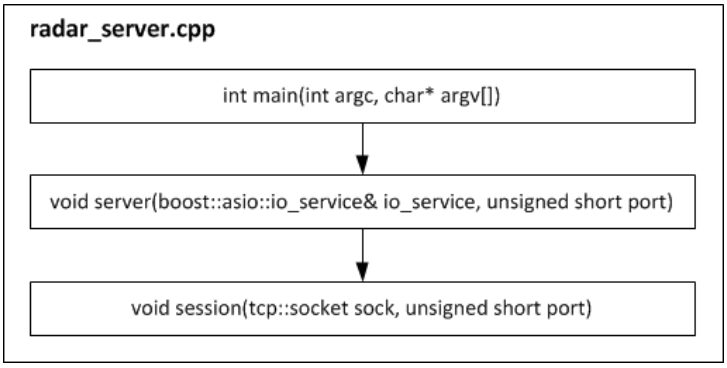

Fig. 1. Structure of C/C++ source code of the radar data server emulator for Terma devices

The `session()` function is the heart of the emulator, which implements particular VDT protocols. At the beginning of its operation it reads the data file for the chosen protocol. It is assumed that this file is located in the same directory as the emulator application (`radar_server`) and named [port].txt (e.g. 18000.txt for OUMP). The data file contains the information captured from a real VDT. Its each line concerns payload of one packet and includes relative generation time, tab key and packet data in hexadecimal notation. In our research traffic generated between the VDT and the client software (SCANTER Radar Service Tool) was captured using the Wireshark [16] protocol analyzer and saved as *.pcap files. Subsequently, the TShark [16] tool was used to extract data from *.pcap files to achieve properly formatted *.txt files for particular VDT protocols.

After reading the data file, the `session()` function performs operations dependent on the type of handled VDT protocol:

- for TCIP it generates TCP packets with the content and intervals defined in the data file, it also keeps the connection alive [13] (for each `ping` control message received from the client software it replies with a `pong` control message);
- for TDIP, PDP and OUMP similar actions are taken, additionally dates and times in the generated `track`, `extplot`, `gps`, `gyro`, `log` messages [11 – 13] are updated (replaced with current UTC dates and times);

– for VCCP the `session()` function informs the client software about the protocol version and SCANTER 2001 radar parameters (e.g. range) [10]; it also handles the request for processed radar video (it starts generating VDP packets from the data file to the destination UDP port indicated in the request) and keeps the VCCP connection alive (`PING`/`PONG` message exchange).

The emulator of an ARPA radar implemented in Linux Bash (Bourne-Again Shell) scripting language was designed with the aim of providing functionality of ARPA data source, achieved with simple tools, preferably existing within the operating system and its core applications installed by default. The idea behind development of a simple emulator of an ARPA radar was the ability to replay ARPA messages stored in a plain text file (one message per line) to any device available in Linux operating system, particularly (but not limited) to serial port (RS-232) or USB port with USB-serial converter, text console or even another file. The key feature of that solution is to provide archival ARPA messages with updated, current time and restore original time intervals between messages at the output. That possibility allows to emulate ARPA radar device with the aid of a PC computer equipped with a serial port and Linux operating system. In the proposed solution a simple Bash script executes an AWK script responsible for main processing of ARPA messages and provides it with the appropriate file name and current time in UTC±00:00 zone for time manipulation of NMEA (National Marine Electronics Association) 0183 [9] messages. AWK is a standard UNIX tool for advanced manipulation of text data organized in lines, which utilizes scripting language similar to C and should be considered as a simple extension to Bash. Despite its simplicity and existence of modern and more sophisticated scripting languages like Perl or Python, AWK is available on all UNIX/Linux platforms and seemed the first choice for fast development of a simple ARPA emulator. During the processing of particular ARPA NMEA messages stored in the source file, the AWK script calculates time interval between current and previous message in the file considering the first message as a reference (beginning of sequence), to print or emit the stream of messages with original timings. In that task a time field in the message is inspected, which unfortunately has different position from beginning for different ARPA message types. Moreover during that operation time field of the message is updated according to current time. That operation forces recalculation and update of the checksum field in the message. From implementation point of view NMEA checksum calculation procedure was the most demanding, due to limitation of the AWK scripting language regarding lack of support for binary operators, particularly binary XOR. However the problem was

solved with association tables available in AWK language. The last operation performed by the developed AWK script is emission (printing to the output) of updated message maintaining original timing between messages. Originally the output of the ARPA emulator was directed to a PC serial port used as a testing equipment in the STRADAR project. Other performed tests regarded possibility to transport ARPA messages over a TCP/IP network. In that case simple network tools were used to create TCP/IP clients and servers for text oriented protocol. That tools called 'netcat' and 'socat' are available for most UNIX/Linux platform and allow sending text data over IP network directly from Linux text consoles, files and devices without necessity to develop advanced programs, which is very useful in case when time and simplicity constrains are crucial.

