Integration of Digital Mobile Radio in a Sensor Network

Anders Fongen
Norwegian Defence University College (FHS)
Lillehammer, Norway
email: anders@fongen.no

***Abstract*—The integration of a sensor network with services from voice communication, text messaging and Global Positioning System (GPS) creates opportunities for improved situational awareness, better safety for field operators, higher confidence in sensor readings and improved return on equipment investment. Digital Mobile Radio (DMR) has been the choice of communication technology for a series of experiments where these potentials have been investigated. Additional technology components used were mostly inexpensive and open source.**

Keywords*—*Internet of things; Sensor Networks; Digital Mobile Radio; MQTT

I. Introduction

Examples of the services provided by a combination of digital handheld radios with GPS and text messaging services, and a sensor network are listed below:

- Unified user interface - One unit, one battery, one charger used for personal communication and interaction with the sensor network.
- Sensor selection - Based on the location of a field operator, events from nearby sensors are selected for reporting, while others are suppressed.
- Sensor activation - Based on location, sensors near the operator may be deactivated to avoid false alarms/events.
- Automatic geofencing - Based on sensor readings (radiation, toxins, gases), operators may be warned no to enter or exit certain areas.
- Based on the velocity of GPS readings, operators during transport (e.g., while driving a vehicle) can have notification sent as voice messages, while receiving text messages when stationary.

Digital Mobile Radio (DMR) [1] is a non-proprietary digital radio standard maintained by European Telecommunications Standards Institute (ETSI) and supported by a number of manufacturers of radio equipment as well as open source and radio amateur efforts. Contrary to other digital radio systems (e.g., TETRA), DMR operates with the same channel bandwidth as Frequency Modulation (FM) systems and owners of spectrum licences can migrate from FM to DMR using the same channel plan. The DMR physical layer uses Frequency Shift Keying (4FSK) modulation and offers 9600 bits per second (bps) transfer speed, so even FM radios equipped with a data port for 9600 bps can participate in a DMR network by connecting its data port to a DMR modem. Handheld DMR radios can be bought for as low as 120 Euro and are very cost-effective.

The DMR standards also specify the coding and transmission of Internet Protocol version 4 (IPv4) packets, which leads to the assumption that DMR radios can be used as IP routers in a wide area wireless network. During the experiments, IP routers interconnected by DMR data links were constructed and will be presented and demonstrated in this paper, although the net bitrate offered by the DMR coding standard is far lower than 9600 bps and the applicability of IP services is limited.

DMR used for signalling of sensor readings and the operation of remote devices is a more promising application though, since such services are less hindered by bandwidth limitations and long transmission latency. For Internet-of-Things (IoT) and sensor network application, the DMR radio handset can be employed as a User Agent (UA) for the presentation of events to the operator (in text or voice form) and for the operator to invoke actuators in the IoT network, as well as to allow the operator to act as a sensor and enter information through text messaging.

Moreover, most DMR radio models are equipped with a GPS receiver and are able to report their location to a predefined receiver. The IoT network may use this information to select or modify information sent to the radios, or to provide location based services to their users.

During a field operation, these combined services of the handset greatly enhances the Situational Awareness (SA) of the operators without imposing additional equipment or increased cost. The integration of DMR radio handsets into a sensor network with Message Queuing Telemetry Transport (MQTT) [2] protocol will be demonstrated and presented in this paper as a proof-of-concept. The MQTT protocol is used to build a publish-subscribe based distributed systems, which offers a loose coupling between the system components and a high abstraction level on the information exchange.

The programmed interface between the DMR and MQTT protocol is relatively low level, and enables us to overcome some of the interoperability problems between radios from different manufacturers, e.g., during exchange of text messages.

Apart from the DMR radio handsets, all the technology and software components used in the experiments are open source and non-proprietary.

There are other wireless communication protocols that could be studied during our experiment, e.g., LORA and NB-IoT. These are offered as communication units (modems) without a user interface, and they do not offer GPS positioning or voice communication. They are therefore not taken into account for this experiment.

The contribution of the presented efforts is a middleware solution for others to build on using their preferred equipment of sensors and radios. Most industrial solutions in this area are found to be vendor-specific stovepipe systems with little attention to interoperability.

