Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

# SEDNA Voyage Data Recorder

*Deployable MSA Data Collection and Ex-Filtration*

*Andrew MacInnis*

**Defence R&D Canada – Atlantic**

Technical Memorandum
DRDC Atlantic TM 2013-135
April 2014

Canada

# SEDNA Voyage Data Recorder

*Deployable MSA Data Collection and Ex-Filtration*

Andrew MacInnis

**Defence R&D Canada – Atlantic**

Technical Memorandum
DRDC Atlantic TM 2013-135
April 2014

## Abstract

In 2012, the Maritime Information Support group at DRDC Atlantic began work on the SEDNA (Situational information for Enabling Development of Northern Awareness) Project with focused research into issues surrounding Maritime Domain Awareness in the Arctic. This paper describes the design, development, deployment and results of Maritime Situational Awareness data collection and ex-filtration system, the SEDNA Voyage Data Recorder (VDR). The goal of this effort was to design and test methods for remote collection and ex-filtration of Maritime Situational Awareness data. The SEDNA VDR system is built upon a commercial-off-the-shelf Voyage Data Recorder and uses controlling software and an Iridium L-band transceiver to re-transmit selected data to a central location via the military Iridium gateway. The SEDNA VDR was deployed on-board CFAV QUEST during the Arctic deployment of 2012 designated Q-346. In addition to traditional maritime sensors such as GPS, AIS and RADAR using the NMEA standard messaging, an ADS-B receiver was also installed and integrated into the system. In addition to backhauling data via Iridium, application-specific AIS messages were transmitted to ex-filtrate data via the exactEarth AIS satellite constellation as a proof-of-concept.

## Résumé

En 2012, le groupe du Soutien à la prise de décisions maritimes à RDDC Atlantique a commencé le projet SEDNA (information en support de la connaissance de la situation maritime Arctique) sur la recherche ciblée sur les questions portant à la connaissance du domaine maritime dans l’Arctique. Ce mémorandum technique décrit le modèle, le développement, et les résultats d’un système de collection et exfiltration de données de la connaissance de la situation maritime : l’enregistreur de données de voyage (VDR) SEDNA. Le but de cet effort était de concevoir et évaluer les méthodes pour la collection et exfiltration des données de la connaissance de la situation maritime. Le système SEDNA VDR était construit d’un enregistreur de données du voyage, un produit commercial, et il utilise un logiciel de commande et un émetteur-récepteur bande L de Iridium pour transmettre les données sélectionnées au lieu central en utilisant la passerelle Internet militaire de Iridium. Le SEDNA VDR était déployé à bord du NFAC QUEST durant le voyage vers l’Arctique en 2012 appelé Q-346. En supplément aux capteurs maritimes traditionnels tel que GPS, AIS et RADAR utilisant le standard de message NMEA, un récepteur ADS-B était aussi installé et intégré dans le système. En supplément à l’exfiltration de données par Iridium, les messages de AIS spécifiques au logiciel étaient transmis par la constellation de satellites exactEarth comme preuve de concept.

This page intentionally left blank.

# Executive summary

## SEDNA Voyage Data Recorder: Deployable MSA Data Collection and Ex-Filtration

**Andrew MacInnis; DRDC Atlantic TM 2013-135; Defence R&D Canada – Atlantic; April 2014.**

**Background:** Naval vessels gather maritime situational awareness (MSA) data from a suite of sensors. These data are then integrated to produce the local operating picture. This picture is used by the ship's command team in operation and tactical decision-making. While some important information may be forwarded to a shore-based operations centre, it is not standard procedure for the ship to then forward this picture to an operations centre. This paper describes the design, development, deployment and results from the Maritime Situational Awareness data collection and ex-filtration system, the SEDNA Voyage Data Recorder (VDR). The goal of this project was to design and test methods for remote collection and ex-filtration of MSA data. Data are collected by the system and then re-transmitted using the military Iridium gateway in Hawaii. As a proof-of-concept, a constellation of commercial AIS satellites was also used for data back-haul.

**Results:** The SEDNA VDR was deployed aboard CFAV QUEST during her deployment to the Arctic in the summer of 2012. During this trial the SEDNA VDR successfully collected unclassified MSA data from the ship's GPS, RADARs, AIS as well as an installed ADS-B receiver. The data were then successfully ex-filtrated in real-time to DRDC Atlantic allowing for a continuously updated display of QUEST's local operating picture. The second method of transmitting application-specific AIS messages to passing commercial satellites was successful but, as expected, was not a reliable communication medium.

**Significance:** The successful deployment of the SEDNA VDR demonstrates the ability to reliably re-transmit in real-time the local operating picture of a vessel or other collection node. The military Iridium link used is available at a flat-rate usage plan allowing for continuous data transmission at a fraction of the cost of commercial satellite services. The link was not encrypted but use of the US Department of Defense approved secure sleeve is a possible encryption option should this be required. A potential use of this technology is the transmission on contact data to a remote vessel to expand their MSA beyond their sensor horizon.

**Future plans:** The future plans for the SEDNA VDR include a subsequent QUEST deployment to waters off Halifax in the fall of 2013 and potentially redeployment to the Arctic in the summer of 2014. A goal in 2014 will be to provide, via military Iridium, an over-the-horizon operating picture to QUEST to enhance her own operational plot.

# Sommaire

## SEDNA Voyage Data Recorder: Deployable MSA Data Collection and Ex-Filtration

**Andrew MacInnis; DRDC Atlantic TM 2013-135; R & D pour la défense Canada – Atlantique; avril 2014.**

**Introduction ou contexte :** Les navires de la marine recueillent les données de la connaissance de la situation maritime (MSA) d'un ensemble de capteurs. Ces données sont ensuite intégrées pour produire l'image d'exploitation locale. Cette image est utilisée par l'équipe de commandement du navire pour la prise de décisions opérationnelles et tactiques. Même si l'information importante peut être transmise à un centre d'opérations à terre, transmettre l'image d'exploitation locale n'est pas une pratique courante. Ce mémorandum technique décrit le modèle, développement, et les résultats d'un système de collection et exfiltration de données de la connaissance de la situation maritime, soit l'enregistreur de données du voyage (VDR) SEDNA. Le but de ce projet était de concevoir et évaluer les méthodes pour la collection et exfiltration des données de la connaissance de la situation maritime. Les données sont collectionnées par le système et transmises de nouveau en utilisant la passerelle Internet militaire de Iridium à Hawaii. Comme preuve de concept, on utilisait aussi une constellation de satellites AIS commerciaux pour l'exfiltration des données.

**Résultats :** Le SEDNA VDR était déployé à bord du NFAC QUEST durant le voyage vers l'Arctique dans l'été de 2012. Durant le voyage le SEDNA VDR a collectionné avec succès les données MSA non-classifiées du GPS, AIS et les radars du navire et aussi un récepteur ADS-B installé à bord. Les données étaient ensuite exfiltrées, en temps réel, à RDDC Atlantique, ce qui permettait la mise à jour continuelle de la visualisation de l'image d'exploitation local de QUEST. La deuxième méthode d'exfiltration, la transmission des messages AIS spécifiques au logiciel a été un succès mais, comme on s'y attendait, ne représentait pas un moyen de communication fiable.

**Importance :** Le succès du SEDNA VDR démontre la capacité de transmettre en temps réel, et de façon fiable, l'image d'exploitation locale d'un navire ou d'un autre nœud de collection. Le lien militaire Iridium utilisé est disponible via un plan d'utilisation forfaitaire qui permet la transmission continuelle de données à une fraction du coût des services de satellites commerciaux. Le lien n'a pas été chiffré, mais l'utilisation du manchon sécurisé, approuvé par le Département de la Défense des États-Unis est une option possible de cryptage si nécessaire. Une utilisation possible de cette technologie est la transmission des données des contacts maritimes à un navire à distance pour élargir leur MSA au-delà de leur horizon de capteur.

**Perspectives :** Les plans futurs pour le SEDNA VDR comprennent un déploiement ultérieur sur le QUEST dans les eaux au large de Halifax dans l'automne de 2013 et potentiellement un déploiement en Arctique dans l'été de 2014. Un objectif en 2014 sera de fournir, via Iridium militaire, une image d'exploitation au-delàs de l'horizon à QUEST pour améliorer sa propre image opérationnelle.

# Table of contents

# List of figures

# List of tables

## Acknowledgements

The author would like to thank Sean Webb and Jason Murphy for their assistance with Python scripting and trouble-shooting, Vincent Boucher and Dang Phan for technical support and equipment installation, Bob MacDonald and Mark Baldin for their support onboard QUEST during Q346, Anthony Isenor for his assistance in writing and proof-reading of this report, Sean Webb, Anthony Isenor and Liesa Lapinski for setting up and maintaining the data display segment at DRDC Atlantic.

This page intentionally left blank.

# 1 Introduction

## 1.1 Background

The surveillance of Arctic North America traces its roots to the establishment of the North American Aerospace Defense Command (NORAD) in 1957. During the 1950's, the United States and Canada invested hundreds of millions of dollars in the construction of the Distant Early Warning (DEW) Line. The DEW Line was a string of RADAR stations that ran from Alaska, through the Canadian Arctic and onto Greenland. The stations' RADAR coverage areas provided a continuous screen that was designed to detect a Soviet airborne invasion.

In the following three decades, the Northern frontier was constantly monitored in hopes of detecting the earliest signs of a Soviet Invasion. The DEW Line was deactivated in the early 1990's after the fall of the Soviet Union. While this project did provide some economic development for the communities in the North and helped Canada assert its sovereignty in the region, the costs were enormous. In addition, the sites generated a considerable amount of hazardous waste, the handling of which is ongoing and at significant cost to the Canadian taxpayers. Since the DEW Line, there has been a lack of land-based sensors to collect data on air, land and sea based activities in the North.

The Cold War effectively ended in the early 1990's but Arctic surveillance is once again a priority of the Canadian government. Factors such as climate change and depleting natural resources have renewed interest in the Arctic to Soviet-era levels. Commercial shipping traffic is expecting to increase significantly as the North-west Passage is beginning to be ice-free and navigable in the summer months. This will make maritime surveillance in this environment a matter of increasing importance.

To date, the United States have arguably played a larger role in monitoring the Canadian North than Canada. However, the Canadian Forces and Royal Canadian Navy in particular, see Canada as a three-ocean state and are committed to developing the requisite capabilities. "By FY12/13, I envisage that significant scientific progress will have been made, in amongst other areas, in developing the information architectures required for maritime domain awareness in our three-ocean maritime estate." *Vice-Admiral D.W. Robertson*, CMS, from 12 Mar 08 Maritime Commander's Intent for 2009 to 2012.

Maritime Domain Awareness (MDA) is defined by the International Maritime Organization (IMO) as the effective understanding of anything associated with the maritime domain that could impact the security, safety, economy, or environment [1]. The Canadian Arctic offers significant challenges to the development of MDA in this region. It is a vast area with extreme weather including punishing winter temperatures and near constant darkness that make operating in the winter months extremely difficult. As well, the Arctic is a sparsely populated area, resulting in relatively little infrastructure to leverage. The cost of establishing and maintaining facilities is enormous and it is often cost prohibitive to transport the required supplies. Finally, the Canadian Arctic is a vast archipelago with some 36,000 islands [2]. The complex geography of an archipelagic region increases the difficulty of surveillance. It also makes transiting through the region more difficult.

Today the threat of invasion from the North has diminished but there is still a need to achieve MDA in the Arctic. Illegal fishing activities, the possible pollution of the environment by vessels and support to search and rescue can be added to defence on the list of concerns in the North. These concerns reflect the increasingly inter-departmental nature of Arctic MDA. The costs associated with Arctic projects further serve to motivate the government to employ its Whole-of-Government Framework where the Artic is concerned.

However, when multiple departments are working together, sensitivity and classification of information can become an issue. All opportunities to collect information and share it across departments should be seized. To mitigate already high costs, every chance to leverage existing technology in new ways should also be exploited. Keeping data at the lowest security classification levels (i.e., unclassified) will allow for more efficient data collection, dissemination and use.

Modern technological solutions to the problem of MDA in the North have much to offer. Satellites can provide a wealth of information, and while they have low maintenance costs, the initial costs are large. Satellites also have limitations such as low revisit rates and low resolution of their on-board sensors due to high altitudes. As well, satellites do not provide Canada with the same establishment of presence that came with the DEW Line stations and the people who manned them. Research into remotely controlled, unmanned monitoring stations is progressing. DRDC's Northern Watch Project is an example of an unattended sensor suite that could monitor strategic 'choke-points' along the North-west Passage [3].

Another modern development that could be leveraged for Arctic surveillance is the ship-borne voyage data recorder (VDR). In 2002, the IMO mandated the carriage of VDRs on all cargo vessels of at least 3,000 gross tonnes. In 2006, as a result of the sinking of the BC ferry *Queen of the North*, the Canadian government extended this requirement to Canadian passenger vessels of at least 500 gross tonnes [4].

The VDR is analogous to the flight data recorder, or 'black box' used in aircraft to support incident investigation. The maritime VDR records various data including the ship's position, course and speed, the state of the rudder and engines, status of certain water-tight doors and alarm systems, as well as audio from bridge microphones. Situational awareness (SA) data are also collected in the form of Automatic Identification System (AIS) and RADAR contacts received by the vessel. All of these data help investigators reconstruct events and provide context in the case of a collision or other serious incident.

The SA data are of interest to the ship's captain and crew but may also be of interest to other parties. Organizations such as the DND Regional Joint Operation Centres (RJOC), the Canadian Coast Guard and joint organizations such as the Joint Rescue Coordination Centres (JRCC) and Maritime Security Operations Centres (MSOC) require SA data in their day-to-day operations. A vessel transiting in the North that could communicate SA data with organizations such as these could provide the vital information to rescue a wayward sailor, persecute a drug trafficker or alert the military of a potential threat to national security.

The ex-filtration of SA data from a remote collection point to a shore-based location has been investigated in the past. The Advanced Vessel Monitoring System (AVMS) was a remote AIS collection system with satellite communication capability to back-haul data [5]. It was a deployable system built into a pelican case for easy installation to locations exposed to the elements. AVMS was successfully deployed on vessels of opportunity, such as a Maritime Coastal Defence Vessel (MCDV) and on the Hibernia oil platform. The latter installation was particularly popular with the RJOC at Canadian Forces Base (CFB) Halifax. After four years the AVMS project wound down as the units began to suffer end-of-life problems. The success of AVMS led to the decision to create a follow-on system with enhanced capabilities and that would address some lessons learned.

## 1.2 Outline

This report will present the results of the first field trial of the follow-on system to AVMS, the SEDNA VDR. First, an introduction will describe the basic design of the SEDNA VDR followed by a description of the field trial objectives. This is followed by a description of the trial concept and technical implementation. The results and analysis will follow, concluding with thoughts on future work.

The investigation of MSA data collection and ex-filtration involving the VDR on Q346 was composed of a set of discrete experiments. As a whole, these experiments encompass the collection, processing and ex-filtration of MSA data through a pair of communication links. The following list serves as a quick reference to the location of the experimental details:

- Data backhaul using military Iridium – description: 3.2.5.1, results: 4.1.
- Northern Situation Awareness Simulator Validation – description 3.1, results: 4.4.
- Data backhaul using exactEarth AIS satellites – description: 3.2.5.2, results: 4.5.1.
- Use of type 15 interrogation messages to increase likelihood of vessel detection by satellite – description: 3.2.5.2, results: 4.5.2.

# 2 System Design

## 2.1 SEDNA VDR

The SEDNA VDR system is the follow-on effort to AVMS. As with the preceding system, the SEDNA VDR was designed to receive (AIS) and Global Positioning System (GPS) data and use satellite communications to backhaul the data. In addition to AIS, the new system would be able to receive additional data feeds such as RADAR and Automatic Dependent Surveillance – Broadcast (ADS-B).

The SEDNA VDR design took into account two primary lessons learned from AVMS. The first concerned the lack of DRDC technical expertise in the AVMS system. As it was developed by a contractor, DRDC personnel were unable to address hardware and software malfunctions without incurring contracting costs and significant delays in operation while the units went out for repair. As a result, it was decided that the SEDNA VDR development would be conducted by DRDC. The second major concern with AVMS was the cost of satellite communications for the data backhaul. The commercial Iridium network was used and airtime costs often reached monthly costs in the thousands of dollars. Indeed, when the research funding for AVMS ceased, the RJOC was unwilling to assume the satellite costs to keep AVMS operational despite its popularity with the operators. This led to the investigation of the military Iridium service managed for the United States Department of Defense (US DoD) by the Defense Information Systems Agency (DISA).

The SEDNA VDR system comprises hardware and software components whose purpose is to receive, store and forward in real-time situational awareness data to a central location. Data from various ship sensors are stored and a selection is then retransmitted so that the data comprising the local operating picture can be sent to another entity, such as an operations centre to support maritime operational decision making. The development of the SEDNA VDR began in the spring of 2011 with the goal of having a prototype system operational in time to sail aboard Canadian Forces Auxiliary Vessel (CFAV) QUEST on her deployment to Gascoyne Inlet for trial Q346. As with AVMS, the SEDNA VDR was developed using commercial off-the-shelf hardware components combined with software to integrate the components and run the algorithms.

The core of the system is a Rutter VDR-100G3 Voyage Data Recorder [6]. This is a commercial off-the-shelf Windows-based PC that fulfills the SOLAS-class vessel requirement to record voyage data for the purposes of incident investigation. This 'black box' is analogous to the flight data recorders used in aircraft. The VDR-100G3 stores these data in a proprietary format which may be played-back using Rutter software. Additional software is used to do real-time decimation and forwarding of the data via satellite. The VDR100-G3 that was acquired from Rutter did not contain some of the components that were deemed unnecessary for this project such as the hardened data capsule and the audio and video processing units.

The back-haul component of the SEDNA VDR is a 9522B Iridium L-band transceiver (LBT) with a DoD Iridium SIM card. The LBT establishes a connection with DISA's Apollo server via the DoD Iridium gateway in Hawaii. The server assigns the LBT an IP address and the data then flow from the VDR to a server at DRDC in the form of User Datagram Protocol (UDP) packets. The Transmission Control Protocol (TCP) is favored but due to software limitations and time constraints, UDP was used for the Q346 trial.

# 3 Q346 Trial

## 3.1 Trial Objectives

The Q346 Trial was a deployment of CFAV QUEST on a round-trip from Halifax, Nova Scotia to Gascoyne Inlet, Devon Island, Nunavut with two stops for fuel in Nuuk Greenland. The vessel's track is shown in Figure 1. Its main objective was to support the Northern Watch Technology Demonstration Project as a target and as a freighter delivering equipment to the camp on Devon Island [3].