The results achieved in the above-mentioned field were the inspiration for the development of a simple but functional emulator of a Terma SCANTER 2001 radar with a VDT in Linux Bash scripting language. The idea behind implementation of a Bash-based emulator of a Terma SCANTER VDT was similar to the ARPA radar emulator. A simple, script based TCP/IP tool was developed. The solution consists of a Bash script which coordinates execution of multiple TCP servers for different VDT protocols. Each TCP server is an instance of the 'netcat' process providing the connected client with the radar data from text files (which were extracted from *.pcap files with the aid of the TShark tool). The most difficult part of the implementation of Bash-based servers was programming the interaction between client and server, regarding handling of 'keep-alive' (PING/PONG control) messages, and development of VDT video protocol server as the most complicated among all used. As a result, a simple but functional emulator of a Terma VDT radar server was developed, which could be used as a testing tool (radar data generator) for the STRADAR project or when development of a VDT server network client as element of the STRADAR system is considered.

SIMULATOR OF RADAR DATA SERVER

As already mentioned, simulation of objects on the sea is another approach to generate real-time data compatible with ARPA radar as well as Terma SCANTER 2001 radar with a VDT server. In our previous research [4], an AIS/ARPA/GPS data generator was implemented, which aim is to simulate movement of tracking PBG ships and tracked ships on a definable part of the Baltic Sea as well as produce position

data streams according to AIS (Automatic Identification System), GPS (Global Positioning System) and ARPA (Automatic Radar Plotting Aid) radar specifications [2, 9, 14] in real-time. These position data streams are formatted as standardized NMEA 0183 text strings and sent to serial ports. The concept of the generator takes into account some details about the equipment of PBG, however, it is universal and its application is not limited to systems designed for PBG. The above-mentioned AIS/ARPA/GPS [4] data generator has been extended to provide position information compliant with Terma devices (SCANTER 2001 radar with a VDT server) in terms of OUMP and TDIP protocols. The resultant software, called the AIS/ARPA/GPS/Terma data generator, is described in this section.

The first important task for the AIS/ARPA/GPS/Terma data generator is to simulate movement of PBG ships and tracked ships on a definable part of the Baltic Sea. It is assumed that tracked objects are represented by large ships with class A AIS equipment [14]. Two categories of these objects with different sets of parameters are considered. Information about position of tracked ships is gathered by the following tracking units:

- PBG ships with AIS receivers and ARPA radars (radars are optional; maximum one radar per one PBG ship is assumed);
- PBG Observation Points (OPs) located at fixed position on the coast of the Baltic Sea (each one is equipped with a Terma SCANTER 2001 radar and VDT server);
- AIS receiver network of Maritime Authority in Gdynia located at the coast of the Baltic Sea (position data provided by the Web Service; these data concern the whole simulated part of the Baltic Sea and are distributed as AIS packets).

The simulated sea area is represented by a rectangle definable in configuration file and is by default given by the following coordinates: 54,5°N–55,22°N; 13,8°E–20,8°E. It assumed that both PBG ships and tracked ships sail straight lines and may periodically change speed according to defined random variables. Tracked ships generate AIS position data during movement (according to AIS standard [14]), which are gathered by PBG ships receivers with a limited range (defined in simulation parameters) and definable packet loss probability. PBG ships may also have ARPA radars installed to collect additional position information about tracked ships (for radars we also can define range and information loss probability). Contrary to AIS position data (which for the same tracked ship are identical in all PBG ships), information gathered for one particular tracked object by different ARPA radars may differ from each other. This is caused by azimuth and distance measurement errors (definable parameters), which are generally different for different

radars. Additionally, particular ARPA radars independently assign identifiers (target numbers) for tracked ships and in most cases the same tracked ship will have different identifiers assigned by different radars.

The radar operation principles of PBG OPs (with Terma devices) are similar to that of PBG ships (with ARPA radars). Observation Points are, however, fixed at the coast of the Baltic Sea (their number and location is definable). Moreover, they provide position data with a slightly different set of parameters and format (according to the specification of VDT OUMP and TDIP protocols [11, 13]).