The remainder of this paper is organized as follows: Section II will briefly present the technology aspects of DMR, followed by a description of the experimental design in Section III, and the choice of equipment in Section IV. Aspects of the Controller software design are discussed in Section V, while the efforts made on IP routing across DMR links are addressed separately in Section VI. Finally, Section VII presents a summary of the article and identifies future research activities.

II. A BRIEF PRESENTATION OF DMR

DMR is a digital communication standard developed and maintained by ETSI. The physical requirements are well suited for implementation in inexpensive hardware: The 4FSK coding for 9600 bps is available in, e.g., the ADF7021 IC from Analog Devices, which even includes small transceiver with transmission power up to 20 milliwatts.[3] The physical timing and signalling requirements need to be handled by a processor equipped with suitable firmware.

The transmission frames are divided in two independent time slots of 27.5 ms duration, each containing 264 bits. The signalling which takes place within a time slot is called a *burst*, while the structured information inside the burst is called a *block*. With necessary guard time, the two slots take up 60 ms transmission time. Two bit streams can flow independently using different slots, in the same or opposite directions. This division enables two simplex channels to be active at the same time, or a Single Frequency Repeater (SFR) operation using the two slots for in- and out-bound traffic. The two slots may also provide full-duplex communication, e.g., during connection to a telephone network.

One great advantage of DMR is the 12.5 kHz channel width, which facilitates a smooth migration from FM systems, since the licensed channels allocated for the FM service can be directly used by a DMR network. An older digital radio system also standardized by ETSI, TETRA, does not offer this advantage since it requires a large channel bandwidth.

As the gross bit rate of 9600 bps is divided in two time slots, and with bits for synchronization, error detection and signalling, the bit rate for voice communication is reduced to approx. 3600 bps, which requires a codec with heavy compression. The resulting sound quality is adequate for spoken language, but easily garbled if not spoken relatively clearly.[4] The choice of codec is not specified by ETSI, but the proprietary Advanced Multiband Excitation (AMBE+2) [5] algorithm seems to be the de facto standard.

During data communication, a thick layer of error correcting coding (Block Product Turbo Codes, BPTC) [6] reduces the number of information bits per time slot from 196 to 96, as seen on Figure 1. 96 bits every 60 ms minus transmission time for preambles, header blocks and checksums yields a net maximum transmission rate on 1300-1500 bps. This bandwidth is too low for general IP-based delay-sensitive applications like web access, and leaves the DMR link suitable mostly for delay-tolerant tasks like e-mail, sensor readings and certain SA applications. Beside the transfer speed, one also needs to consider the acceptable duration for a shared channel to be occupied by data traffic. The radio turnaround time will also have to be taken into regard, since it will affect the Round Trip Time (RTT) of a simplex channel and the net throughput of a Transmission Control Protocol (TCP) connection.

DMR radio handsets include a short text messaging service (aka. SMS), which can be sent over a DMR data link in variety of ways, using a variety of character sets. The DMR specifications are vague on these details, and the resulting interoperability between manufacturer's equipment is weak.

A. Talkgroups and DMR-ids

Data and voice transmissions are given *addresses*. Each radio is configured with a unique DMR-id, and transmissions addressed to a DMR-id will only be received by this individual unit. Transmissions can also be sent to *Talk Groups* (TG), which is a way to separate radios into "Communities of Interest". Radios are normally only member of one TG and receives transmissions addressed to that TG in addition to its individual DMR-id. The radio will be allowed to change TG through the user interface. TG could be regarded as similar to the concept of *topics* in a Publish-Subscribe system.

B. Structure of data transmission

In order to send an IPv4 packet from one radio to another, it is first prepended with a 96 bits header block containing DMR-id of sender and receiver, transmission type, confirmation request etc. The IPv4 packet is split into 96 bits fragments which are placed in subsequent data blocks. The last block is padded to correct length and includes a 32 bit Cyclic Redundancy Check (CRC) checksum. Each block is coded using Turbo Coding which extends it length to 196 bits.

As seen on Figure 1, the 196-bit Turbo Coded block is split into two parts which are separated with a 68 bit synchronization and signalling field, bringing the total size of the burst to 264 bits. The signalling field is not Turbo coded so the transmission type of the burst can be observed without decoding the entire block.