Various DRDC groups and several other government departments took advantage of the opportunity to conduct their own work in the Arctic environment as secondary payloads on Q346. The Maritime Information Support (MIS) group at DRDC Atlantic was one such group and used Q346 as the first field trial of the SEDNA VDR. Due to bunk availability on board QUEST, the plan was for the SEDNA VDR System to run autonomously and unattended for the first 38 days of the trial. The MIS engineer who developed the system (i.e., author) would meet the ship in late August to begin data analysis and to modify the system's configuration if required.

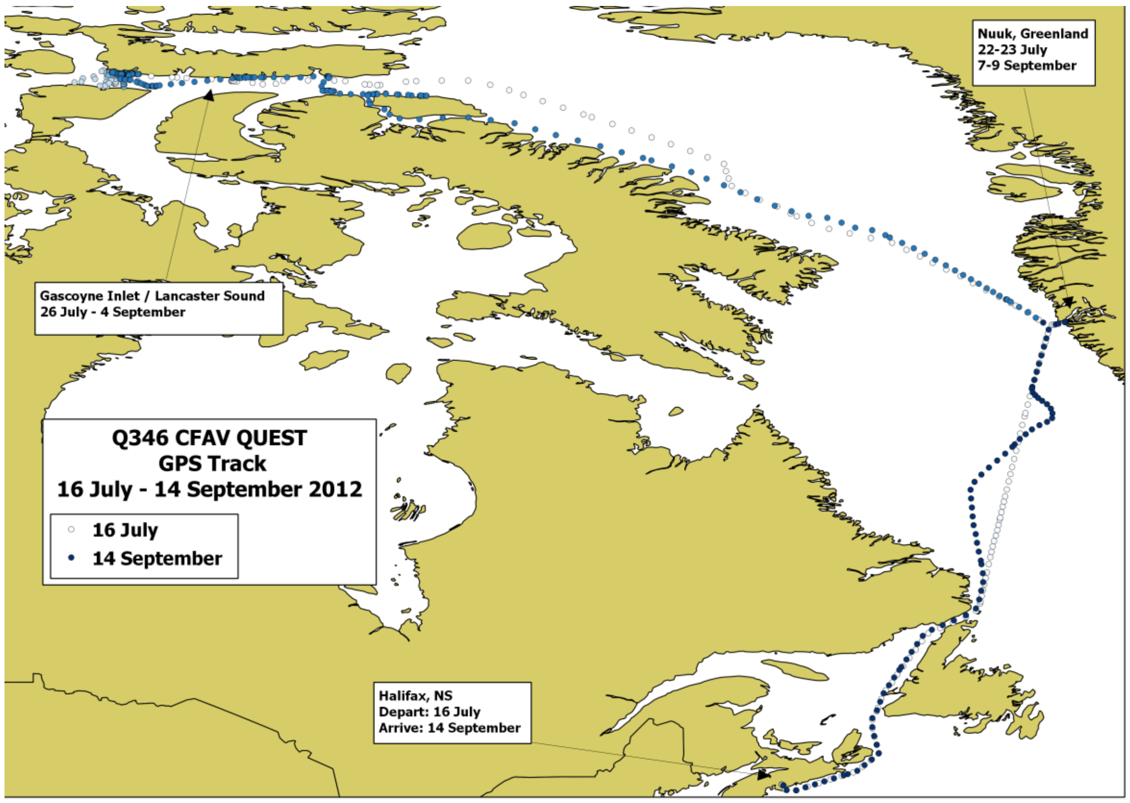

*Figure 1: Track of CFAV QUEST during Q346. Lighter dots represent the beginning of the voyage and darker dots the end.*

There were two primary objectives for MIS during Q346. The first was to collect a 'ground-truth' data set of MDA data in the Arctic environment. These data would be used to evaluate unclassified MDA sources such as satellite AIS data. These data will also provide information on AIS propagation characteristics in the Arctic and be used as a validation tool for the MIS Northern Situational Awareness Simulator. This Simulator is an Analytical Graphics Inc., Systems Tool Kit based application designed to model and predict contact detections in the Northern environment [7] [8]. The second objective was to trial a data backhaul mechanism employing the military Iridium gateway and the decimation and forwarding software. Of particular interest was the performance at high latitudes.

For the MIS effort on Q346, a pair of AIS-related experiments were also conducted. The first involved the use of application-specific AIS messages as a secondary backhaul mechanism. Data selected for backhaul would be encoded as an AIS message then transmitted to passing exactEarth AIS satellites. The AIS standard allows for users to send short messages to another user or all users as a broadcast. Instead of encoded text, the bits can be used to encode any information such as RADAR or ADS-B contact details. Messages received on these satellites would then be delivered to DRDC Atlantic as part of the active spaced-based AIS data delivery contract with exactEarth incurring no additional cost. The objective was to determine if this mechanism is a viable and reliable option for data backhaul. The second experiment involved the use of the AIS interrogation message to increase the amount of position reports transmitted by the vessels QUEST would encounter. The objective was to determine if this practice would improve the detection probability of vessels by AIS satellites and the quality of the resulting tracks.

## 3.2 Trial Concept and Technical Implementation

This section describes the technical details of the SEDNA VDR system and the details of the configuration used during the Q346 trial.

### 3.2.1 System Configuration

The following describes the configuration used during the first field test of the SEDNA VDR system on CFAV QUEST during Q346. Figure 2 shows the main components of the SEDNA VDR.

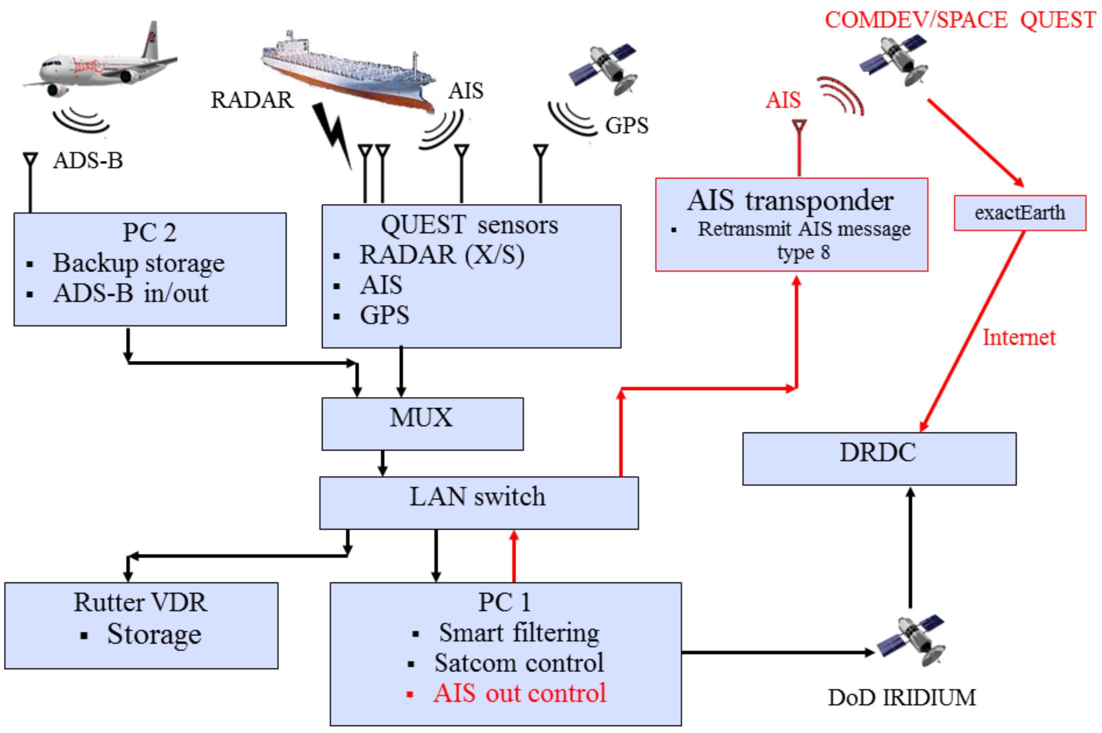

*Figure 2: SEDNA VDR data flow. The Iridium backhaul route is shown in black and the AIS via satellite route in red.*

The VDR-100G3 contains a multiplexer known as the National Marine Electronics Association (NMEA) module. This module can receive up to eight separate data feeds in the NMEA 0183 format [9]. For Q346, the ship's two RADARS, GPS and AIS were fed into the NMEA module. Additionally an ADS-B feed was also introduced from a stand-alone receiver. The data from the NMEA module are also fed via network switch into a Panasonic Toughbook running Windows 7 that runs the decimation/forwarding script written in Python. The purpose of this code is to vet the incoming data by discarding unimportant message types and by using user-defined parameters to decimate pertinent messages to reduce the bandwidth required to back-haul the data.

Connected to the serial port of the Toughbook is an Iridium 9522B L-band transceiver [10] used to forward the data to a static Internet Protocol (IP) address on the R&D segment of the DREnet at DRDC Atlantic. This is done using a point-to-point protocol (PPP) connection to the military Iridium gateway in Hawaii that essentially provides the modem with an IP address and an open internet connection. This service is called Iridium Direct Internet. A stand-alone Kongsberg AIS200P Class A AIS transponder [11] was also used to forward data in the form of type 8 Binary messages over the exactEarth constellation of polar-orbiting AIS satellites. Figure 3 shows the location of the AIS200P on board CFAV QUEST.

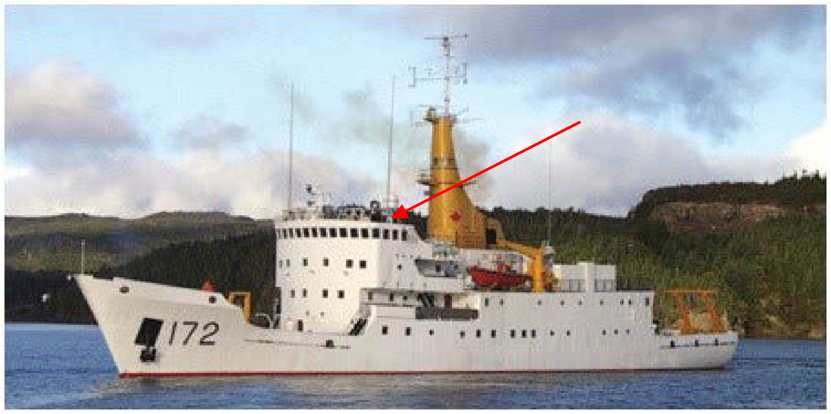

*Figure 3: CFAV QUEST with the location of SEDNA VDR antennas and secondary AIS during Q346 indicated.*

The Northern Situational Awareness Simulator was used to determine when the exactEarth AIS satellites would be overhead to receive the type 8 transmissions [12]. An approximate track for QUEST was plotted using the Northern Situational Awareness Simulator prior to the trial. The only fixed dates and positions were the start and end of the trip as well as the two stops in Nuuk, Greenland. The rest of the track was input assuming QUEST would spend the time between Nuuk visits in the vicinity of Gascoyne Inlet except for the time spent transiting to and from Nuuk. The orbits of the three exactEarth satellites were also plotted and communication windows with QUEST were. A buffer of three minutes was added to the beginning and end of each predicted window to allow for QUEST's deviation from the estimated track and beyond line-of-sight propagation.

A second laptop (see PC2 in Figure 2) was used as a redundant data-logger using a plain text format. It was also used to perform all of the functions related to ADS-B including running the proprietary software to decode the ADS-B messages, plot the contacts, and running the Python script used to translate the messages into a NMEA standard and forward them to the VDR.

### 3.2.2 Data Types

The SEDNA VDR system uses the NMEA 0183 format. All of the ship's sensors output data in this format. The NMEA module multiplexes the incoming streams and appends a two-digit channel designation as per Table 1.

This channel designation is useful when dealing with multiple sensors that use the same sentences and NMEA talker ID as was the case with the two RADARs. All of the NMEA sentences were logged locally while only specific sentences were considered for forwarding via satellite.

The ship's X and S-band RADARs (talker ID 'RA') output the OSD, RSD and TTM messages of which only the TTM were considered for forwarding. These 'tracked target messages' contain information of targets currently being tracked by the RADAR. The OSD and RSD messages provide information about the current settings of the RADAR's Automatic RADAR Plotting Aid (ARPA) display.

*Table 1: The various data streams feeding into the VDR and their corresponding NMEA multiplexer channel identifier.*

| Sensor Input | NMEA Mux Channel Identifier |
|---|---|
| AIS | 11 |
| GPS | 12 |
| S-Band RADAR | 13 |
| ADS-B | 14 |
| X-Band RADAR | 15 |

The ship's GPS (talker ID 'GP') outputs various NMEA sentences including RMC, GGA, GLL, RMB, VTG, XTE, APB, BOD, BWR and ZDA. Only the RMC (Recommended Minimum C) messages were considered for forwarding and contain the current time, position, course and speed of CFAV QUEST.

During Q346, QUEST employed two different Class A AIS transponders (talker ID 'AI'). For the first four weeks a Saab R4 was used and was switched out for an L-3 PROTEC AIS with Secure Tracking on 11 August. The secure tracking function was not enabled. Both transponders output the VDO (own-ship) and VDM (received from other transponder) messages. Only the VDM messages were considered for forwarding. Additionally, the Saab R4 outputs a proprietary PSTT,10C message once a minute containing the number of currently received AIS contacts. This message was also forwarded on every tenth occurrence.

The Kinetic SBS-1 receiver was used to receive ADS-B transmissions on Q346 [13]. The SBS-1 is a 1090MHz Extended Squitter ADS-B 'Mode S' receiver. This receiver requires Kinetic's proprietary BaseStation software to receive data and a second laptop was used for this function. While the BaseStation software is running, the raw ADS-B sentences are output on a virtual port on the local host.

Due to legacy constraints, the Mode S ADS-B data-link is severely limited in the amount of data that can be included per transmission. There are eight different sentence types, each containing a subset of the aircraft parameters. These include dynamic parameters such as position, speed, altitude and course and static parameters such as ICAO designator and call-sign. As ADS-B is not a maritime system, it does not use the NMEA standard.

A translation script was written in Python to convert the ADS-B sentences into the NMEA format. The new NMEA sentences are then output on a serial port and into one of the channels of the NMEA module in the same manner as the other feeds. The translation script takes in these ADS-B messages and outputs a Type 9 (search and rescue (SAR) aircraft) VDM AIS message with 'BS' for talker ID. Only the position, altitude, course and speed fields are used. Other parameters present in the type 9 AIS message are not contained in ADS-B messages, such as rate of turn, and are therefore left blank. The Maritime Mobile Service Identity (MMSI) field is replaced by the 24-bit ICAO designator using the formula below.

ADS-B MMSI = 800000000 + decimal representation of the 24-bit ICAO code. (1)

For example, an ICAO designator of A342C9 becomes (A342C9 -> 10699465 + 800000000 = 810699465). The range for converted ICAO designators is then 800000000 to 816777215. No countries use MMSIs beginning with an eight, thus making confusion with a ship impossible. To make the messages even more readily identifiable, the repeat indicator in all messages is set to 3, another extremely rare occurrence in AIS. The call-sign of the aircraft is included in place of some of the unused fields at the end of the Type 9 VDM message. This call-sign is encoded using the special six-bit ASCII table included in the AIS Specification [14]. Due to the type 9 message's design for use with low-flying SAR aircraft, the translating script reports the altitude in 10's of metres rather than just metres as indicated in ITU-R M.1371-4. The sentences include a proper checksum so that they appear to be a valid NMEA sentences and can be processed as such by the NMEA module, the VDR and other third-party software. Table 2 shows the full bitwise payload or the modified type 9 AIS message for ADS-B contacts.

*Table 2: Format of the modified type 9 AIS message for ADS-B contacts.*

| Parameter | Number of bits | Description |
|---|---|---|
| Message ID | 6 | Identifier for Message 9; always 9 |
| Repeat Indicator | 2 | Used by the repeater to indicate how many times a message has been repeated. Always set to 3 for ADS-B contact |
| User ID | 30 | Decimal representation of 24-bit ICAO ID, plus 800 000 000 |
| Altitude | 12 | In 10's of metres |
| Speed over ground | 10 | Speed over ground in knot steps (0-1 022 knots) 1 023 = not available, 1 022 = 1 022 knots or higher |
| Position Accuracy | 1 | Always set to 0 (default) |
| Longitude | 28 | Longitude in 1/10 000 min (±180º, East = positive (as per 2's complement), West = negative (as per 2's complement) |
| Latitude | 27 | Latitude in 1/10 000 min (±90°, North = positive (as per 2's complement), South = negative (as per 2's complement) |
| Course | 12 | Course over ground in 1/10 degrees (0-3599) 3 600 = not available, 3 601 – 4 095 not used |
| Call-sign | 36 | Six character call-sign if available, otherwise set to zeros (@@@@@@) As per ITU-R M.1371-4 Table 44, Annex 8 |
| Fill bits | 4 | All messages will end with 4 trailing zeros |
| Total number of bits | 168 | This single slot message is the same length as all AIS position reports, with the appropriate checksum. Ex: !BSVDM,1,1,,A,9swM;KD;7eJ0JqLapDiRnjk3LdM0,4*7C |

### 3.2.3 Data Logging

The data from Q346 were stored in two locations. The first is in the proprietary file-system of the Rutter VDR 100-G3. This data format allows for playback using Rutter software. The VDR's internal clock is kept to within two seconds of UTC by the 100G3 software using the ship's GPS data feed. The raw NMEA messages can be extracted from files in the VDR format however the precise time-stamp cannot.