The output of the generator are the following position data streams:

- data generated by PBG ships (one stream per one PBG ship) containing:
  - AIS data received by PBG ships with definable range and loss probability,
  - data from ARPA radars installed on PBG ships (definable range and errors of radar are taken into account),
  - GPS data with position of PBG ships;
- data generated by PBG OPs (two streams per one OP) containing:
  - position information about the OP (VDT OUMP protocol stream),
  - position information about the objects detected by the SCANTER 2001 radar installed in the OP (VDT TDIP protocol stream; definable range and errors of radar are taken into account),
- AIS data from Maritime Authority in Gdynia (Web Service) — one stream from the whole simulated part of Baltic Sea.

The AIS/ARPA/GPS/Terma data generator is implemented in the OMNEST [6] simulation framework (commercial version of OMNeT++). It allows running applications in graphical mode (Tkenv) and command-line text mode (Cmdenv). Graphical mode provides visualization of the simulated objects and allows detailed analysis of generated position data. In text mode the amount of available information is very limited, however, it consumes less processor resources and allows simulating more objects. In the OMNEST framework simulated objects (called modules), their parameters and connections between them are defined using NED (NEtwork Description) language [7]. Basic operations in the developed application are performed by simple modules, which functionality is described using C++ programming language. Simple modules may be grouped to form compound modules, which perform more complicated functions.

Structure of the AIS/ARPA/GPS/Terma generator software is depicted in fig. 2. The elements related to AIS, ARPA and GPS data generation functionality [4] are marked using black color. Objects tracked by PBG (large ships) are represented by the `tracked_ship` simple module. PBG ships are implemented in the `bg_ship` simple module. In the generator very important is also the `global` module, which is

responsible for providing other modules with values of global variables (through its global functions). This functionality is used by other modules of the generator both before starting to simulate movement of objects and during this simulation. In the first case global functions of the `global` module are used for example to set initial geographic positions of objects. In the second situation they are invoked among others to get new values of object speed.

The `global` module also receives AIS/ARPA/GPS data from PBG ships and AIS data from the `ma_webs` simple module representing the WebService of Maritime Authority in Gdynia. Position data gathered by the `global` module are converted from internal format used in the generator (messages – C++ objects) to standardized NMEA0183 text strings and sent over IP network to `ServerUDP_RS` applications. Each instance of the `ServerUDP_RS` application operates on a separate UDP socket (a pair of IP address and UDP port), receives one position data stream and sends it to a defined RS-232 port [5].

The set of internal generator messages related to AIS, ARPA and GPS data generation functionality includes (black solid lines with arrows in fig. 2):

- `ARPA_REQ`: requests sent to all tracked ships to receive their position, speed and course (equivalent of transmitting radio waves by the ARPA radar);
- `ARPA_RESP`: responses to `ARPA_REQ` messages (containing position information of particular tracked ships);
- `ARPA_DATA`: messages with information necessary to form ARPA and GPS NMEA0183 text strings in the `global` module; they are created in the `bg_ship` module based on the received `ARPA_RESP` messages;
- `AIS_M1`, `AIS_M5`: prototypes of AIS type 1 message (scheduled position report) and AIS type 5 message (scheduled static and voyage related vessel data report) [14]; they are sent from the `tracked_ship` module to the `bg_ship` and `ma_webs` modules, which forward them to the `global` module.

Blue color in fig. 2 represents new modules added for support of Terma VDT protocols (OUMP, TDIP). PBG OPs are implemented as the `observ_point` simple module, which interacts with the `global` module (it gets values of global variables) and the `tracked_ship` module (it receives position information from tracked ships — using `TERMA_REQ` and `TERMA_RESP` messages, analogically to the ARPA radar). For these interactions appropriate modifications of the source code of the `global` and `tracked_ship` modules had to be performed.