When the IPv4 packet is addressed to a DMR unit, the IPv4 addresses used are derived from the DMR-ids of the sending and receiving units, by giving the first 8 bit the value `0xC0` and the DMR-id, regarded as a decimal number, as the 24 next bits. For sending to a TG, a multicast IPv4 address is formed by using `0xE8` as the first 8 bits.

C. Structure of text messages

As earlier stated, the structure and service aspects of text messages show variations between radio manufacturers and many interoperability problems are observed. Equipment from Motorola is observed not to use IPv4, but to embed the text directly into a data block. Other manufacturers are using UDP/IPv4 with a reserved UDP port (port 5016 is recommended by ETSI, port 4007 is also being observed). The syntax of the messages reveals frequent use of the UTF-16LE character set (although ETSI specifies UTF-16GE for

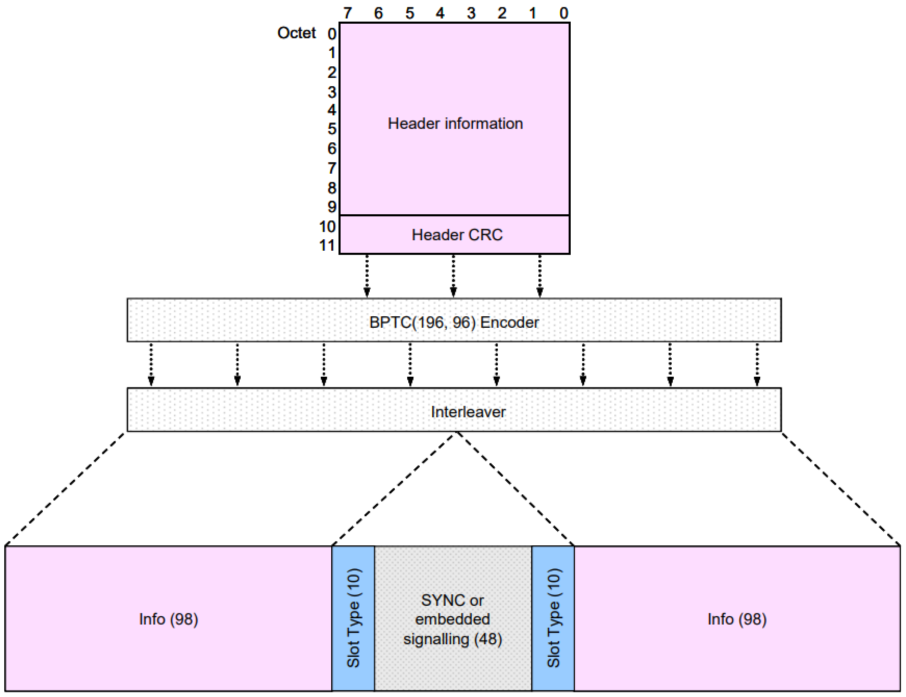

Figure 1. The burst structure of a data header block. Source: [1]

text messages) with leading characters of unknown function, so this part is totally left to the manufacturer to decide.

D. *Structure of voice bursts*

Voice transmissions are treated a little differently, where the start and end of a transmission are marked with data blocks indicating addresses etc, and the voice blocks in between is a mix of 4 voice blocks for each sync block. The actual sound information is coded with the proprietary AMBE+2 codec.

The transcoding of AMBE+2 sound to coding standards used in Voice over IP (VoIP) systems, e.g., the G.729 [7], would certainly add interesting opportunities to the experiments. Due to the proprietary status of AMBE+2, this effort has been left for future research.

III. Experimental set-up

Figure 2 illustrates how DMR may be employed in an IoT/sensor network configuration. The MQTT protocol for Publish-Subscribe (PubSub) distribution was chosen as a backbone for communication between sensors, actuators, decision makers and UAs. Decision makers are actors which contribute to the value chain by processing published data into a refined form for publishing to actuators or other decision makers. The DMR link on the right is shown used as connection between two IP routers, and the link on the left for communication with ordinary radio handsets for user information and actions. The two roles of the DMR links are quite different; While the IP router connection is purely a network layer service, agnostic of what is happening at the transport and application layer, the interface to the MQTT broker is fitted with programming code specific to the information messages being published and subscribed to during system operation. For the traffic across the IP router (right side), the MQTT-SN protocol was chosen. The MQTT-SN is designed exactly for this type of environment, and employs a compact UDP payload for reasons of brevity and fewer round-trips.