The data were also recorded on a secondary laptop running Windows Vista using a pair of freeware applications called AIS Decoder and NMEA Router [15]. AIS Decoder was used from July 16 to September 5 and NMEA Router used for the remainder of the trial. The only difference in the two applications is the format used in the log files. AIS Decoder simply appends a human-readable time stamp to the end of each NMEA sentence while NMEA Router uses the NMEA 2000 format and uses the UNIX epoch timestamp. Examples of each data format are shown in the figures below. Figure 4 shows a sample from AIS Decoder and Figure 5 a sample using NMEA Router.

```
12$GPZDA,030009,03,09,2012,05,00*4C,03/09/2012 3:00:11 AM
15$RAOSD,278.2,A,277.0,P,5.2,P,,,N*4C,03/09/2012 3:00:11 AM
15$RARSD,,,,,,,,,,,12.0,N,N*67,03/09/2012 3:00:11 AM
15$RATTM,08,8.08,288.6,T,0.8,118.2,T,1.1,80.4,N,,T,*6A,03/09/2012 3:00:11 AM
11!AIVDO,1,1,,,14eLFt000nrE8@Db;wR;3`dB0000,0*1C,03/09/2012 3:00:11 AM
15$RATTM,07,1.51,167.3,T,0.9,089.6,T,1.4,-4.8,N,,T,*73,03/09/2012 3:00:11 AM
14!BSVDM,1,1,,A,9swM;KD;7eJ0JqLapDiRnjk3LdM0,4*7C,03/09/2012 3:00:12 AM
13$RAOSD,278.2,A,283.0,P,5.4,P,,,N*41,03/09/2012 3:00:12 AM
13$RARSD,0.00,,0.40,,,,,,,,3.0,N,N*53,03/09/2012 3:00:12 AM
12$GPAPB,A,A,0.00,R,N,,,278,T,001,278,T,278,T*1E,03/09/2012 3:00:12 AM
12$GPBOD,278,T,329,M,001,,*5F,03/09/2012 3:00:12 AM
12$GPBWR,030011,7346.710,N,08030.697,W,278,T,329,M,021.09,N,001*0E,03/09/2012
3:00:12 AM
11!AIVDO,1,1,,,14eLFt000nrE8@Db;wR;3`dB0000,0*1C,03/09/2012 3:00:12 AM
12$GPGGA,030011,7343.668,N,07916.233,W,2,12,0.69,18,M,,,1,0000*29,03/09/2012
3:00:12 AM

12$GPGLL,7343.668,N,07916.233,W,030011,A*3B,03/09/2012 3:00:12 AM
```

*Figure 4: Sample from data log using AIS Decoder.*

```
\c:1346980935,d:Connection2,s:Connection1*43\15$RARSD,,,,,,,,,,,24.0,N,N*62
\c:1346980935,d:Connection2,s:Connection1*43\12$GPGLL,6503.748,N,05404.290,W,012
216,A*3B
\c:1346980935,d:Connection2,s:Connection1*43\11!AIVDO,1,1,,,14eLFt001Kt8NgHU>b:5
gTTP0000,0*1D
\c:1346980935,d:Connection2,s:Connection1*43\12$GPRMB,A,0.00,R,,002,6440.000,N,0
5326.500,W,028.60,146,08.8,V*18
\c:1346980935,d:Connection2,s:Connection1*43\12$GPRMC,012216,A,6503.748,N,05404.
290,W,08.8,147,070912,33,W*7A
\c:1346980935,d:Connection2,s:Connection1*43\12$GPVTG,147,T,180,M,08.8,N,16.2,K*
40
\c:1346980935,d:Connection2,s:Connection1*43\14!BSVDM,1,1,,A,9t6unADaGvLD:CtVLob
T;U4hu>M0,4*4C
\c:1346980935,d:Connection2,s:Connection1*43\14!BSVDM,1,1,,A,9t6unADaGvLD:CtVLob
T;U4hu>M0,4*4C
\c:1346980935,d:Connection2,s:Connection1*43\12$GPXTE,A,A,0.00,R,N*70
\c:1346980935,d:Connection2,s:Connection1*43\12$GPZDA,012216,07,09,2012,03,00*42
\c:1346980936,d:Connection2,s:Connection1*40\11!AIVDO,1,1,,,14eLFt001Ht8NgHU>b:5
gTTP0000,0*1E
\c:1346980936,d:Connection2,s:Connection1*40\14!BSVDM,1,1,,A,9t6unADaGvLD<WpVLmI
4;U4hu>M0,4*13
\c:1346980936,d:Connection2,s:Connection1*40\13$RAOSD,146.0,A,147.0,P,8.8,P,,,N*
47
\c:1346980936,d:Connection2,s:Connection1*40\13$RARSD,0.00,,0.40,,,,,,,,12.0,N,N
*63
\c:1346980936,d:Connection2,s:Connection1*40\12$GPAPB,A,A,0.00,R,N,,,146,T,002,1
46,T,146,T*13
```

*Figure 5: Sample from data log using NMEA Router.*

The PC clock on the secondary laptop was updated hourly through a dedicated GPS time server and the application Tac32Plus [16]. The Tac32Plus software did fail on two occasions and the PC's clock varied by as many as 59 seconds during the trial.

### 3.2.4 Data Decimation

The suite of sensors feeding the SEDNA VDR produces a large amount of data. For example, the GPS receiver is outputting several sentences per second. The AIS and RADARs are also outputting messages every one or two seconds. From a tactical perspective data generated at this frequency is required to make best use of the sensors. However, from an operational perspective a subset of these data is sufficient to relay the desired information to a central operations centre. Available bandwidth is another reason to decimate the incoming data.

The data decimation algorithm uses a set of parameters to check for changes in contact status that warrant forwarding an update. What follows is a brief overview of the decimation parameters used by the script for Q346. The full annotated code is included as Annex A.

Each forwarded message is given a message number to help measure the amount of lost messages and is appended with a UNIX epoch timestamp. For the GPS RMC position messages, every tenth message was forwarded. This translates to an update of the own-ship parameters every ten seconds.

For AIS, all messages that were not regular Class A, basestation or SAR aircraft position reports (AIS types 1-4, 9), were automatically forwarded. This includes all Class B messages. The type 4, 20 and 22 basestation messages as well as type 15 interrogation messages were also not forwarded. Class A and SAR aircraft position reports (types 1-3, 9) were forwarded if: 1) it was the first received from a given MMSI or 2) the vessel's speed has changed by 2 knots or more or 3) the vessel's course has changed by 15 degrees or more or 4) the vessel has moved by 5 nautical miles (NM) or more or 5) no messages have been forwarded from this MMSI in the last ten minutes or 6) the vessel is reporting a suspect or invalid MMSI, course or speed. A MMSI was only deemed to be invalid if it exceeded the nine digit maximum for MMSIs. While the Saab R4 transponder was in use, the 10C message containing the number of currently received contacts was forwarded every ten minutes.

ADS-B messages were essentially treated the same as AIS messages while RADAR TTM messages had the same parameters for changes in course and speed as AIS messages and a minimum reporting interval of five minutes instead of ten. The first forwarded message from a RADAR contact is the first message in which the ARPA reports that it is tracking the contact and not merely querying it. During the querying period, the reported parameters can vary greatly as a precise fix on the target is acquired. As RADAR TTM messages only contain a relative position, whenever one is forwarded, the most recently received RMC message is also forwarded. This allows for the calculation of the contact's true position.

### 3.2.5 Data Backhaul

#### 3.2.5.1 Military Iridium Gateway

The same Python script used to receive and decimate the data also handled the two data backhaul procedures. The primary means of forwarding data for Q346 was via an Iridium 9522B L-band Transceiver using a military SIM card. This means that the data is down-linked through the satellite gateway in Hawaii, versus Arizona as is the case with the commercial service.

This military communications pathway was chosen for its flat-rate billing service as opposed to the usage-based billing of the commercial SIM cards. Messages that pass the screening and decimation are forwarded as UDP packets to a PC on the R&D segment of the DREnet with a static IP address. This is achieved using Iridium's Direct Internet service in which the Iridium LBT is given an IP address and essentially connected to the open internet, albeit it with an extremely limited data rate of 2.4 kB/s.

A PPP link is used for the higher-level modem handshaking and connection procedures. During testing it was discovered that the Direct Internet IP connection can go down while the PPP link between the modem and satellite is still active with no indication that messages are no longer being sent out.

With an Internet connection, a choice of transfer protocols was available. TCP is the preferred protocol but unresolved connectivity issues and limitations with the Python programming language used as well as time constraints led to the decision to use UDP. UDP does not guarantee delivery however using UDP, the script was better able to handle and automatically recover from connection problems. It was decided that the expected low amount of lost messages was worth the more stable operation that UDP offered. As a means of mitigation, periodic pinging of a website was performed after sending every ten messages to ensure a live connection. If the check failed, the connection was terminated then restarted with the sending of the last ten messages.

At the receiving end, the messages were logged. Periodically recently received data were converted for display on Google Earth using a Python script to create comma separated value files and GPS Babel to convert those to a Google KML file.

#### 3.2.5.2 Satellite AIS

A secondary mode of data forwarding uses an AIS200PAIS transponder to send binary type 8 messages. Using the planned route for QUEST during Q346 and the Northern Situational Awareness Simulator developed at DRDC Atlantic, transmission windows were calculated for the three extant satellites of the exactEarth constellation. This method of data forwarding was intended as an experimental proof-of-concept and as a test of the Simulator rather than an effective means of transferring data. There is no handshake process to establish a connection to these satellites. Messages are sent over AIS frequencies during the transmission windows with the hope that they will be received and decoded by the satellite system. AIS satellites in low-Earth orbit suffer from high instances of transmission slot collision as their footprints extend far beyond the typical transmission range of AIS on the surface and distant ships may transmit over each other. However, low traffic density in the Arctic and high revisit rates by polar-orbiting satellites made Q346 the ideal opportunity for this experiment. Annex B describes the format of the application-specific type 8 messages. There were four message types used to send QUEST's GPS position as well as AIS, ADS-B and RADAR contact reports.

In addition to these type 8 messages, the AIS200P transponder was also used to send the type 15 interrogation message to vessels in range. Upon receipt of a type 15, the receiving ship responds with a type 3 position report. The goal was to determine if these added position reports would increase the likelihood of their detection in space and improve the resolution of vessel tracks in the exactEarth feed.

# 4 Results and Analysis

## 4.1 Data Backhaul via Military Iridium Gateway

### 4.1.1 Iridium LBT Operation

The results of the SEDNA VDR trial as part of Q346 were influenced by a number of factors related to equipment and personnel. The most important factor was the operation of the military Iridium backhaul mechanism. It was determined during the first week of the trial that the Iridium transmissions from the SEDNA VDR Iridium LBT were interfering with an ocean mapping system called Olex, operated by Canadian Hydrographic Services (CHS) [17]. The Olex system is used for acquiring bathymetric data for charting and requires an Iridium satellite connection to receive GPS corrections.

It was decided by the QUEST lab supervisor that the SEDNA VDR's Iridium antenna would be moved to the stern of the ship using an extension RF cable, away from the Olex antenna. Unfortunately, the Iridium LBT was damaged during the relocation and the backhaul mechanism was rendered inoperable.

Two RF cables connect to the Iridium LBT, one for the Iridium satellite antenna and one for a GPS antenna. In the original installation configuration these cables were 20 metre lengths of LMR 400 cabling with SMA connectors. The small size of the SMA connectors combined with the stiff cabling presented the risk of damage if the cables or the LBT itself were moved so as to put stress on the connections. One cable connector was broken off of the Iridium LBT and it would be 33 days before it could be replaced and the installation re-configured to mitigate the risk of the incident reoccurring. Figure 6 is a photograph of the damaged LBT. For the last week of the trial, the SEDNA VDR Iridium antenna was returned to its original location when the Olex system was no longer in use.

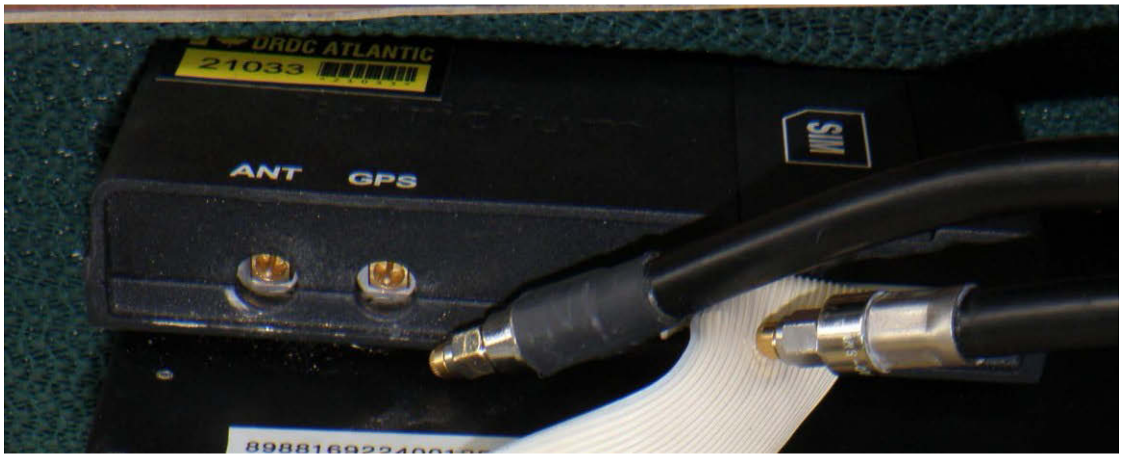

*Figure 6: Photo of the Iridium LBT with two broken connectors. Pieces of the connectors (gold in colour) are visible still attached to the antenna cables.*

The trial may be divided into four operation phases where the backhaul mechanism is concerned:

- The first is the period from 16 – 22 July 2012 when the backhaul mechanism functioned essentially continuously with some interruptions for eliminating software bugs.
- The second phase is from 23 July to 24 August when the backhaul mechanism was inoperative due to the damaged Iridium LBT.
- The third phase was from 25 August to 8 September when the backhaul mechanism operated with the new Iridium LBT installation configuration and with the Iridium antenna placed at the stern of the ship using an extended RF cable.
- The fourth and final phase is from 9 to 14 September when the antenna was returned to its original location.

The difference between the first and fourth phases is configuration of the Iridium LBT connections. The LBT was placed inside a metal housing with mounted N-Type to SMA couplers. The RF cables from the GPS and Iridium antennas were then connected to the N-type connectors via a short length of more flexible, two metre length of LMR 195, SMA to N-Type cable. This configuration alleviated the stress on the LBT connectors at the expense of signal loss. The approximately 20 metre length of cabling used to facilitate the relocation of the Iridium antenna also introduced cable losses. In effect each of the three operational phases of the backhaul mechanism experiment suffered differing amounts of cable loss. The first phase suffered the fewest losses while the third phase, using the new antenna location and new installation configuration, suffered the most.

For the final phase, when the 20 metre extension cable was removed and the antenna returned to its original location, the cable losses were reduced but were still greater than the first phase of the trial.

Table 3 lists the approximate signal losses of the backhaul mechanism relative to the initial configuration. The effect of the differing signal losses becomes apparent when examining the data transmission success rates during the four phases.

*Table 3: Relative cable losses of the different hardware configurations.*

| Phase | Signal Losses |
|---|---|
| 1. 16 – 22 July<br>(Initial configuration) | 0 dB |
| 2. 23 July – 24 August<br>(Inoperable) | - |
| 3. 25 August – 8 September<br>(Extended, reconfigured LBT connection) | 4.25 dB[a] |
| 4. 9 – 14 September<br>(Reconfigured LBT connection) | 1.4 dB[b] |

a) Calculated as the sum of added component insertion losses: (20m LMR400 = 2.7 dB, TNC-TNC adapter ≈ 0.15 dB, 2m LMR195 = 0.9 dB, SMA-SMA adapter ≈ 0.1 dB, N-Type-SMA adapter ≈ 0.4 dB).

b) Calculated as the sum of added component insertion losses: (2m LMR195 = 0.9 dB, SMA-SMA adapter ≈ 0.1 dB, N-Type-SMA adapter ≈ 0.4 dB).

### 4.1.2 Results

#### 4.1.2.1 Data Transfer

Table 4 and Figure 7 show the ratio of messages successfully transferred on the first attempt (initial success ratio) as well as the ratio of messages successfully transferred when taking into account messages that were re-sent after a lost connection was re-established (final success ratio). An independent analysis was conducted to determine the transfer ratios [18]. The results of that analysis agree with the values in Figure 7 but varied slightly. This was due to the specific file-structure of the data logs and the code used to do the analysis.

*Table 4: Daily and total Iridium Backhaul success rates.*

| Date | Forwarded Messages | Backhauled Messages | Initial Success Ratio | Recovered Messages | Final Success |
|---|---|---|---|---|---|
| 16-Jul | 15728 | 15598 | 0.9917 | 51 | 0.995 |
| 17-Jul | 2732 | 2689 | 0.9843 | 19 | 0.9912 |
| 18-Jul | 5781 | 5636 | 0.9749 | 44 | 0.9825 |
| 19-Jul | 9595 | 9454 | 0.9853 | 103 | 0.996 |
| 20-Jul | 7488 | 7389 | 0.9868 | 76 | 0.997 |
| 21-Jul | 8216 | 8086 | 0.9842 | 121 | 0.9989 |
| 22-Jul | 3881 | 3839 | 0.9892 | 37 | 0.9987 |
| ***Phase 1*** | *53421* | *52691* | *0.986335* | *451* | *0.994777* |
| 25-Aug | 105 | 86 | 0.819 | 11 | 0.9238 |
| 26-Aug | 413 | 337 | 0.816 | 48 | 0.9322 |
| 27-Aug | 702 | 536 | 0.7635 | 150 | 0.9772 |
| 28-Aug | 771 | 666 | 0.8638 | 93 | 0.9844 |
| 29-Aug | 1064 | 912 | 0.8571 | 137 | 0.9859 |
| 30-Aug | 1009 | 811 | 0.8038 | 182 | 0.9841 |
| 31-Aug | 830 | 712 | 0.8578 | 105 | 0.9843 |
| 1-Sep | 258 | 256 | 0.9922 | 0 | 0.9922 |
| 4-Sep | 11 | 11 | 1 | 0 | 1 |
| 7-Sep | 24 | 16 | 0.6667 | 0 | 0.6667 |
| ***Phase 3*** | *5187* | *4343* | *0.837286* | *726* | *0.977251* |
| 9-Sep | 2451 | 2300 | 0.9384 | 127 | 0.9902 |
| 10-Sep | 4861 | 4557 | 0.9375 | 252 | 0.9893 |
| 11-Sep | 5109 | 4900 | 0.9591 | 146 | 0.9877 |
| 12-Sep | 7450 | 7105 | 0.9537 | 232 | 0.9848 |
| 13-Sep | 6890 | 6594 | 0.957 | 234 | 0.991 |
| 14-Sep | 4930 | 4812 | 0.976 | 82 | 0.9927 |
| ***Phase 4*** | *31691* | *30268* | *0.955098* | *1073* | *0.988956* |
| **Total** | **90299** | **87307** | **0.9669** | **2250** | **0.9918** |

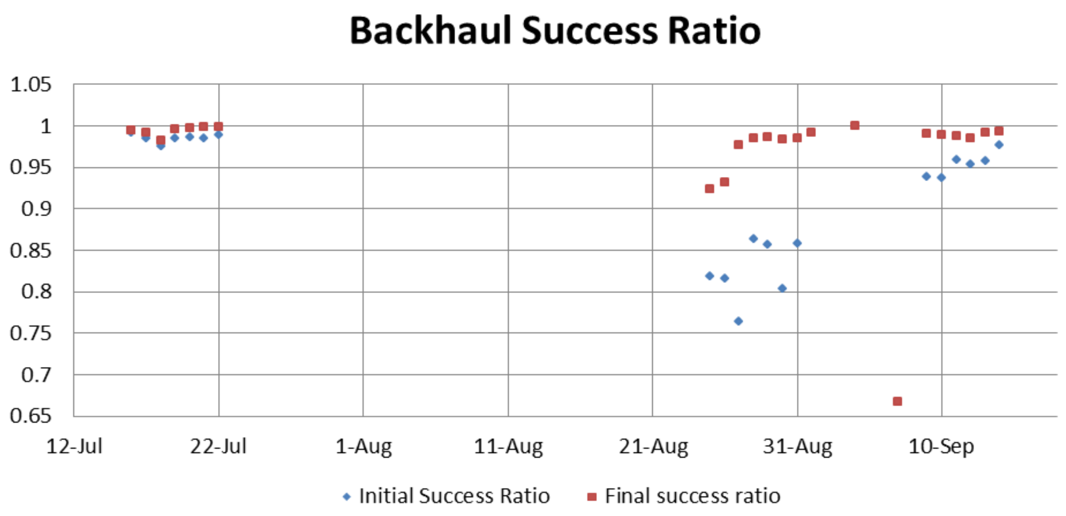

*Figure 7: Daily backhaul success ratio.*

The ratio of messages successfully transferred on the first attempt (Table 4, Initial success ratio) strongly correlates with the cable losses in the backhaul mechanism hardware. However, regardless of the configuration, the re-sending procedure was effective in recovering most of the lost messages. Because the UDP protocol was used and it does not use transmission receipt acknowledgments, it was possible for messages to be lost without detection by the controlling software.