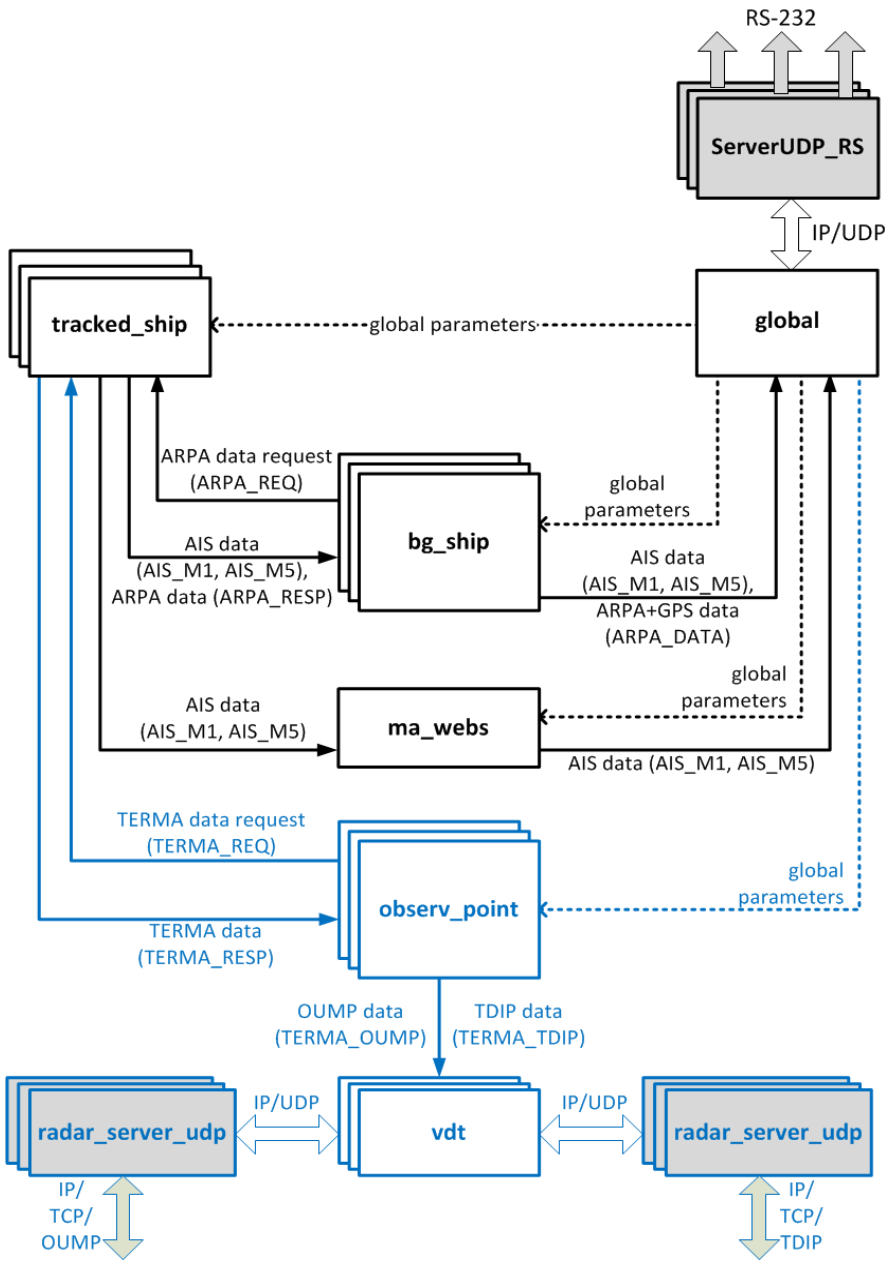

Fig. 2. Structure of the real-time generator of AIS/ARPA/GPS/Terma data; solid lines with arrows represent messages exchanged directly between modules, dashed lines with arrows — parameters passed using global functions of the `global` module; blue color is used to mark the modules added for support of Terma VDT protocols (OUMP, TDIP); `ServerUDP_RS` and `radar_server_udp` applications cooperate with the generator and are not its integral part

Having the position information about the tracked objects and itself, each instance of the `observ_point` module forms prototypes of TDIP and OUMP protocol messages (in internal generator format; `TERMA_TDIP` and `TERMA_OUMP` messages in fig. 2) and sends them to a dedicated instance of the `vdt` simple module. This module is responsible for creating TDIP `track` and OUMP `gps`, `gyro`, `log` messages in the format specified by the Terma company [11, 13] and sending them over IP network to `radar_server_udp` applications. Each instance of the `vdt` simple module cooperates with two instances of the `radar_server_udp` application, from which one receives TDIP and another one OUMP protocol messages over a unique UDP socket.

The `radar_server_udp` is a universal application that performs the role of VDT OUMP or TDIP protocol server based on the messages received (over UDP protocol) from the `vdt` simple module of the AIS/ARPA/GPS/Terma data generator. Its source code (`radar_server_udp.cpp` file – C/C++ programming language with Boost.Asio 1.55 library) is implemented based on the code of the `radar_server` (C/C++ radar data server emulator for Terma devices described in the previous section) and `ServerUDP_RS` applications. Due to the fact that messages received from the AIS/ARPA/GPS/Terma data generator already have proper format, dates, times and intervals, the tasks of the `radar_server_udp` application are limited to keeping the OUMP/TDIP connection alive and generating position data to the client software. Therefore its code is much simpler than for the `radar_server` emulator.

The source code of the AIS/ARPA/GPS/Terma data generator complies with the rules of the OMNEST framework [7]. Each simple module of the generator has implemented a `handleMessage()` callback function, which is responsible for handling all messages incoming to this module (sent by the simple module itself or by the other modules). The `initialize()` and `finish()` functions are also used, which allow performing some operations at the beginning (e.g. setting initial object parameters) or end of simulation (e.g. deleting dynamically created objects). Many other functions are additionally defined according to the needs, examples are public functions of the `global` module providing global simulation parameters to other modules.

Due to space limitations further details about the implementation, configuration and running the AIS/ARPA/GPS/Terma data generator as well as the cooperating applications cannot be provided. More information on these aspects can be found in paper [4], which regards the previous version of the generator (without support of Terma VDT protocols). The implementation of the new radar functionality is, however, strongly based on the previous one (for ARPA radars) and the idea is preserved.

SOFTWARE TESTS

Operation of radar emulation software (all variants) was consecutively verified during source code development. Additionally, when all functionalities were implemented, thorough final tests were performed, for which position data from PBG radar devices located in different places and of various time durations were applied. During tests the following tools were used:

- Linux terminal (for displaying debugging information, such as values of variables; this functionality was disabled in final software versions in order to increase performance);
- Wireshark protocol analyzer (for analysis of communication scenarios and content of messages exchanged between radar data server emulators for Terma devices and client software);
- SCANTER Radar Service Tool (client software allowing visualization of position data from Terma VDT servers).

Performed tests indicated that all variants of radar data server emulation software were properly implemented. For emulators of Terma devices this involved checking conformity of communication procedures with the manufacturer specifications and visualization of the generated position data using the client software (SCANTER Radar Service Tool). Examples of results for both these aspects are presented in the next part of the section. The described final test was performed for emulator of a Terma SCANTER 2001 radar with a VDT implemented in C/C++ programming language. Fragment of the obtained TDIP protocol message exchange is as follows (the Wireshark tool was used to capture messages; one line represents one message; all presented messages were sent by the emulator to SCANTER Radar Service Tool):

```
currval,Protocol revision,1.1
track,2500,2019,5,11,21,56,58,676,FA,TARGET,,24,60,19398,4.96230,54.8
    5939,18.06117,0.00000,0.00000,30,9,1,168,0.01593,264.00
track,2439,2019,5,11,21,56,58,676,FA,TARGET,,24,56,14527,5.04850,54.8
    5950,18.14029,0.00000,0.00000,30,9,0,82,0.01208,102.00
track,2673,2019,5,11,21,56,58,676,FA,TARGET,,60,196,26010,5.11190,54.
    90696,17.98028,5.20000,1.59210,30,9,0,236,0.01504,174.00
track,2750,2019,5,11,21,56,58,676,FA,TARGET,,52,492,32839,4.98470,54.
    89499,17.86088,4.40000,1.49740,30,9,0,351,0.01949,218.00
```

The first presented TDIP message (`currval,Protocol revision,1.1`), sent upon connection to the emulator, indicates the protocol version. The following `track` messages contain position information about the objects detected by the radar [13].

Fragment of the obtained OUMP protocol message exchange is presented below (the Wireshark tool was used to capture messages; one line represents one message; all presented messages were sent by the emulator to SCANTER Radar Service Tool):

```
currval,Protocol revision,1.1
gps,2019,5,11,21,56,58,888,54.81665,18.35385,0.00000,0.00000
gyro,2019,5,11,21,56,58,888,0.00000
log,2019,5,11,21,56,58,888,0.00000,0.00000
```

As in the previous case, the first OUMP message announces the protocol version. The remaining (`gps`, `gyro`, `log`) messages carry position information about the unit (PBG OP) with the SCANTER 2001 radar [11].