It is possible for a transmission to have been lost somewhere in the chain between the Iridium antenna on QUEST, the Iridium satellite, the satellite ground station and the DRDC server. As long the connection was still active at the next check, the lost message would go undetected. This was found to occur with 0.82% of messages showing that the UDP protocol is very reliable despite its lack of transmission receipt acknowledgments.

#### 4.1.2.2 Connection temporal statistics

Table 5 and Figure 8 show the temporal statistics of the Iridium link. Again, there is a strong correlation with the cable losses in the four phases of operation. Connections were dropped even in the most ideal configuration with the fewest cable losses. This can be due to non-graceful hand-offs of the data calls between satellites, a known issue with satellite communications [19]. The Iridium constellation is purported to have global coverage but with reduced revisit rates at the extreme North and South latitudes. Due to the timing and nature of the four phases of the trial, the range of latitudes transited by QUEST was not covered by any single system configuration. This makes it difficult to draw any conclusions concerning the performance of the Iridium backhaul mechanism at the different latitudes. While Iridium employs a constellation of sixty-six polar-orbiting satellites, the frequency of dropped calls is known to be higher at higher latitudes [20]. The third phase of operation covered the largest proportion, from 74°N to 64°N and no significant change in connection time was observed. The same is true of the first phase covering 45°N to 64°N.

*Table 5: Daily and average connection temporal statistics.*

| Date | Number of Connections | Mean Duration (s) | Total connection time (hrs) |
|---|---|---|---|
| 16-Jul | 17 | 3515 | 16.6 |
| 17-Jul | 10 | 3661.1 | 10.17 |
| 18-Jul | 16 | 3052.5 | 13.56 |
| 19-Jul | 29 | 2527.93 | 20.37 |
| 20-Jul | 25 | 3123.24 | 21.69 |
| 21-Jul | 34 | 2340.74 | 22.11 |
| 22-Jul | 17 | 2139.65 | 10.1 |
| *Phase 1* | *148* | *2787.542* | *114.6* |
| 25-Aug | 5 | 347.2 | 0.48 |
| 26-Aug | 23 | 259.04 | 1.65 |
| 27-Aug | 39 | 257.51 | 2.79 |
| 28-Aug | 31 | 311.48 | 2.68 |
| 29-Aug | 29 | 313.24 | 2.52 |
| 30-Aug | 26 | 328.85 | 2.38 |
| 31-Aug | 24 | 163.79 | 1.09 |
| 1-Sep | 3 | 283 | 0.23 |
| 4-Sep | 1 | 21 | 0.01 |
| 7-Sep | 1 | 71 | 0.02 |
| *Phase 3* | *182* | *274.1687* | *13.85* |
| 9-Sep | 33 | 1002.39 | 475 |
| 10-Sep | 64 | 1093.7 | 575 |
| 11-Sep | 53 | 1288.98 | 750 |
| 12-Sep | 66 | 1086.55 | 884 |
| 13-Sep | 65 | 983.8 | 568 |
| 14-Sep | 23 | 1089.95 | 583 |
| *Phase 4* | 304 | 1092.499 | 220.72 |
| **Total** | **634** | **1272.25** | **2250** |

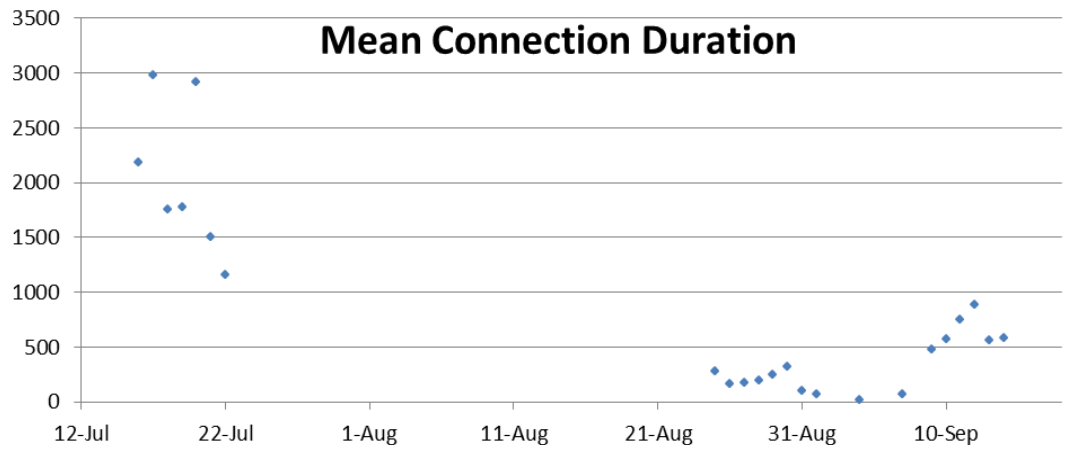

*Figure 8: Daily mean connection duration in seconds.*

### 4.1.2.3 Data Display

Data received from QUEST on the DRDC server were logged into daily files. A script written in Python was used to convert the incoming data into CSV (comma separated values) files which were in turn converted to Google Keyhole Markup Language (KML) files for display on Google Earth (see Figure 9). Short animations showing QUEST's daily track along with the AIS, ADS-B and RADAR contacts it received were displayed on a large monitor for viewing by DRDC Atlantic personnel.

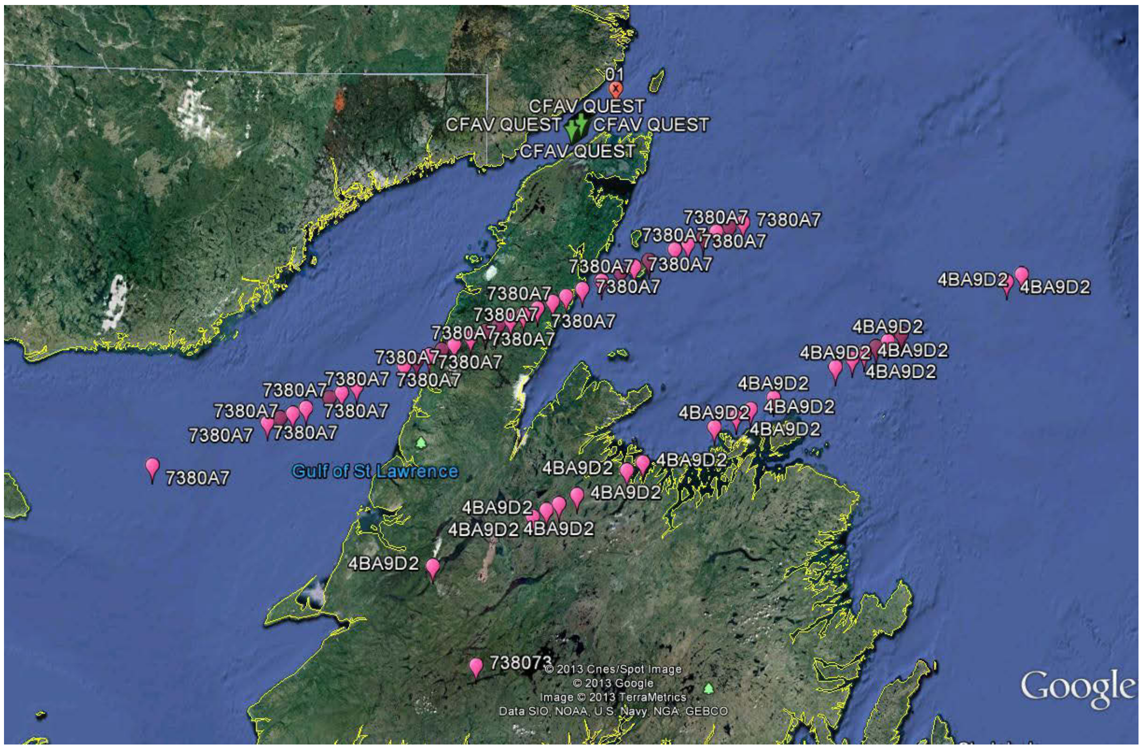

*Figure 9: Screenshot of Google Earth display from 18 July 2013. Pink balloon markers represent ADS-B contacts, red represent RADAR contacts and green arrows represent CFAV QUEST.*

## 4.2 AIS Performance

Another significant factor affecting the results of the trial was the performance of QUEST's AIS transponder. It was noted by the ship's crew that during Q346, the ship's AIS suffered low reception ranges. Prior to Q346, QUEST typically received AIS messages in the 25 to 40 nautical mile range and sometimes as far as 80 miles with the latter cases likely due to atmospheric ducting. Indeed, using the following formula, line-of-sight for two vessels with AIS antennas mounted at 13 metres (the approximate height of QUEST's AIS antenna) is 16 nautical miles.

$$d \approx 4.12 \times h^{1/2}; \text{[21]} \tag{2}$$

In this formula, d is line-of-sight distance in km, h is the height of antenna in metres and 1 nautical mile equals 1.852km. In practice this range is typically somewhat greater as radio signals tend to be refracted toward the Earth by the atmosphere. In extreme weather this range can be reduced however QUEST did not encounter any such weather during Q346. QUEST should expect to detect large commercial vessels with antennas mounted at 30 metres at a minimum of 20 nautical miles. During Q346, QUEST rarely received its first transmission from as far as 20 nautical miles.

The average maximum range of detection for vessels was 14.7 nautical miles with several vessels coming within one nautical mile before detection with AIS. Vessels were regularly received by RADAR before AIS, a reversal of the normal situation. This poor performance made it impossible to evaluate the performance of AIS in the Arctic environment.

On 11 August, the ship's Saab R4 AIS transponder was switched out for an L-3 PROTEC transponder. The antenna, antenna cable and antenna location were not changed. No improvement in AIS reception range was observed. As such, the cause of the poor performance was not determined. The location of the AIS antenna on Monkey's Island (the deck immediately above the bridge) may have played a role as this area hosted a myriad of new antennas for the trial. Some RF interference had already been observed between the SEDNA VDR's Iridium antenna and the Olex system's Iridium antenna.

Overall QUEST detected 113 unique AIS contacts including 19 Coast Guard basestations. The vast majority were detected when QUEST was in proximity to land, especially in the approaches and harbours of Halifax and Nuuk, Greenland. North of the Arctic Circle, QUEST received 11 vessels on AIS. QUEST was north of the Arctic Circle for 44 days from 24 July to 6 September.

QUEST did receive the occasional vessel at ranges beyond line-of-sight, likely due to atmospheric ducting. The maximum range for an AIS vessel was 123 nautical miles, nearly twice as distant as the next nearest contact detected at 65 miles. Of the 16 contacts detected at greater than 20 nautical miles, 11 were detected on 13 September while QUEST transited the Gulf of St. Lawrence suggesting that atmospheric conditions on that day were very conducive to VHF propagation.

Ranges for the basestations were often much larger as expected due to the elevation of basestation antennas. Four basestations were received at over 100 nautical miles with the maximum being the Fortune, NL site at 179.7NM. All basestation detections over 60NM as well as the maximum detection range for each basestation occurred during the favourable conditions of 13 September. This does not suggest however that 13 September was the only day with such favourable VHF propagation conditions. Other similar days would have gone unnoticed if no AIS contacts or basestations were present at the appropriate ranges. This was the case for the majority of the period between stops in Nuuk, a period which constitutes the majority of the trial. However, it does suggest that conditions were more favorable in the Gulf of St Lawrence on 13 September than they were when QUEST first transited the area on 17 July. Figure 10 displays the maximum reception range for each received AIS contact.

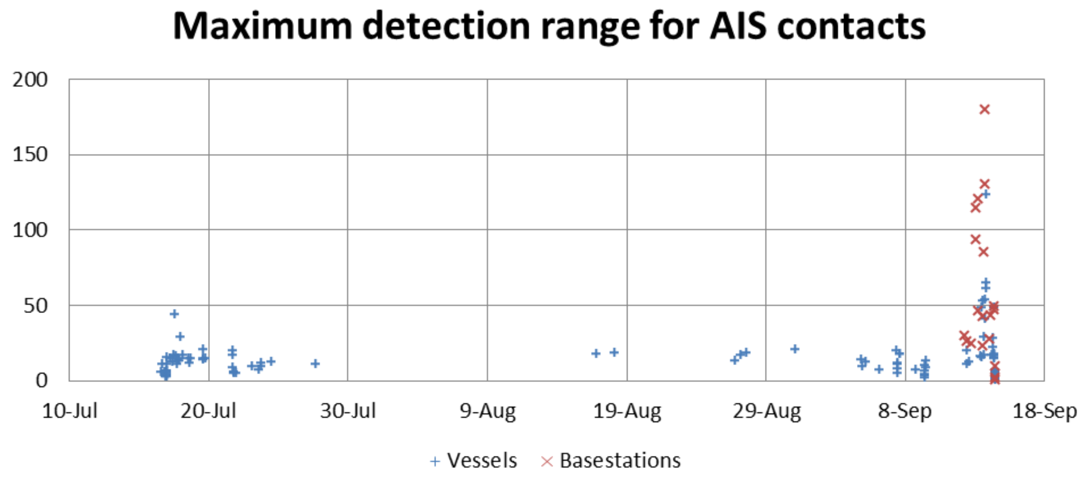

*Figure 10: Maximum detection ranges in nautical miles for AIS contacts.*

## 4.3 ADS-B Performance

The ADS-B receiver installed on QUEST for the trial performed well. The conversion to NMEA standard messages for ingestion into the VDR worked as designed and dozens of aircraft position reports were logged and backhauled to DRDC Atlantic. Reception ranges were often near 200 nautical miles which is expected for aircraft flying at or near 13000 metres. These ranges varied significantly as proximity to elevated geography and buildings often placed QUEST in dark regions where ADS-B reception would be blocked. Noticeable dips in reception range due to geography occurred while QUEST was in Nuuk Harbour from 22–23 July and 7–9 September. Some ducting was observed and aircraft were received at up to 830 nautical miles.

All of the 11 aircraft detected at ranges greater than 250 nautical miles were detected over a 60 hour period from 18 to 20 July. In total 1367 unique aircraft were observed with the majority detected on multiple flights across the Atlantic during Q346. North of the Arctic Circle the number of detected aircraft decreased but was substantial. However, the script used to convert ADS-B messages into NMEA standard messages failed on August 11 for reasons unknown and was not restarted until August 26 after the arrival of the MIS engineer. Figure 11 displays the range of the first and last received messages from each unique ADS-B contact.

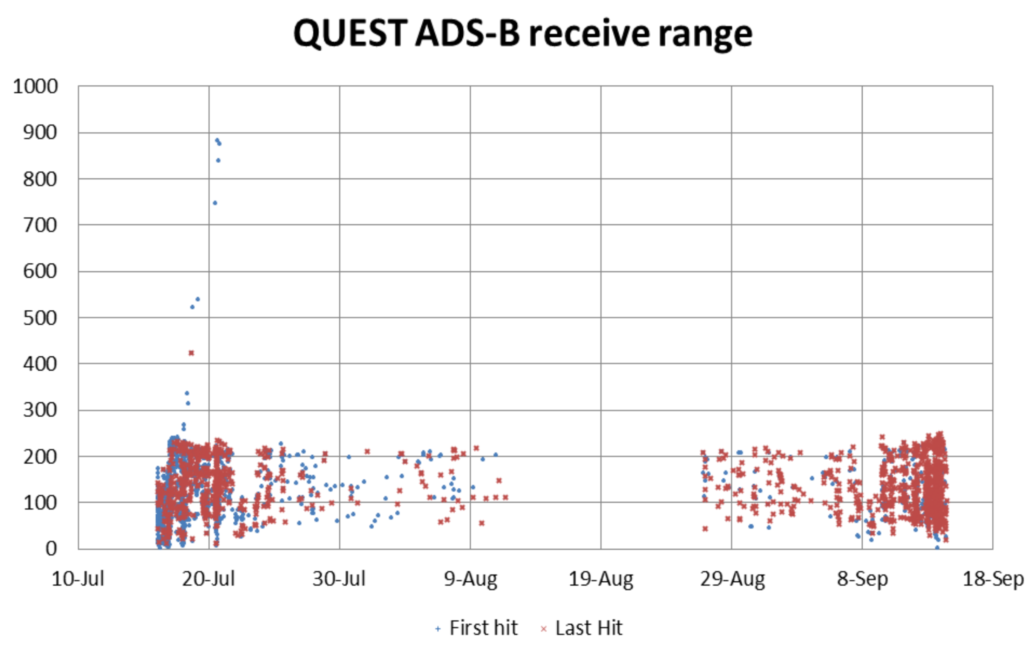

*Figure 11: Range, in nautical miles, of first and last received messages from each unique aircraft.*

## 4.4 Northern Situational Awareness Simulator Validation

Another objective of the trial was to validate an aspect of the Northern Situational Awareness Simulator. The Simulator was used to calculate the periods of time that the satellites used by exactEarth would be in view of QUEST. For the seven day period ending on 17 July 2012, while QUEST was in Halifax, the messages received from QUEST were logged and their timestamps compared to those predicted by the simulator.

Of the 854 messages received during that the seven-day period, 833 (98%) fell within a predicted window of visibility. All of the 21 messages that fell outside the window came from the same exactEarth satellite, EV03, and all of them came within the ten minutes preceding a predicted window. As well, no messages were received in the last ten minutes of the predicted window for EV03.

This suggests that the predicted orbital path itself was correct and only its predicted position on that path erroneous. Thus the predicted satellite visibility window for that satellite was temporally shifted by ten minutes for the remainder of the trial to correct the error. The cause of the misaligned satellite window is unknown but may be due to the use of out-of-date ephemeris satellite data. The orbital characteristics of satellites are relatively stable and predictable but the Simulator must periodically update the ephemeris data from online resources as new data are published.

## 4.5 AIS-Related Experiments

### 4.5.1 Type 8 Binary Messages

The AIS-related experiments were largely unsuccessful. For the vast majority of the trial, no special type 8 binary messages were transmitted by the secondary AIS.

For the first phase of the trial, the backhaul mechanism worked well. However no type 8 messages were transmitted. The reason for this is unknown. As the secondary AIS transponder was transmitting its regular position reports during this time, it is likely that the LAN connection between the VDR and the secondary AIS transponder was not functioning properly.

During the second phase of the trial, the backhaul mechanism was not functioning. As a result, the transmission of the type 8 messages was also not functioning. This is because the Iridium connection and type 8 message transmission were controlled by the same Python script and the establishment of the Iridium connection took precedence. Thus type 8 messages could only be sent while the Iridium connection was active.

During the third phase, few type 8 messages were transmitted as the Iridium connection was erratic and was not active for long periods. During this phase, two periods of time were set aside and dedicated to transmitting the type 8 messages. Of the 150 type 8 messages received by exactEarth satellites, 121 were received during the two dedicated transmission periods. The Python script was modified so that the Iridium connection was not required in order to transmit on AIS. As well, the type 8 AIS messages were transmitted regardless of the calculated satellite windows. At an unknown time during the summer months of 2012, exactEarth brought online a fourth AIS satellite. As the pre-planned satellite windows only contained the original three satellites, the satellite windows were ignored in an attempt to maximize the amount of type 8 messages received in space.

Unfortunately, a complete list of transmitted type 8 messages is not available. A list of unique messages was logged but each message was forwarded to the secondary AIS transponder for transmission five or ten times depending on the message type. Acknowledgment messages from the secondary AIS, indicating that the request to transmit the message was received, were logged. However, there was no independent confirmation that the messages were actually transmitted. The secondary transponder would output some AIVDO messages indicating successful transmission but usually for one or two or a burst of ten requests. Similarly, the primary QUEST transponder often only recorded one or two of these and they would be the same or different ones logged by the secondary transponder. The exactEarth satellites also received some messages that were logged by neither transponder on QUEST. The end result is three lists of messages each containing some not present on the other two. As such, it is impossible to say how many messages were actually transmitted.

It is possible however, to estimate the number of transmitted type 8 messages by measuring the ratio of type 1 and 3 position messages received on exactEarth satellites from the secondary transponder. Unlike the type 8 messages, all of the regular type 1 and 3 messages from the secondary transponder were logged and a full list of messages is available. Figure 12 plots the ratio of type 1 and 3 messages received from the secondary AIS transponder per day of the trial.

The average success rate was 0.45%. This low percentage of successfully decoded messages is due in part to messages transmitted when no satellite was in view; and also the possibility that messages could not be decoded due to slot collisions [22].

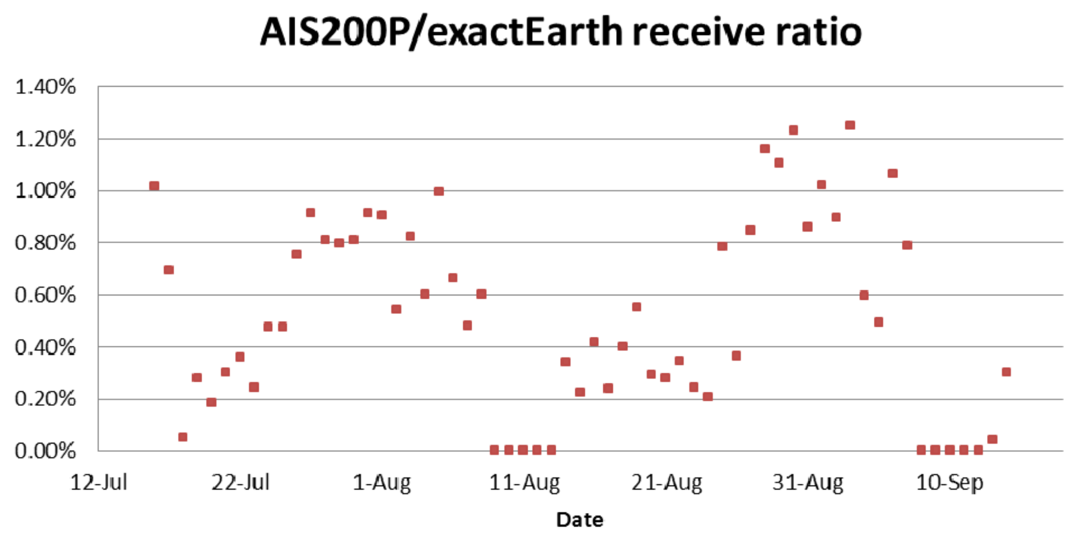

*Figure 12: Percentage of AIS200P type 1 and 3 messages received on exactEarth satellites.*

For comparison, Figure 13 shows a similar chart for QUEST's transponder, first the Saab R4 AIS transponder and the L-3 PROTEC Marine AIS after 11 August. The combined average success rate was 1.21%.

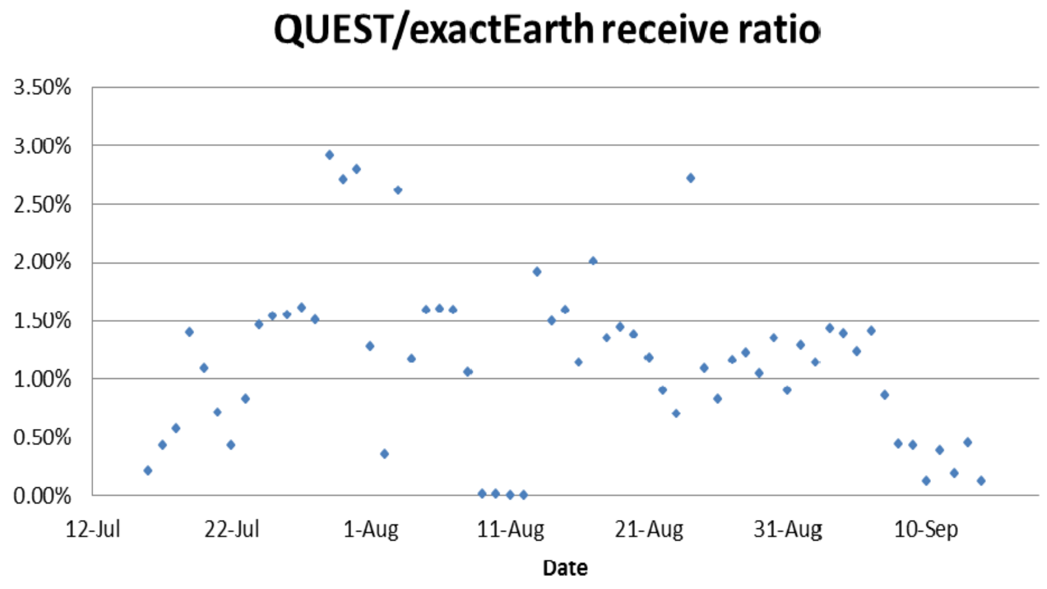

*Figure 13: Percentage of QUEST type 1 and 3 messages received on exactEarth satellites.*

There was only a slight difference in the ratio of two-slot type 5 AIS messages successfully decoded by the exactEarth satellites. It was expected that multi-slot messages would be more susceptible to slot collisions and thus reduce success of message decoding. For the QUEST AIS transponder, 1.22% of single-slot type 1 and 3 position messages were decoded while 1.08% of two-slot type 5 messages were decoded. For the secondary transponder these values were both 0.45%. Figures 14 and 15 show the reported latitude from QUEST's transmitted AIS position reports and those received on exactEarth satellites.

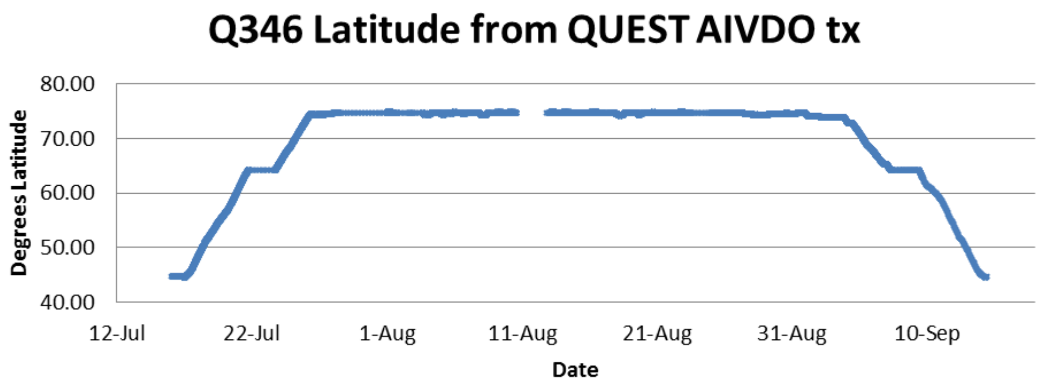

*Figure 14: Reported latitude from QUEST AIS position reports.*

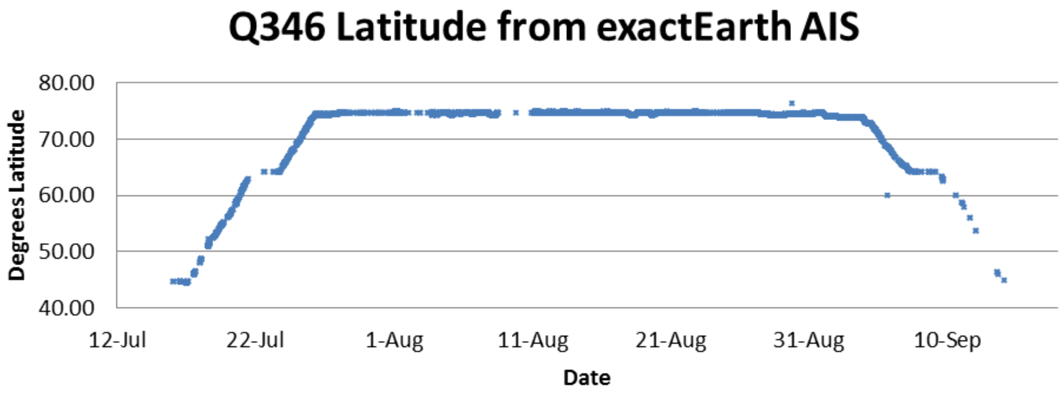

*Figure 15: Reported latitude from QUEST AIS position reports received on exactEarth satellites.*

The first period used for the type 8 message transmissions was from 3 September, 14:27 GMT to 4 September, 12:01 GMT while QUEST transited Navy Board Inlet after leaving Lancaster Sound en route to Nuuk, Greenland. During this period a total of 110 of these type 8 messages were received on exactEarth satellites and forwarded to DRDC Atlantic. Table 6 summarizes these results. All of the AIS/ADS-B contact report messages were for ADS-B aircraft as QUEST received no AIS messages during this period. All RADAR contact reports reported the location of ice. Figure 16 is a Google Earth screen shot plotting the positions indicated by the received type 8 messages while transiting Navy Board Inlet.

*Table 6: Receive success rates of type 8 messages.*

| **Message Type** | **Message length (slots)** | **Theoretical maximum number sent** | **Total received by satellite** | **Success rate** |
|---|---|---|---|---|
| Short GPS report from QUEST | 0.5 | 34860 | 41 | 0.11% |
| Long GPS Report from QUEST | 1 | 34860 | 22 | 0.06% |
| AIS/ADS-B contact report | 1 | 900 | 0 | 0.00% |
| RADAR contact Report | 1 | 107930 | 47 | 0.04% |
| Total | - | 178550 | 110 | 0.06% |

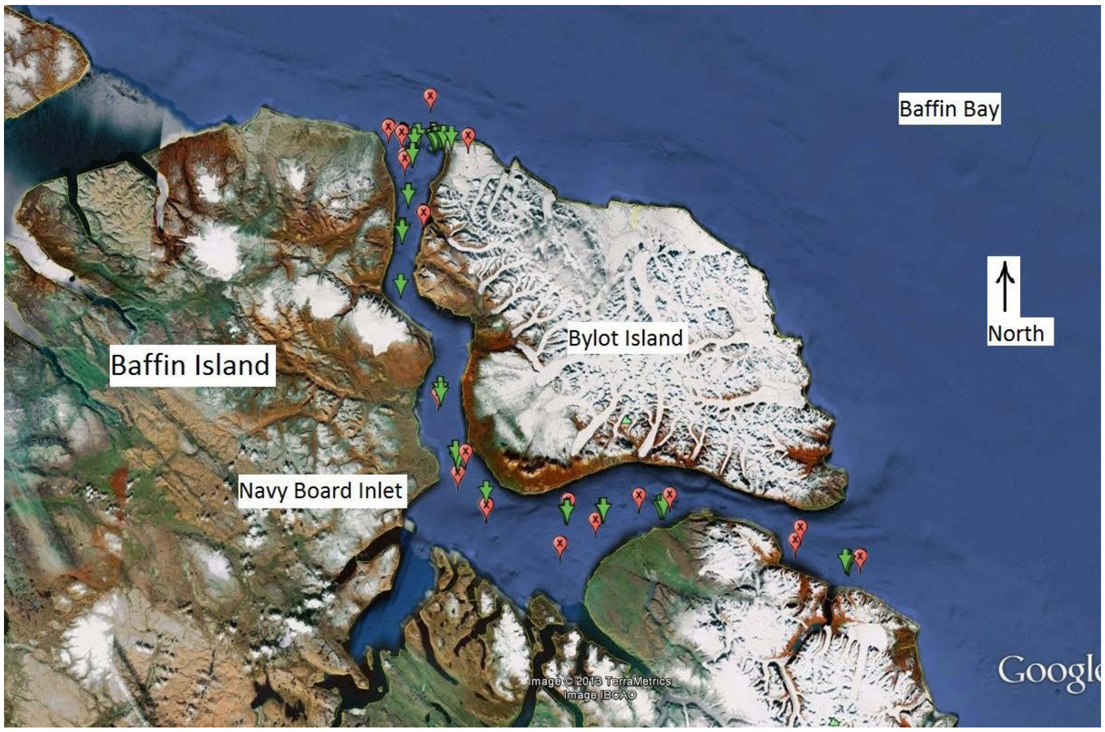

Figure 1: Google Earth plot of contact reports contained in type 8 messages received on

*Figure 16: Google Earth plot of contact reports contained in type 8 messages received on exactEarth satellites 3–4 September 201. Green arrows represent CFAV QUEST; red place marks represent RADAR contacts (ice).*

Assuming every type 8 message transmission request resulted in a transmitted type 8 message the total amount of messages sent for the 3 to 4 September timeframe was 178550. From the observed success rate of the secondary transponder, it is estimated that at least 24444 messages were transmitted during the transit of Navy Board Inlet, or about 15% of the theoretical maximum number of messages transmitted. The primary transponder on QUEST failed to receive and decode more than one or two messages out of a burst of ten. It may be reasonable to assume that the satellite receivers did the same and that the true number of messages transmitted is closer to the theoretical maximum than an estimate based on the space-based detection of type 1 and 3 position messages from the secondary transponder.

The second period dedicated to type 8 AIS message transmission was from 16:30 to 17:33 GMT on 7 September while QUEST was at anchor in Nuuk Harbour. During this time 10 messages were received containing position reports for QUEST, one aircraft and 3 AIS contacts.

Because it was known ahead of time that QUEST would be in Nuuk on 7 September, calculations of satellite visibility were performed by the Northern Situational Awareness Simulator and further refined prior 7 September. Using these calculated satellite windows it is possible to gain a better understanding of the percentage of AIS messages properly decoded as those messages transmitted without a satellite in view can be ignored. While these satellite windows were programmed into the Python script controlling the transmission, this function was disabled and messages were allowed to be sent at any time as a fourth satellite with then unknown orbital characteristics had been added to the exactEarth constellation. Figure 17 is a Google Earth screen capture showing the local operating picture derived from the received type 8 messages while QUEST was in Nuuk Harbour.

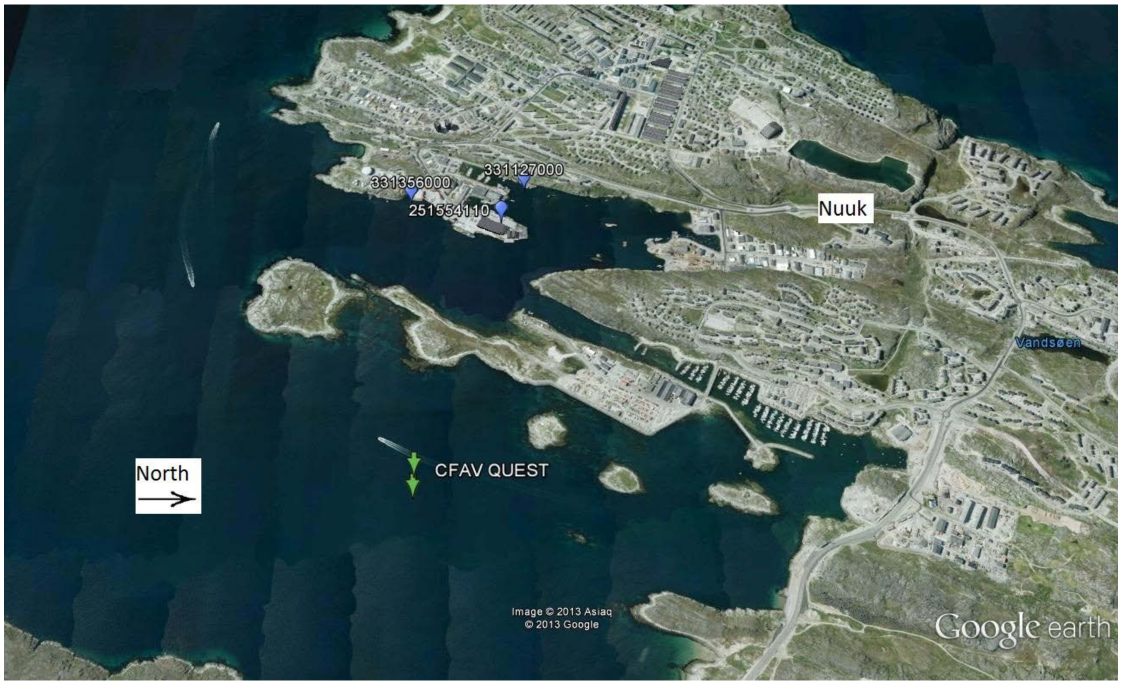

*Figure 17: Local surface picture in Nuuk Harbour, 7 September 2012, derived from backhaul type 8 messages via exactEarth.*

### 4.5.2 Type 15 Interrogation Messages

The second AIS-related experiment dealt with the transmission of type 15 interrogation messages to increase the reporting frequency of AIS contacts in range of QUEST. Due to an error in the

Python script, type 15 interrogation messages were only sent to the primary transponder on QUEST and not to all encountered AIS targets. However, this error served to reveal a false assumption regarding the transmission of type 3 messages.