Fragment of the obtained communication scenario for VCCP protocol is as follows (the Wireshark tool was used to capture messages; one line represents one message; blue color is used to mark messages sent by the emulator; red color — messages sent by SCANTER Radar Service Tool):

```
VERSION 2.2 4096 4096 32
REQUEST_VIDEO_UDP 13 62527
OK
RANGE 61378
PING
PONG
PING
PONG
```

Upon connection the emulator sends the first VCCP message (`VERSION 2.2 4096 4096 32`) with the information about the protocol version (2.2) as well as radar video parameters (number of sweeps in a full scan, number of range cells in a complete sweep, sector size) [10]. Subsequently, the client software generates a request for unicast UDP based video (`REQUEST_VIDEO_UDP 13 62527`). The first parameter of this request (13) is the video type to subscribe [10] and the second one (62527) represents the UDP port number where the client is listening for incoming video data. The emulator confirms the received request (`OK`), gives information about the radar range in meters (`RANGE 61378`) and starts sending video packets (VDP protocol) to the specified UDP port. The VCCP protocol connection is periodically checked and refreshed using the `PING` and `PONG` keep-alive messages.

Visualization of the position data from the described test using SCANTER Radar Service Tool is illustrated in fig. 3. It contains processed radar video (the coastline; VCCP and VDP protocols), SCANTER 2001 radar position (large light blue point; OUMP protocol) as well as tracked objects positions (track data; white circles; TDIP protocol).

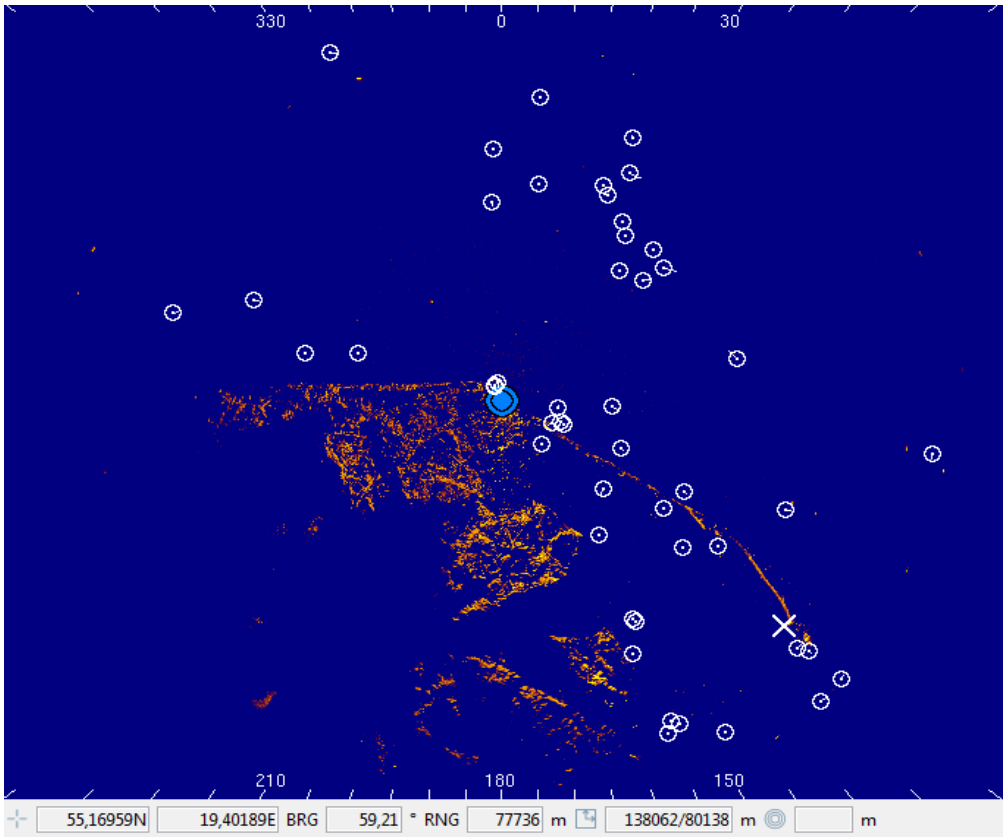

Fig. 3. Position data visualization from the C/C++ radar data server emulator for Terma devices using SCANTER Radar Service Tool (a fragment of window is presented)

Similarly to radar data server emulators, new functionality of the AIS/ARPA/GPS/Terma generator (support for Terma VDT protocols — OUMP, TDIP) was tested during source code development. Thorough final tests with various sets of input parameters were performed as well and confirmed proper software implementation. During the tests values of variables and contents of messages exchanged between the generator modules (fig. 2) were examined using both the text log available in graphical simulation mode (containing diagnostic information; bottom left hand corner of fig. 4) and the OMNEST Sequence Chart tool [7] (allowing detailed analysis of all events occurring during simulation).

The `radar_server_udp` application (performing the role of VDT OUMP or TDIP protocol server based on the messages received from the generator; fig. 2) was successfully tested analogically to C/C++ radar data server emulator for Terma devices. Linux terminal was used for debugging at the stage of source code development.

During all tests the Wireshark tool was used to check conformity of OUMP and TDIP communication procedures between the `radar_server_udp` application and the client software. Visualization of the generated position data was performed with the SCANTER Radar Service Tool and compared to the visualization built in the graphical mode of the AIS/ARPA/GPS/Terma generator (fig. 4).

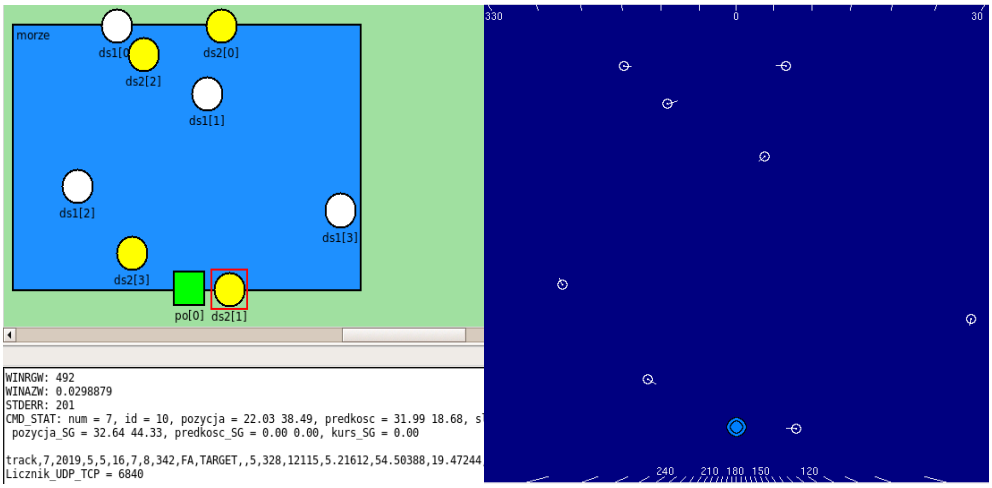

Fig. 4. The AIS/ARPA/GPS/Terma generator run in the graphical mode (left; the square represents a PBG OP with a SCANTER 2001 radar and VDT; circles are used to mark tracked ships) and visualization of the generated position data using SCANTER Radar Service Tool (right); fragments of windows are presented

CONCLUSIONS

The paper describes two approaches to generate real-time position data compatible with ARPA radar as well as Terma SCANTER 2001 radar cooperating with a VDT server: emulation and simulation. Presented emulators of radar data servers are based on the data captured from real devices. These data are properly transformed in real time (e.g. dates and times are updated) and made available to the client radar software. Three variants of radar data server emulators were developed: of Terma devices — C/C++ programming language, of Terma devices — Linux Bash scripting language, of ARPA radar – Linux Bash scripting language.

The aim of the radar data server simulator (called the AIS/ARPA/GPS/Terma data generator) is to simulate movement of tracking ships and tracked ships on a definable part of the Baltic Sea as well as produce position data streams according to AIS, GPS, ARPA radar as well as Terma VDT protocol specifications in

real-time. This paper is focused on the new functionality of the generator related to Terma VDT OUMP and TDIP protocols. Generation of AIS, ARPA and GPS position data is only briefly described as it was implemented in the previous version of the simulator [4].

For the developed software a description of architecture and implementation details are provided. Examples of test results, which confirmed its proper operation, are also included. The performed tests involved both analysis of correctness of the generated radar messages and visualization of position data using external tools.

The implemented emulators and simulator are very flexible tools, which can be used for testing different types of map software. Although the concept of the AIS/ARPA/GPS/Terma generator takes into account some details about the equipment of PBG, it is universal and its application is not limited to systems designed for PBG. It supports a very large set of input parameters (related to simulated objects, ARPA and Terma radars, AIS and GPS receivers, etc.) and two running modes (graphical mode convenient for debugging and text mode ideal simulating many objects). As a result, it is useful for both functional and performance tests of the cooperating systems. The concept of the described ARPA radar data server emulator is not limited to ARPA or even NMEA 0183 standard formatted messages, but allows providing the client software with any other text data.