According to the AIS specification the type 3 message is a special position report message that is used on two occasions: 1) upon entry into the network. A type 3 message should be the first message transmitted by an AIS transponder upon power-up and 2) in response to a type 15 interrogation [23]. However, this was not the case with QUEST's AIS transponders as neither the Saab R4 nor the L-3 Protec functioned this way. Instead, the transponders transmit a type 3 message anytime an incremental time division multiple access (ITDMA) transmission is required. These ITDMA transmissions are required when the reporting rate of an AIS transponder is less than two reports per minute and when the reporting rate needs to increase in response to a change in speed or rate of turn. Often two type 3 messages would be transmitted within the same second, even in the absence of a type 15 interrogation message. This makes it impossible to separate the type 3 messages transmitted in response to a type 15 interrogation from the ones that were not. As such, no measurement of the increase of the detection probability of QUEST by satellite-based receivers could be performed.

### 4.5.3 Low Power AIS and exactEarth Satellite Reception

The AIS transponder was placed in low power mode to avoid damaging the onboard sonobuoy receiver that operates in the same VHF band as AIS. There was no noticeable difference in the satellite receive success rate when the secondary AIS transponder operated in low power mode (2W versus 12W) during the period of 22 July to 24 August.

# 5 Summary and Conclusion

## 5.1 Results Summary

Overall, Q346 was a successful trial of the SEDNA VDR system. Despite numerous unforeseen incidents, the system performed very well when used in the designed configuration. The military Iridium backhaul mechanism was extremely effective during the first phase of the trial. The use of UDP packets combined with the procedure of re-sending lost messages did well to improve the success rates for data transfer when the Iridium connection was hampered by cable losses.

The trials offered the first successful validation of the Northern Situational Awareness Simulator by showing its satellite orbit predictions were accurate. One satellite window was shown to be slightly misaligned in time and this could be due to out-of-date ephemeris data used by the simulator at the time of calculation.

The AIS-related experiments were not as successful.

- Using the exactEarth AIS satellites to pass information via type 8 messages was found to be a technically feasible means of communication though unreliable and unsuitable for most applications.
- Ambiguities surrounding the type 3 messages made it impossible to measure the efficacy of the interrogation message to increase vessel detection via AIS satellites.

## 5.2 Future Considerations

Given the results of Q346 there are some potential enhancements that would make the SEDNA VDR a more effective and capable system. The primary enhancement would be the development of two-way communication using the military Iridium link. The ability to change settings and request status reports from a remote location would be a significant improvement. Parameters related to the decimation algorithm could be modified to increase or decrease the reporting rate. Also, a list of vessel of interest requiring extra reports could be sent to the VDR.

It may also be desirable to forward contacts out of the vessel's sensor range to the vessel to be displayed on a system such as ECPINS. The information could be especially useful in the Arctic or other remote areas should a significant incident occur at sea. During Q346, CFAV QUEST's transit was hindered by ice and was only made possible with the help of the near-by, but out of sensor range ice-breaker CCGS Louis St-Laurent.

The hardware requirements of the three Windows-based PCs could be reduced to a single machine, ideally using the PC inside the Rutter VDR 100-G3 Voyage Data Recorder. Similarly, only one AIS transponder is required.

# References

[1] "Amendments to the International Aeronautical and Maritime Search and Rescue (IAMSAR) Manual", International Maritime Organization, http://www.imo.org/publications/documents/supplements%20and%20cds/english/1367.pdf (Access Date: 10 Jul. 2013).

[2] Marsh, J.H., (1988), Arctic Archipelago, *The Canadian Encyclopedia*, Toronto: Hurtig Publishers.

[3] Heard, G., et al. (2011), Underwater Sensor System 2009 Field Trial Report: Northern Watch Technology Demonstration Project, (DRDC Atlantic TM 2010-241) Defence R&D Canada – Atlantic.

[4] "Voyage Data Recorder Regulations", Canada Gazette, http://gazette.gc.ca/rp-pr/p1/2010/2010-11-06/html/reg2-eng.html (Access Date: 10 Jul. 2013)

[5] Hammond, T. (2005), A Case Study of Hibernia Illustrating the Benefits of AIS Transponders on Oilrigs, (DRDC Atlantic TM 2004-210) Defence R&D Canada – Atlantic.

[6] "VDR 100G3/G3S", Netwave Systems, http://www.netwavesystems.com/wp-content/uploads/2013/02/Rutter-G3-brochure.pdf (Access Date: 10 Jul. 2013).

[7] "Product Explorer", Analytical Graphics Inc., https://www.agi.com/product-explorer/Default.aspx (Access Date: 10 Jul. 2013).

[8] Peori, K., Thibodeau, M., Gingell, M., (2013), High Fidelity Simulated Data System Functional Design Document, (DRDC Atlantic CR 2011-089) General Dynamics Canada, Dartmouth, Nova Scotia.

[9] "NMEA", National Marine Electronics Association, http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp (Access Date: 10 Jul. 2013).

[10] "Iridium 9522B Satellite Transceiver", Iridium Communications Inc., http://www.iridium.com/products/Iridium9522SatelliteTransceiver.aspx.

[11] "AIS 200 P, portable Automatic Identification System – Kongsberg Maritime", Kongsberg Maritime, http://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/390183F91A2B52A9C12572220055E4CC (Access Date: 10 Jul. 2013).

[12] "Satellite AIS", exactEarth, http://www.exactearth.com/products/exactais/ (Access Date: 10 Jul. 2013).

[13] “BaseStation SBS-1eR, SBS-1e and SBS-1 Reference Manual”, Kinetic Avionics, http://www.kinetic.co.uk/DownloadFiles/Assorted/SBS1-eR-ReferenceManual.pdf (Access Date: 10 Jul. 2013).

[14] International Telecommunications Union, Recommendation M.1371-4, April 2010, “Technical Characteristics for an Automatic Identification System using Time-division Multiple Access in the VHF Maritime Band”, Table 44, Annex 8.

[15] “AIS Decoder, NMEA Router, ShipPlotter.XLS by Neal Arundale”, Neal Arundale, http://web.arundale.co.uk/docs/ais/ais_decoder_v3_downloads.html (Access Date: 10 Jul. 2013).

[16] “Tac32Plus Software for the CNS Clock”, CNS Systems Inc., http://www.cnssys.com/cnsclock/Tac32PlusSoftware.php (Access Date: 10 Jul. 2013).

[17] “Olex: Main Page”, Olex AS, http://www.olex.no/index_e.html (Access Date: 10 Jul. 2013).

[18] Radulescu, D. and Hadzagic, M. (2013), Q346 Maritime Domain Awareness Data Processing, (DRDC Atlantic CR 2013-075), OODA Technologies Inc., Montreal, Quebec.

[19] Tom, C. and Wagner, L. (1999), Evaluation of Iridium Satellite Phone Voice Services for Military Applications, (DREO-TM-1999-086), Defence Research Establishment Ottawa.

[20] Frost & Sullivan (2010), Satellite Phone Comparison Iridium and Inmarsat, *Frost & Sullivan White Papers*, p.12. www.frost.com

[21] Bakshi, U. and Bakshi, A.V. (2009), Antenna and Wave Propagation – First Edition 2009, Pune: Technical Publications, p. 6–26.

[22] Burzigotti, P., Ginesi, A. and Colavolpe, G., (2012), Advanced receiver design for satellite-based automatic identification system signal detection, *International Journal of Satellite Communications and Networking*, 30, 52–63.

[23] International Telecommunications Union, Recommendation M.1371-4, April 2010, “Technical Characteristics for an Automatic Identification System using Time-division Multiple Access in the VHF Maritime Band”, Table 43, p.96.

[24] International Electrotechnical Commission – International Standard IEC 61162-1, “Maritime navigation and radiocommunication equipment and systems – Digital Interfaces – Part 1: Single talker and multiple listeners”, Third Edition, April 2007.

# Annex A    Python Code

## A.1    Data Decimation and Forwarding Script

NOTE: The Python code listing here is NOT properly indented.

```
import urllib2
from httplib import BadStatusLine
from socket import *
from time import *
import math
import socket
import os
from subprocess import call
from time import strftime

#initialize variables
curr_rmc = ''
curr_rmc_time = 0.0
gps_count = -1
dec_aivdm = ''
ais_filter = ('1','2','3','4','5','9','D','F','?')
ais_1_2_3 = ('1','2','3','9')
time1 = time()
timestamp = time()
ship_table = {}
RADAR_table1 = {}
RADAR_count1 = 0
RADAR_table2 = {}
RADAR_count2 = 0
ais_contacts = 0
ais_contacts_countA = -1
ais_contacts_countB = -1
message_count = -1

udp_host = ""
udp_port = 1234
udp_buf = 1024
```

```
udp_addr = (udp_host,udp_port)
UDPSock = socket.socket(AF_INET,SOCK_DGRAM)
UDPSock.bind(udp_addr)

online = 0
host = '128.43.1.154'
port = 43732

ais_HOST = '192.168.0.200'    # The remote host
ais_PORT = 4712              # The same port as used by the server
sais = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sais.connect((ais_HOST, ais_PORT))

not_sent = set()

def resend(not_sent):
    for string in not_sent:
        time1 = str(math.trunc(time()))
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.sendto((string), (host,port))
        sock.close()
        print ('re-sent'+string+time1+'\r\n')
        date2 = strftime('%d%b%Y')
        log_file = open("fwd_messages_"+date2+".txt", "a")
        log_file.write('re-sent'+string+time1+'\r\n')
        log_file.close()
    return

def internet_test(test_sock):
    loop_value = 1
    resend_status = 0
    while (loop_value == 1):
        try:
            urllib2.urlopen("http://www.google.com")
        except urllib2.URLError, e:
            print 'internet connection down...reconnecting...'
            date1 = strftime('%d%b%Y')
            time1 = str(math.trunc(time()))
            log_file = open("fwd_messages_"+date1+".txt", "a")
            log_file.write(time1+'internet connection lost...reconnecting\r\n')
            log_file.close()
            #test_sock.close()
            connect(0)
            resend_status = 1
        except BadStatusLine, e2:
            print 'the unknown error has occured...reconnecting...'
            #test_sock.close()
            connect(0)
            resend_status = 1
        else:
            print 'still connected'
            loop_value = 0
    return resend_status

def connect(status):

    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    connected = Iridium_connect("rasdial.exe")
    sleep(15)
    loop_value1 = 1

    while (loop_value1 == 1):
        try:
            urllib2.urlopen("http://www.google.com")
        except urllib2.URLError, e:
            print 'Iridium connection failed...reconnecting...'
            connect(0)
        except BadStatusLine, e2:
            print 'the unknown error has occured...reconnecting...'
            #test_sock.close()
            connect(0)
            resend_status = 1
        else:
            print 'Connected'
            loop_value1 = 0
    return s


def Iridium_connect(command):
    call("rasdial" + " /d", shell=True)
    try:
        retcode   =   call(command   +   "   \"Iridium   Direct   Internet\"   881676328037
881676328037")
        if retcode < 0:
            print "Child was terminated by signal", -retcode
        else:
            pass
    except OSError, e:
        print "Execution failed:", e
    return retcode



def distance_on_Earth(lat1, long1, lat2, long2):
        # Convert latitude and longitude to
        # spherical coordinates in radians.
    degrees_to_radians = math.pi/180.0

        # phi = 90 - latitude
    phi1 = (90.0 - lat1)*degrees_to_radians
    phi2 = (90.0 - lat2)*degrees_to_radians

        # theta = longitude
    theta1 = long1*degrees_to_radians
    theta2 = long2*degrees_to_radians

        # Compute spherical distance from spherical coordinates.

        # For two locations in spherical coordinates
        # (1, theta, phi) and (1, theta, phi)
        # cosine( arc length ) =
        #    sin phi sin phi' cos(theta-theta') + cos phi cos phi'
        # distance = rho * arc length

    cos = float(math.sin(phi1)*math.sin(phi2)*math.cos(theta1 - theta2) +
               math.cos(phi1)*math.cos(phi2))
    arc = (math.acos( cos ))*3440 # mean radius of the Earth is 3440 NM
```

```
        # Remember to multiply arc by the radius of the earth
        # in your favorite set of units to get length.
    return arc

def int2bin(n, count=24):
    return "".join([str((n >> y) & 1) for y in range (count-1,-1,-1)])

def char_to_bin(string):
    binstring = ''
    for i in range(0, len(string), 1):
        char_val = ord(string[i])- 48
        if char_val > 39:
            char_val -= 8
        binstring += int2bin(char_val,6)
    return binstring

def fwd_and_log(string, sock, host, port, not_sent):
    time1 = str(math.trunc(time()))
    fwd_message = (str(message_count)+'R:'+time1+':'+string)
    not_sent.add(fwd_message)
    #print not_sent
    try:
        net_test_message = str(message_count)
        if net_test_message[-1] == '1':
                resend_stauts = internet_test(sock)
                if resend_stauts == 1:
                    resend(not_sent)
                    not_sent.clear()
                else: not_sent.clear()
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.sendto((str(message_count)+':'+time1+':'+string), (host,port))

        #sleep(2)
        sock.close()
        print (str(message_count)+'-'+string[:-1]+' '+time1+'\r\n')
        date1 = strftime('%d%b%Y')
        log_file = open("fwd_messages_"+date1+".txt", "a")
        log_file.write(str(message_count)+'-'+string[:-1]+' '+time1+'\r\n')
        log_file.close()
    except IOError as (errorno):
        #sock.shutdown(SHUT_RDWR)
        #sock.close()
        if '10051' in str(errorno):
            print  'Iridium  connection  down,  this  message  logged  locally  but  not
forwarded: '
            print (str(message_count)+'-'+string[:-1]+' '+time1+'\r\n')
            print 'trying to reconnect...'
            s = connect(0)
            print 'Connected, trying again...'
        elif '10060' in str(errorno):
            print 'Timed out. This message logged locally but not forwarded: '
            print (str(message_count)+'-'+string[:-1]+' '+time1+'\r\n')
            s = connect(0)
        else:
            print 'this message logged locally but and error occured:'
            print (str(message_count)+'-'+string[:-1]+' '+time1+'\r\n')
            print errorno
            print ''
            s = connect(0)
        log_file = open("fwd_messages.txt", "a")
        log_file.write('***'+str(message_count)+''+fwd_message+'\r\n')
        log_file.write(str(errorno)+'\r\n')
        log_file.close()

    return

# 6-bit conversion
def to_asciisixbit_chars(bitstring):
    sixbit_chars = ""
    # Process 6-bit chunks of the bit string
    for i in range(0, len(bitstring), 6):
        # As per AIVDO payload armoring table, convert bits to ASCII character
        sixbitint = int(bitstring[i:i+6], 2) + 48
        if sixbitint > 87:
            sixbitint += 8
        sixbit_chars += chr(sixbitint)
    return sixbit_chars

# 6-bit conversion for string variables
```

```
def string_to_bin(string):
    binstring = ''
    # Process 1 6-bit character at a time
    for i in range(0, len(string), 1):
        char_val = ord(string[i])
        if 64 < char_val < 96: char_val -= 64
        if 96 < char_val < 127: char_val -= 96
        binstring += int2bin(char_val,6)

        # As per AIS string encoding table, convert bits to ASCII character
    return binstring

#Get time fields
def get_time():
    time = localtime()
    day = int2bin(time[2], 5)
    hour = int2bin(time[3], 5)
    minute = int2bin(time[4], 6)
##    month = -1
##    if num_fields == 3:
##        second = int2bin(time[5]/5, 4)
    second = int2bin(time[5], 6)
    month = int2bin(time[1], 4)
    return (day, hour, minute, second, month)

def send_gps_ais(ev02, ev03, ev56,RADAR_num,ais_numA,ais_numB,gps_string):
    gps_fields = gps_string.split(",")
    gps_lat = gps_fields[3]
    gps_lat_h = gps_fields[4]
    gps_long = gps_fields[5]
    gps_long_h = gps_fields[6]
    gps_lat_d = int(gps_lat[0:2])
    gps_lat_m = float(gps_lat[2:9])
    gps_lat_10m = int((600*gps_lat_d)+(10*gps_lat_m))
    gps_lat_10000m = int((600000*gps_lat_d)+(10000*gps_lat_m))
    if gps_lat_h == 'S':
        gps_lat_10m = gps_lat_10m*-1
        gps_lat_10000m = gps_lat_10000m*-1
    gps_long_d = int(gps_long[0:3])
```

```
gps_long_m = float(gps_long[3:10])
gps_long_10m = int((600*gps_long_d)+(10*gps_long_m))
gps_long_10000m = int((600000*gps_long_d)+(10000*gps_long_m))
if gps_long_h == 'W':
    gps_long_10m = gps_long_10m*-1
    gps_long_10000m = gps_long_10000m*-1
latitude10bin = int2bin(math.trunc(gps_lat_10m), 16)
latitude10000bin = int2bin(math.trunc(gps_lat_10000m), 27)
longitude10bin = int2bin(math.trunc(gps_long_10m), 17)
longitude10000bin = int2bin(math.trunc(gps_long_10000m), 28)
sog10 = int(10*float(gps_fields[7]))
sog10bin = int2bin(sog10, 8)
cog = math.trunc(round(float(gps_fields[8])))
if cog == 360: cog = 0
cogbin = int2bin(cog,9)
if ais_numA > ais_numB:
    ais_num = ais_numA
else: ais_num = ais_numB
ais_contactsbin = int2bin(ais_num, 6)
RADAR_contactsbin = int2bin(RADAR_num, 5)
(daybin, hourbin, minutebin, secondbin, monthbin) = get_time()
ag_time = localtime()
secondbin2 = int2bin(ag_time[5]/5, 4)
bintext_gps1 = "000%s%s%s%s%s%s0000" % (longitude10bin, latitude10bin,
                                      daybin, hourbin, minutebin, secondbin2)
sixbit_chars1 = to_asciisixbit_chars(bintext_gps1)
bintext_gps2 = "001%s%s%s%s%s%s%s%s%s%s%s00" % (ais_contactsbin,
                                              RADAR_contactsbin, longitude10000bin,
                                              latitude10000bin, sog10bin, cogbin,
                                              monthbin, daybin, hourbin, minutebin,
                                              secondbin)
sixbit_chars2 = to_asciisixbit_chars(bintext_gps2)
telnet_command1 = "!SNBBM,1,1,0,0,8,%s,4\r\n" % sixbit_chars1
output_command1 = "!SNBBM,1,1,0,0,8,%s,4" % sixbit_chars1
telnet_command2 = "!SNBBM,1,1,0,0,8,%s,2\r\n" % sixbit_chars2
output_command2 = "!SNBBM,1,1,0,0,8,%s,2" % sixbit_chars2
ais_HOST = '192.168.0.200'    # The remote host
ais_PORT = 4712              # The same port as used by the server
print "\nGoing to send this over TCP to %s:" % ais_HOST
```

```
    print telnet_command1
    if RADAR_num < 32:
        print telnet_command2
    sais = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sais.connect((ais_HOST, ais_PORT))
    gps_tx = 0
    while gps_tx < 9:
        sais.sendall(telnet_command1)
        if RADAR_num < 32:
            sais.sendall(telnet_command2)
        gps_tx += 1
    sais.sendall("$SNAIR,316086000,3,,3,,316086000,3,,,,,\r\n")
    sais.close()
    #print "\nSent to transponder.  Waiting for transmitted message...\n"
    #sais.close()
    tx_time = str(time())
    date3 = strftime('%d%b%Y')
    ais_log_file = open("ais_fwd_messages_"+date3+".txt", "a")

ais_log_file.write(tx_time+','+output_command1+','+str(ev02)+str(ev03)+str(ev56)+'\r\n')
    if RADAR_num < 32:

ais_log_file.write(tx_time+','+output_command2+','+str(ev02)+str(ev03)+str(ev56)+'\r\n')
    ais_log_file.close()
    return


def send_aivdm_ais(ev02, ev03, ev56, mmsi_ais, lat_ais, long_ais, speed_ais, course_ais):
    lat_aisb = int2bin(math.trunc(lat_ais*6000), 21)
    lon_aisb = int2bin(math.trunc(long_ais*6000), 22)
    round_sp_ais = math.trunc(round(speed_ais))
    sog_aisb = int2bin(round_sp_ais, 5)
    round_co_ais = math.trunc(round(course_ais))
    cog_aisb = int2bin(round_co_ais, 9)
    (daybin, hourbin, minutebin, secondbin, monthbin) = get_time()
    bintext_ais = "010%s%s%s%s%s%s%s%s%s00" % (mmsi_ais, lon_aisb,lat_aisb,
                                            sog_aisb,     cog_aisb,     daybin,hourbin,
minutebin, secondbin)
    sixbit_chars_ais = to_asciisixbit_chars(bintext_ais)
    telnet_command_ais = "!SNBBM,1,1,0,0,8,%s,2\r\n" % sixbit_chars_ais
    output_command_ais = "!SNBBM,1,1,0,0,8,%s,2" % sixbit_chars_ais
    ais_HOST = '192.168.0.200'    # The remote host
    ais_PORT = 4712              # The same port as used by the server
    print "\nGoing to send this over TCP to %s:" % ais_HOST
    print telnet_command_ais
    mmsi_aiss = int(mmsi_ais, 2)
    if mmsi_ais < 800000000:
        telnet_inter_command = "$SNAIR,%s,3,,3,,%s,3,,,,,\r\n" % (mmsi_aiss,mmsi_aiss)
        output_inter_command = "$SNAIR,%s,3,,3,,%s,3,,,,," % (mmsi_aiss,mmsi_aiss)
        print telnet_inter_command
    sais = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sais.connect((ais_HOST, ais_PORT))
    ais_tx = 0
    while ais_tx < 9:
        sais.sendall(telnet_command_ais)
        ais_tx += 1
    if mmsi_ais < 800000000:
        sais.sendall(telnet_inter_command)
    #print "\nSent to transponder.  Waiting for transmitted message...\n"
    #sais.close()
    sais.close()
    tx_time = str(time())
    date3 = strftime('%d%b%Y')
    ais_log_file = open("ais_fwd_messages_"+date3+".txt", "a")

ais_log_file.write(tx_time+','+output_command_ais+','+str(ev02)+str(ev03)+str(ev56)+'\r\n
')
    ais_log_file.close()
    return


def send_RADAR_ais(ev02, ev03, ev56, ttmgps_string):
    ttmgps_fields = ttmgps_string.split(",")
    gps_lat = ttmgps_fields[17]
    gps_lat_h = ttmgps_fields[18]
    gps_long = ttmgps_fields[19]
    gps_long_h = ttmgps_fields[20]
    gps_lat_d = int(gps_lat[0:2])
    gps_lat_m = float(gps_lat[2:9])
    gps_lat_100m = int((6000*gps_lat_d)+(100*gps_lat_m))
    if gps_lat_h == 'S':
        gps_lat_100m = gps_lat_100m*-1
    gps_long_d = int(gps_long[0:3])
    gps_long_m = float(gps_long[3:10])
    gps_long_100m = int((6000*gps_long_d)+(100*gps_long_m))
    if gps_long_h == 'W':
        gps_long_100m = gps_long_100m*-1
    latitude100bin = int2bin(math.trunc(gps_lat_100m), 16)
    longitude100bin = int2bin(math.trunc(gps_long_100m), 17)
    (daybin, hourbin, minutebin, secondbin, monthbin) = get_time()
    range_RADAR = float(ttmgps_fields[2])
    range_r_bin = int2bin(math.trunc((range_RADAR)*10), 9)
    bearing_RADAR = float(ttmgps_fields[3])
    bearing_r_bin = int2bin(math.trunc((bearing_RADAR)*10), 12)
    sog = math.trunc(round(float(ttmgps_fields[5])))
    sogbin = int2bin(sog, 8)
    cog_r = math.trunc(round(float(ttmgps_fields[6])))
    if cog_r == 360: cog_r = 0
    cog_r_bin = int2bin(cog_r,5)
    ARPA_id = int(ttmgps_fields[1])
    ARPA_idbin = int2bin(ARPA_id, 7)
    if '$13RATTM' in ttmgps_string:
        RADAR_id = '01'  #mapping 13RATTM to aft plot room (normally S-band)
    else: RADAR_id ='00' #mapping 15RATTM to bridge (normally X-band)
    bintext_r = "100%s%s%s%s%s%s%s%s%s%s%s%s%s00" % (RADAR_id, ARPA_idbin, range_r_bin,
                                                          bearing_r_bin,    longitude100bin,
latitude100bin,
                                                          sogbin,     cog_r_bin,     daybin,
hourbin, minutebin, secondbin)
    sixbit_chars_r = to_asciisixbit_chars(bintext_r)
    telnet_command_r = "!SNBBM,1,1,0,0,8,%s,2\r\n" % (sixbit_chars_r)
    output_command_r = "!SNBBM,1,1,0,0,8,%s,2" % (sixbit_chars_r)
    ais_HOST = '192.168.0.200'    # The remote host
    ais_PORT = 4712              # The same port as used by the server
    print "\nGoing to send this over TCP to %s:" % ais_HOST
    print telnet_command_r
    sais = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sais.connect((ais_HOST, ais_PORT))
    ais_tx = 0
    while ais_tx < 9:
        sais.sendall(telnet_command_r)
        ais_tx += 1
    #print "\nSent to transponder.  Waiting for transmitted message...\n"
    #sais.close()
    sais.close()
    tx_time = str(time())
    date3 = strftime('%d%b%Y')
    ais_log_file = open("ais_fwd_messages_"+date3+".txt", "a")
ais_log_file.write(tx_time+','+output_command_r+','+str(ev02)+str(ev03)+str(ev56)+'\r\n')
    ais_log_file.close()
    return

def ais_tx_status(check_t):
    index1 = 0
    index2 = 0
    index3 = 0
    early1 = 0
    early2 = 0
    early3 = 0
    #with open("Q346-EV02_access_test.txt","rb") as log_file:
    with open("Q346-EV02_access.csv","rb") as log_file:
        log_file.seek(index1,0)
        line1 = log_file.next()
        open_t = int(line1[0:10])
        close_t = int(line1[11:-2])
        early1 = 0
        while early1 == 0:
            if check_t > (close_t+180):
                index1 += len(line1)
                line1 = log_file.next()
                open_t = int(line1[0:10])
                close_t = int(line1[11:-2])
            else:
                early1 = 1
        if (open_t-180) < check_t < (close_t+180):
            transmit1 = 1
        elif check_t < (open_t-180):
            transmit1 = 0
        log_file.close()

    with open("Q346-EV03_access.csv","rb") as log_file:
        log_file.seek(index2,0)
        line2 = log_file.next()
        open_t = int(line2[0:10])
        close_t = int(line2[11:-2])
        early2 = 0
        while early2 == 0:
            if check_t > (close_t+180):
                index2 += len(line2)
                line2 = log_file.next()
                open_t = int(line2[0:10])
                close_t = int(line2[11:-2])
            else:
                early2 = 1
        if (open_t-180) < check_t < (close_t+180):
            transmit2 = 1
        elif check_t < (open_t-180):
            transmit2 = 0
        log_file.close()

    with open("Q346-EV56_access.csv","rb") as log_file:
        log_file.seek(index3,0)
        line3 = log_file.next()
        open_t = int(line3[0:10])
        close_t = int(line3[11:-2])
        early3 = 0
        while early3 == 0:
            if check_t > (close_t+180):
                index3 += len(line3)
                line3 = log_file.next()
                open_t = int(line3[0:10])
                close_t = int(line3[11:-2])
            else:
                early3 = 1
        if (open_t-180) < check_t < (close_t+180):
            transmit3 = 1
        elif check_t < (open_t-180):
            transmit3 = 0
        log_file.close()
    return (transmit1, transmit2, transmit3)
```

```
s = connect(online)

while True:

# Receive messages
    data,addr = UDPSock.recvfrom(udp_buf)
    if not data:
        print "No UDP feed"
        break
    else:
        #GPS
        if '12$GPRMC' in data: #change to 12 for QUEST
            gps_count += 1
            rmc_fields = data.split(",")
            curr_rmc = data
            curr_rmc_time = time()
            if gps_count > 10 or gps_count == 0: #decimation in seconds
                gps_count = 0
                message_count += 1
                fwd_and_log(data,s,host,port,not_sent)
                (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                    send_gps_ais(transmit1, transmit2, transmit3,

RADAR_count1,ais_contacts_countA,ais_contacts_countB,data)
        #AIS
        if '11!AIVDM' in data or '14!BSVDM' in data: #11 and 14 for QUEST
            dec_aivdm1 = ''
            for i in range(16,24,1): #make 14 and 47 for DRDC
                dec_aivdm1 += data[i]
                #print dec_aivdm
            dec_bin1 = char_to_bin(dec_aivdm1)
            mmsi1231 = int(dec_bin1[8:38], 2)
            mmsi123b1 = dec_bin1[8:38]
            mmsi123s1 = str(mmsi1231)
            if (data[16]) not in ais_filter:  #make 14 for DRDC
                message_count += 1
                if mmsi123s1 != '111222333':
                    fwd_and_log(data,s,host,port,not_sent)
                else:
                    tx_time = str(time())
                    date5 = strftime('%d%b%Y')
                    ais_log_file = open("ais_tx_messages_"+date5+".txt", "a")
                    ais_log_file.write(tx_time+','+data+'\r\n')
                    ais_log_file.close()
            if (data[16]) in ais_1_2_3: #make 14 for DRDC
                dec_aivdm = ''
                for i in range(16,49,1): #make 14 and 47 for DRDC
                    dec_aivdm += data[i]
                    #print dec_aivdm
                dec_bin = char_to_bin(dec_aivdm)
                mmsi123 = int(dec_bin[8:38], 2)
                mmsi123b = dec_bin[8:38]
                mmsi123s = str(mmsi123)
                speed123 = float(int(dec_bin[50:60],2))
                long123 = float(int(dec_bin[61:89],2))
                timestamp = math.trunc(time())
                if dec_bin[61] is '1': #check for negative longitude
                    long123 = (long123 - 268435456)
                lat123 = float(int(dec_bin[89:116],2))
                if dec_bin[89] is '1': #check for negative latitude
                    lat123 = (lat123 - 134217728)
                course123 = float(int(dec_bin[116:128],2))
                long123d = float(long123/600000)
                lat123d = float(lat123/600000)
                if mmsi123s in ship_table:
                    curr_data = ship_table[mmsi123s]

                    curr_long = float((curr_data[0])/600000)
                    curr_lat = float((curr_data[1])/600000)
                    if curr_lat == lat123d and curr_long == long123d:
                        dist_diff = 0
                    elif abs(long123d) >= 180 or abs(lat123d) >= 90:
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_aivdm_ais(transmit1,   transmit2,   transmit3,   mmsi123b,
lat123d, long123d, speed123/10, course123/10)
```

```
                        dist_diff = 0
                        print 'MMSI: %d is reporting an invalid or default position: Lat
%f  Long %f \r\n' % (mmsi123, lat123d, long123d)
                    else:
                        dist_diff = float(distance_on_Earth(curr_lat, curr_long, lat123d,
long123d))


                    if  dist_diff > 5:  #5 NM tolerance
                        ship_table[mmsi123s]  =  (long123,  lat123,  speed123,  course123,
timestamp)
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_aivdm_ais(transmit1,  transmit2,  transmit3,  mmsi123b,
lat123d, long123d, speed123/10, course123/10)
                        print 'MMSI: %d has moved by %f NM \r\n' % (mmsi123, dist_diff)
                    elif speed123 > 500 and speed123 != 1023 and data[16] != '9': #change
16 to 14 for DRDC
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_aivdm_ais(transmit1,  transmit2,  transmit3,  mmsi123b,
lat123d, long123d, speed123/10, course123/10)
                        print 'MMSI: %d is reporting a suspect speed: %g knots \r\n' %
(mmsi123, speed123/10)
                    elif abs((curr_data[2] - speed123)) > 20: #2 knot tolerance
                        ship_table[mmsi123s]  =  (long123,  lat123,  speed123,  course123,
timestamp)
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_aivdm_ais(transmit1,  transmit2,  transmit3,  mmsi123b,
lat123d, long123d, speed123/10, course123/10)
                        print 'MMSI: %d has changed speed from %g knots to %g knots \r\n'
% (mmsi123, curr_data[2]/10, speed123/10)
                    elif course123 > 3600:
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_aivdm_ais(transmit1,  transmit2,  transmit3,  mmsi123b,
lat123d, long123d, speed123/10, course123/10)
                        print 'MMSI: %d is reporting an invalid or default course: %g
\r\n' % (mmsi123, course123/10)
                    elif speed123 > 10: # no course check for stationary targets
                        if  3450  >  abs((curr_data[3]  -  course123))  >  150:  #15  degree
tolerance
                            ship_table[mmsi123s] = (long123, lat123, speed123, course123,
timestamp)
                            message_count += 1
                            fwd_and_log(data,s,host,port,not_sent)
                            (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                            if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                                send_aivdm_ais(transmit1, transmit2, transmit3, mmsi123b,
lat123d, long123d, speed123/10, course123/10)
                            print  'MMSI:  %d  has  changed  course  from  %g  to  %g  \r\n'  %
(mmsi123, curr_data[3]/10, course123/10)
                    elif (timestamp - curr_data[4]) > 599: #10 minute tolerance
                        ship_table[mmsi123s]  =  (long123,  lat123,  speed123,  course123,
timestamp)
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_aivdm_ais(transmit1,  transmit2,  transmit3,  mmsi123b,
lat123d, long123d, speed123/10, course123/10)
                        time_diff = (timestamp - curr_data[4])
                        print  '%g  seconds  since  last  update  from  MMSI:  %d  \r\n'  %
(time_diff, mmsi123)
                    else:
                        #print "no change %d" % mmsi123
                        pass
                elif mmsi123 > 999999999:
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_aivdm_ais(transmit1,  transmit2,  transmit3,  mmsi123b,
lat123d, long123d, speed123/10, course123/10)
                        print 'invalid MMSI: %d \r\n' % (mmsi123)
                else:
                    ship_table[mmsi123s]  =  (long123,  lat123,  speed123,  course123,
timestamp)
```

```
                    message_count += 1
                    fwd_and_log(data,s,host,port,not_sent)
                    (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                    if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                        send_aivdm_ais(transmit1,    transmit2,    transmit3,    mmsi123b,
lat123d, long123d, speed123/10, course123/10)
                    print 'new entry MMSI: %d \r\n' % (mmsi123)
        #RADAR
        if '13$RATTM' in data:  #13 for QUEST
            ttm_fields1 = data.split(",")
            ARPA_id1 = int(ttm_fields1[1])
            ttm_fields1[13] = time()
            range1 = float(ttm_fields1[2])
            bearing1 = float(ttm_fields1[3])
            if  abs(float(ttm_fields1[13])  -  curr_rmc_time)  <  3.0:  #  3  second  tolerance
for GPRMC
                if ARPA_id1 not in RADAR_table1 and ttm_fields1[12] == 'T':
                    RADAR_count1 += 1
                    RADAR_table1[ARPA_id1] = list(ttm_fields1[2:14])
                    data = data+','+curr_rmc
                    message_count += 1
                    fwd_and_log(data,s,host,port,not_sent)
                    (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                    if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                        send_RADAR_ais(transmit1, transmit2, transmit3, data)
                    print 'new RADAR contact, ID: %d \r\n' % (ARPA_id1)
                if ARPA_id1 in RADAR_table1 and ttm_fields1 == 'L':
                    data = data+','+curr_rmc
                    message_count += 1
                    fwd_and_log(data,s,host,port,not_sent)
                    (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                    if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                        send_RADAR_ais(transmit1, transmit2, transmit3, data)
                    RADAR_count1 -= 1
                    del RADAR_table1[ARPA_id1]
                    print 'target lost, ID: %d \r\n' % (ARPA_id1)
                elif ARPA_id1 in RADAR_table1 and ttm_fields1[12] == 'T':
                    curr_data1 = RADAR_table1[ARPA_id1]
                    if   abs(float(curr_data1[3])   -   float(ttm_fields1[5]))   >   2.0   and
ttm_fields1[10] == curr_data1[8]:  #2 knot tolearnce
```

```
                        RADAR_table1[ARPA_id1] = list(ttm_fields1[2:14])
                        data = data+','+curr_rmc
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_RADAR_ais(transmit1, transmit2, transmit3, data)
                        print 'target ID: %d has changed speed from %s knots to %s knots
\r\n' % (ARPA_id1, curr_data1[3], ttm_fields1[5])
                    elif (abs(float(curr_data1[4]) - float(ttm_fields1[6])) > 15.0) and
ttm_fields1[7] == curr_data1[5]: #15 degree tolerance
                        RADAR_table1[ARPA_id1] = list(ttm_fields1[2:14])
                        data = data+','+curr_rmc
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_RADAR_ais(transmit1, transmit2, transmit3, data)
                        print 'target ID: %d has changed course from %s to %s \r\n' %
(ARPA_id1, curr_data1[4], ttm_fields1[6])
                    elif abs(float(curr_data1[11]) - float(ttm_fields1[13])) > 300.0: #5
minute tolerance
                        RADAR_table1[ARPA_id1] = list(ttm_fields1[2:14])
                        data = data+','+curr_rmc
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_RADAR_ais(transmit1, transmit2, transmit3, data)
                        #time_diff_RADAR        =        abs(float(curr_data1[11])        -
float(ttm_fields1[13]))
                        print '%f seconds since last update from target ID: %d \r\n' %
(time_diff_RADAR, ARPA_id1)
                    else: print 'no change target ID: %d' % (ARPA_id1)

        if '15$RATTM' in data:  #15 for QUEST (other RADAR)
            ttm_fields2 = data.split(",")
            ARPA_id2 = int(ttm_fields2[1])
            ttm_fields2[13] = time()
            range2 = float(ttm_fields2[2])
            bearing2 = float(ttm_fields2[3])
            if abs(float(ttm_fields2[13]) - curr_rmc_time) < 3.0: # 3 second tolerance
for GPRMC
                if ARPA_id2 not in RADAR_table2 and ttm_fields2[12] == 'T':
                    RADAR_count2 += 1
                    RADAR_table2[ARPA_id2] = list(ttm_fields2[2:14])
                    data = data+','+curr_rmc
                    message_count += 1
                    fwd_and_log(data,s,host,port,not_sent)
                    (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                    if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                        send_RADAR_ais(transmit1, transmit2, transmit3, data)
                    print 'new RADAR contact, ID: %d \r\n' % (ARPA_id2)
                if ARPA_id2 in RADAR_table2 and ttm_fields2 == 'L':
                    data = data+','+curr_rmc
                    message_count += 1
                    fwd_and_log(data,s,host,port,not_sent)
                    (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                    if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                        send_RADAR_ais(transmit1, transmit2, transmit3, data)
                    RADAR_count2 -= 1
                    del RADAR_table2[ARPA_id2]
                    print 'target lost, ID: %d \r\n' % (ARPA_id2)
                elif ARPA_id2 in RADAR_table2 and ttm_fields2[12] == 'T':
                    curr_data2 = RADAR_table2[ARPA_id2]
                    if  abs(float(curr_data2[3])  -  float(ttm_fields2[5]))  >  2.0  and
ttm_fields2[10] == curr_data2[8]:  #2 knot tolearnce
                        RADAR_table2[ARPA_id2] = list(ttm_fields2[2:14])
                        data = data+','+curr_rmc
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_RADAR_ais(transmit1, transmit2, transmit3, data)
                        print 'target ID: %d has changed speed from %s knots to %s knots
\r\n' % (ARPA_id2, curr_data2[3], ttm_fields2[5])
                    elif (abs(float(curr_data2[4]) - float(ttm_fields2[6])) > 15.0) and
ttm_fields2[7] == curr_data2[5]: #15 degree tolerance
                        RADAR_table2[ARPA_id2] = list(ttm_fields2[2:14])
                        data = data+','+curr_rmc
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_RADAR_ais(transmit1, transmit2, transmit3, data)
                        print 'target ID: %d has changed course from %s to %s \r\n' %
(ARPA_id2, curr_data2[4], ttm_fields2[6])
                    elif abs(float(curr_data2[11]) - float(ttm_fields2[13])) > 300.0: #5
minute tolerance
                        RADAR_table2[ARPA_id2] = list(ttm_fields2[2:14])
                        data = data+','+curr_rmc
                        message_count += 1
                        fwd_and_log(data,s,host,port,not_sent)
                        (transmit1,transmit2,transmit3) = ais_tx_status(time1)
                        if transmit1 == 1 or transmit2 == 1 or transmit3 == 1:
                            send_RADAR_ais(transmit1, transmit2, transmit3, data)
                        #time_diff_RADAR = abs(float(curr_data2[11]) -
float(ttm_fields2[13]))
                        print '%f seconds since last update from target ID: %d \r\n' %
(time_diff_RADAR, ARPA_id2)
                    else: print 'no change target ID: %d' % (ARPA_id2)


        #Received Stations message
        if '11$PSTT,10C,A' in data: #change to 11 for QUEST
            ais_contacts_countA += 1
            if ais_contacts_countA > 10 or ais_contacts_countA == 0: #decimation in
minutes
                pstt_fields = data.split(",")
                rs = pstt_fields[4]
                ais_contacts = rs[0:3]
                ais_contacts_countA = 0
                message_count += 1
                fwd_and_log(data,s,host,port,not_sent)


        if '11$PSTT,10C,B' in data: #change to 11 for QUEST
            ais_contacts_countB += 1
            if ais_contacts_countB > 10 or ais_contacts_countB == 0: #decimation in
minutes
                pstt_fields = data.split(",")
                rs = pstt_fields[4]
                ais_contacts = rs[0:3]
                ais_contacts_countB = 0
                message_count += 1
                fwd_and_log(data,s,host,port,not_sent)








        # Close socket
```