The performed research indicated that implementation of more complicated protocols with Bash-based scripts is quite difficult and should be reserved for rather simple cases, where basic functions are important. Moreover, script-based processing of network protocols should be applied when performance is not crucial.

The presented software will be used to test new functionalities of the multimedia surveillance system implemented for PBG within the STRADAR project [1].

Acknowledgments

This work has been co-financed by NCBiR (The National Centre for Research and Development), project DOB-BIO6/10/62/2014.

The SCANTER Radar Service Tool software has been used with the approval of the Terma A/S company.

REFERENCES

[1] Blok M., Czaplewski B., Kaczmarek S., Litka J., Narloch M., Sac M., *STRADAR — multimedia dispatcher and teleinformation system for the Border Guard*, 'Scientific Journal of Polish Naval Academy', 2019, Vol. 216, No. 1, pp. 69–88, DOI: 10.2478/sjpna-2019-0006.

[2] Bole A., Dineley B., Wall A., *Radar and ARPA manual*, 2nd edition, Elsevier, Oxford 2005.

[3] *Boost.Asio-1.55.0*, [online], https://www.boost.org/doc/libs/1_55_0/doc/html/boost_asio.html [access 11.05.2019].

[4] Kaczmarek S., Sac M., *Real-time generator of AIS/ARPA/GPS data*, 'Scientific Journal of Polish Naval Academy', 2016, Vol. 204, No. 1, pp. 55–67, DOI: 10.5604/0860889X.1202435.

[5] *Maritime navigation and radiocommunication equipment and systems — Digital interfaces*, Part 3, *Multiple Talker and multiple listeners. High speed network bus.*, IEC document 61162-3, 2014.

[6] *OMNEST – High-Performance Simulation for All Kinds of Networks*, [online], http://www.omnest.com [access 11.05.2019].

[7] *OMNEST User Manual*, [online], http://www.omnest.com/documentation/Manual.pdf [access 11.05.2019].

[8] *Radar Service Tool*, Document No. 357641-NF, rev. M, Terma A/S, Denmark, 2007.

[9] Raymond E. S., *NMEA Revealed*, [online], http://www.catb.org/gpsd/NMEA.html [access 11.05.2019].

[10] *SCANTER Network Video Protocol. Software Interface Specification*, Document No. 304124-SI, rev. H1, Terma A/S, Denmark, 2016.

[11] *SCANTER Own Unit Management Protocol*, Document No. 304203-SI, rev. D, Terma A/S, Denmark, 2017.

[12] *SCANTER Plot Management Protocol*, Document No. 304284-SI, rev. H, Terma A/S, Denmark, 2011.

[13] *SCANTER Track Management Protocol*, Document No. 303949-SI, rev. F, Terma A/S, Denmark, 2011.

[14] *Technical characteristics for an automatic identification system using time-division multiple access in the VHF maritime mobile band*, ITU-R Recommendation M.1371-5, 02/2014.

[15] *Video Distribution and Tracking Unit, SCANTER 2001. Product Specification*, Document No. 306872-DP, rev. 3, Terma A/S, Denmark, 2007.

[16] *Wireshark · Go Deep*, [online], https://www.wireshark.org/ [access 11.05.2019].

EMULATOR I SYMULATOR SERWERA DANYCH RADAROWYCH TERMA SCANTER I ARPA

STRESZCZENIE

W artykule opisano oprogramowanie umożliwiające generowanie w czasie rzeczywistym danych pochodzących z radaru ARPA oraz radaru Terma SCANTER 2001 współpracującego z serwerem Video Distribution and Tracking (VDT). Rozważane są dwa różne podejścia do rozwiązania tego

problemu: emulacja bazująca na danych z rzeczywistych urządzeń oraz symulacja obiektów na morzu. Dla obu podejść zamieszczono opisy struktury oprogramowania, szczegółów jego implementacji oraz wykonanych testów funkcjonalnych. Opracowane oprogramowanie zostanie wykorzystane do testowania nowych funkcjonalności multimedialnego systemu nadzoru rozwijanego dla Morskiego Oddziału Straży Granicznej w ramach projektu STRADAR.

Słowa kluczowe:

serwer danych radarowych, emulacja, symulacja, Video Distribution and Tracking, Terma SCANTER 2001, ARPA.

Article history

Received: 05.06.2019
Reviewed: 13.07.2019
Revised: 23.07.2019
Accepted: 24.07.2019