# Annex B AIS200P Binary Message Formats

## B.1 Binary Broadcast Message Request

This message is sent to the transponder via TCP/IP (192.168.0.1:4712) from PC1 shown in Figure 2. In response the transponder creates and sends either a type 8 (binary broadcast) or type 14 (safety-related broadcast) message. For our purposes, these two AIS message types are essentially identical. This is the format of the message that gets sent over TCP/IP to the transponder:

!SNBBM,1,1,0,c,m,d--d,f (B.1)

This is a modified version of the generic sentence found in the international standard [24]. It has been tailored to the AIS200P transponder. Table B-1 describes the binary broadcast request message.

*Table B-1: Binary Broadcast Message Format.*

| Field | Description |
|---|---|
| !SNBBM | Indicates a 6-bit ASCII binary broadcast message (BBM) from talker ‘SN’. The talker ID is defined as ‘SN’ by Kongsberg Seatex. This field will always be the same |
| Total number of sentences | Always set to 1 |
| Sentence Number | Always set to 1 |
| Sequential Message Identifier | Always set to 0 |
| (c) AIS channel for broadcast | 0 = no preference, 1 = channel A, 2 = channel B, 3 = dual (not supported by MIS portable Class A transponder) |
| (m) AIS message type | Decimal integer containing the message code and corresponding data. This field will be either 8 or 14 |
| (d--d) Transmitted data | 6-bit ASCII string representing the 56, 112 or 320 bits of the MIKM 8/14 message |
| (f) Fill bits | Number of LSB to ignore to preserve 8-bit byte boundary in the transmitted data |

## B.2 Special Type 8 Message Formats

Using the Binary Broadcast Message Request, these messages are used to communicate dynamic vessel data on QUEST and contacts received on QUEST's sensors via GPS, AIS, ADS-B and RADAR. Table B-2 outlines the generic message format.

*Table B-2: Generic Special Type 8 Message Format.*

| Parameter | Number of bits | Description |
|---|---|---|
| Message ID | 6 | Message Type, will be either 8 or 14 and is essentially arbitrary |
| Repeat indicator | 2 | Used by the repeater to indicate how many times a message has been repeated. These will always be set to zero |
| Source ID | 30 | MMSI number of source station. This will always be 111222333 |
| Spare | 2 | Not used. Should be set to zero |
| Message Code | 3 | Indicates the content of the rest of the message |
| Message 8 data | 53 or 109 or 309 or 317 | Dependant on message code |
| Fill bits | 0 or 2 or 4 | Bits needed when expressing the 8-bit byte message in 6-bit ASCII |
| Total number of bits | 100 or 154 or 352 | Occupies ½, 1 or 2 slots |

The Message Code (see Table B-2) determines the content of the message. Four different message types were used according to the Table B-3. It should be noted that Message Code 3 was not used.

*Table B-3: Special Type 8 Message Codes.*

| Message Code | Number of data bits | Number of fill bits | Description |
|---|---|---|---|
| 0 | 53 | 4 | Short own-ship message |
| 1 | 109 | 2 | Regular own-ship message |
| 2 | 109 | 2 | AIS contact report |
| 4 | 109 | 2 | RADAR contact report (not held on AIS) |

### B.2.1 Half-Slot Own-ship report

This message uses QUEST's GPS data to construct a short, half-slot length AIS message. Table B-4 describes the contents of this message.

*Table B-4: Half-Slot Own-ship Report Format.*

| Parameter | Number of bits | Description |
|---|---|---|
| Message code | 3 | Indicates the content of the data to follow. Always set to 0. |
| Longitude | 17 | Longitude in 1/10 min (W Longitude), values above 1 080 000 not used |
| Latitude | 16 | Latitude in 1/10 min (N Latitude), values above 540 000 not used |
| Day | 5 | 1-31, 0 not used |
| Hour | 5 | 0-23, 24-31 not used |
| Minute | 6 | 0-59, 60-63 not used |
| Second (rounded) | 4 | 0-11 (multiply by 5 to get rounded seconds value), 12-15 not used |
| Fill bits | 4 | Needed when expressing the 8-bit byte message in 6-bit ASCII |
| Total number of bits | 60 | This ½ slot message reports own-ship position and time to the nearest 5 seconds. |

### B.2.2 Full Slot Own-ship report

This message uses QUEST's GPS data as well as selected data from the ship's AIS and RADAR to construct a single-slot length AIS message. Table B-5 describes the contents of this message.

*Table B-5: Full Slot Own-ship Report Format.*

| Parameter | Number of bits | Description |
|---|---|---|
| Message code | 3 | Indicates the content of the data to follow. Always set to 1. |
| Stations received | 6 | Number of AIS contacts |
| RADAR contacts | 5 | Number of RADAR contacts |
| Longitude | 28 | Longitude in 1/10 000 min (±180º, East = positive (as per 2's complement), West = negative (as per 2's complement) |
| Latitude | 27 | Latitude in 1/10 000 min (±90°, North = positive (as per 2's complement), South = negative (as per 2's complement) |
| SOG | 8 | 0-255, speed over the ground in 1/10 knots |
| COG | 9 | 0-359, course over the ground in degrees, 360-511 not used |
| Month | 4 | 1-12, 0 and 13-15 not used |
| Day | 5 | 1-31, 0 not used |
| Hour | 5 | 0-23, 24-31 not used |
| Minute | 6 | 0-59, 60-63 not used |
| Second | 6 | 0-59, 60-63 not used |
| Fill bits | 2 | Needed when expressing the 8-bit byte message in 6-bit ASCII |
| Total number of bits | 114 | This 1 slot message reports own-ship position, course, speed, number of contacts and the time. |

### B.2.3 AIS/ADS-B contact report

This message uses data from QUEST's AIS or the installed ADS-B receiver to construct a contact report. Table B-6 describes the contents of this message.

*Table B-6: AIS/ADS-B Contact Report Format.*

| Parameter | Number of bits | Description |
|---|---|---|
| Message code | 3 | Set to 2 or 3. (2 = AIS & RADAR contact, 3 = AIS only) |
| Target ID | 30 | MMSI number of the contact |
| Longitude | 22 | Longitude in 1/100 min (±180º, East = positive (as per 2's complement), West = negative (as per 2's complement) |
| Latitude | 21 | Latitude in 1/100 min (±90°, North = positive (as per 2's complement), South = negative (as per 2's complement) |
| SOG | 5 | 0-31, speed over the ground in knots |
| COG | 9 | 0-359, course over the ground in degrees, 360-511 not used |
| Day | 5 | 1-31, 0 not used |
| Hour | 5 | 0-23, 24-31 not used |
| Minute | 6 | 0-59, 60-63 not used |
| Second | 6 | 0-59, 60-63 not used |
| Fill bits | 2 | Needed when expressing the 8-bit byte message in 6-bit ASCII |
| Total number of bits | 114 | This 1 slot message reports a contact MMSI, position, visibility, course and speed and the time. |

### B.2.4 RADAR contact report

This message uses data from either of QUEST's RADAR and GPS to construct a RADAR contact report. The GPS data is required to compute the true position of the contact from the relative data generated by the RADAR ARPA. Table B-7 describes the contents of this message.

*Table B-7: RADAR Contact Report Format.*

| Parameter | Number of bits | Description |
|---|---|---|
| Message code | 3 | Indicates the content of the data to follow. Always set to 4. |
| RADAR ID | 2 | Frequency band (0 = X-band, 1 = S-band, 2 = Furuno, 3 = other) |
| Target ID | 7 | 0-99, ARPA Target number (100-127 not used) |
| Target Distance | 9 | 0-511, range from own-ship in 1/10 NM |
| Target Bearing | 12 | 0-3599, bearing form own-ship in 1/10 degree, 3600-4095 not used. |
| Own-ship Longitude | 22 | Longitude in 1/100 min (±180º, East = positive (as per 2's complement), West = negative (as per 2's complement) |
| Own-ship Latitude | 21 | Latitude in 1/100 min (±90°, North = positive (as per 2's complement), South = negative (as per 2's complement) |
| SOG | 5 | 0-31, speed over the ground in knots |
| COG | 9 | 0-359, course over the ground in degrees, 360-511 not used |
| Day | 5 | 1-31, 0 not used |
| Hour | 5 | 0-23, 24-31 not used |
| Minute | 6 | 0-59, 60-63 not used |
| Second | 6 | 0-59, 60-63 not used |
| Fill bits | 2 | Needed when expressing the 8-bit byte message in 6-bit ASCII |
| Total number of bits | 114 | This 1 slot message reports a RADAR contact's ID, range, bearing, visibility, position, course and speed and the time. |

# List of symbols/abbreviations/acronyms/initialisms

| | |
|---|---|
| ADS-B | Automated Dependant Surveillance Broadcast |
| AIS | Automatic Identification System |
| ARPA | Automatic RADAR Plotting Aid |
| AVMS | Advanced Vessel Monitoring System |
| CFB | Canadian Forces Base |
| CHS | Canadian Hydrographic Services |
| CSV | Comma Separated Values |
| DEW | Distant Early Warning |
| DISA | Defense Information Systems Agency |
| GPS | Global Positioning System |
| IMO | International Maritime Organization |
| IP | Internet Protocol |
| ITDMA | Incremental Time Division Multiple Access |
| JRCC | Joint Rescue Coordination Centre |
| LBT | L-Band Transceiver |
| KML | Keyhole Markup Language |
| MCDV | Maritime Coastal Defence Vessel |
| MDA | Maritime Domain Awareness |
| MMSI | Maritime Mobile Service Identity |
| MSA | Maritime Situational Awareness |
| MSOC | Maritime Security Operations Centre |
| NM | Nautical Mile |
| NMEA | National Marine Electronics Association |
| NORAD | North American Aerospace Defense Command |
| PPP | Point-to-point Protocol |
| RJOC | Regional Joint Operations Centre |
| SA | Situational Awareness |
| SAR | Search and Rescue |
| SEDNA | Situational Information for Enabling Development of Northern Awareness |
| TCP | Transmission Control Protocol |

| | |
|---|---|
| UDP | User Datagram Protocol |
| US DoD | United States Department of Defense |
| VDR | Voyage Data Recorder |

**DOCUMENT CONTROL DATA**

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

| | | | |
|---|---|---|---|
| 1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)<br><br>Defence R&D Canada – Atlantic<br>9 Grove Street<br>P.O. Box 1012<br>Dartmouth, Nova Scotia B2Y 3Z7 | | 2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)<br><br>UNCLASSIFIED<br>(NON-CONTROLLED GOODS)<br>DMC A<br>REVIEW: GCEC APRIL 2011 | |
| 3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)<br><br>SEDNA Voyage Data Recorder: Deployable MSA Data Collection and Ex-Filtration | | | |
| 4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)<br><br>MacInnis, A. | | | |
| 5. DATE OF PUBLICATION (Month and year of publication of document.)<br><br>April 2014 | 6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)<br><br>76 | 6b. NO. OF REFS (Total cited in document.)<br><br>24 | |
| 7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)<br><br>Technical Memorandum | | | |
| 8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)<br><br>Defence R&D Canada – Atlantic<br>9 Grove Street<br>P.O. Box 1012<br>Dartmouth, Nova Scotia B2Y 3Z7 | | | |
| 9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)<br><br>Project 11ho, 11jo, 06eo | 9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.) | | |
| 10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)<br><br>DRDC Atlantic TM 2013-135 | 10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.) | | |
| 11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)<br><br>Unlimited | | | |
| 12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))<br><br>Unlimited | | | |

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

In 2012, the Maritime Information Support group at DRDC Atlantic began work on the SEDNA (Situational Information for Enabling Development of Northern Awareness) Project with focused research into issues surrounding Maritime Domain Awareness in the Arctic. This paper describes the design, development, deployment and results of Maritime Situational Awareness data collection and ex-filtration system, the SEDNA Voyage Data Recorder (VDR). The goal of this effort was to design and test methods for remote collection and ex-filtration of Maritime Situational Awareness data. The SEDNA VDR system is built upon a commercial-off-the-shelf Voyage Data Recorder and uses controlling software and an Iridium L-band transceiver to re-transmit selected data to a central location via the military Iridium gateway. The SEDNA VDR was deployed on-board CFAV QUEST during the Arctic deployment of 2012 designated Q-346. In addition to traditional maritime sensors such as GPS, AIS and RADAR using the NMEA standard messaging, an ADS-B receiver was also installed and integrated into the system. In addition to backhauling data via Iridium, application-specific AIS messages were transmitted to ex-filtrate data via the exactEarth AIS satellite constellation as a proof-of-concept.

En 2012, le groupe du Soutien à la prise de décisions maritimes à RDDC Atlantique a commencé le projet SEDNA (information en support de la connaissance de la situation maritime Arctique) sur la recherche ciblée sur les questions portant à la connaissance du domaine maritime dans l'Arctique. Ce mémorandum technique décrit le modèle, le développement, et les résultats d'un système de collection et exfiltration de données de la connaissance de la situation maritime : l'enregistreur de données de voyage (VDR) SEDNA. Le but de cet effort était de concevoir et évaluer les méthodes pour la collection et exfiltration des données de la connaissance de la situation maritime. Le système SEDNA VDR était construit d'un enregistreur de données du voyage, un produit commercial, et il utilise un logiciel de commande et un émetteur-récepteur bande L de Iridium pour transmettre les données sélectionnées au lieu central en utilisant la passerelle Internet militaire de Iridium. Le SEDNA VDR était déployé à bord du NFAC QUEST durant le voyage vers l'Arctique en 2012 appelé Q-346. En supplément aux capteurs maritimes traditionnels tel que GPS, AIS et RADAR utilisant le standard de message NMEA, un récepteur ADS-B était aussi installé et intégré dans le système. En supplément à l'exfiltration de données par Iridium, les messages de AIS spécifiques au logiciel étaient transmis par la constellation de satellites exactEarth comme preuve de concept.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

SEDNA; arctic; north; awareness; MDA

This page intentionally left blank.