The IP router connection has been tested for transmission rate, latency and RTT. It is not expected to successfully support general IP-based applications, but remain useful for delay-tolerant applications and simple application protocols.

IV. Choice of DMR radio equipment

This section describes the process of equipment selection. The radio which will serve as a part of the DMR-IoT interface will see requirements quite different from a portable communication handset.

A. *Using DMR handsets*

All DMR radios provide voice communication and text messaging, more expensive units also provide arrangements (cable) for connecting to a PC through its USB port. For data communication, the radio connection will appear as an MS Windows virtual network interface, which can be operated either by special applications for file transfer and text messaging, or sometimes through standard networking programs like `ping`, `ftp`, `ssh`, etc. This arrangement allows two Windows PCs to communicate through the DMR link, but does not offer a link between networks (like a router is supposed to) since no physical Ethernet interface is provided. The dependence on the Windows OS also rules out applications that require the use of other OSes.

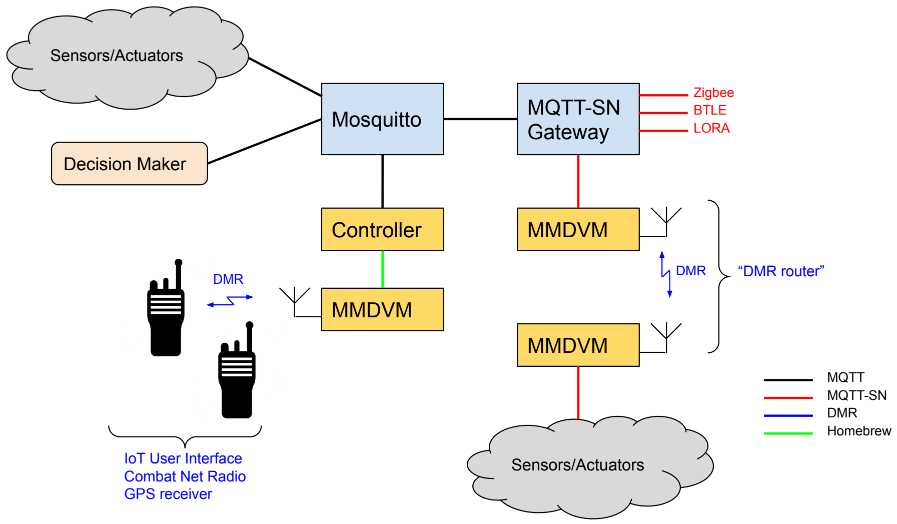

Figure 2. The design of the experimental network

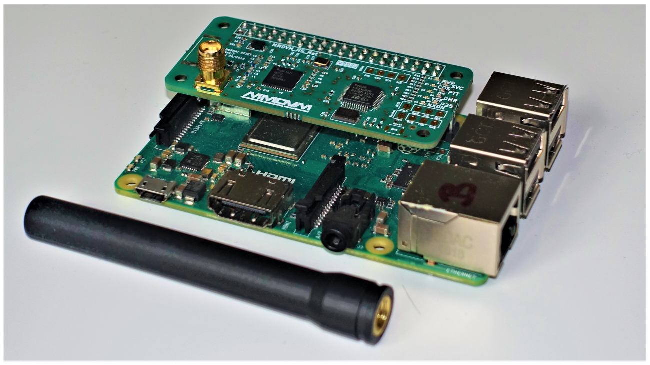

Figure 3. Raspberry Pi with MMDVM-hat

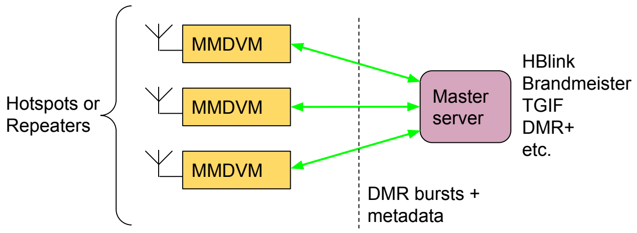

Figure 4. Structure of the DMR hotspot/repeater network as found in the Brandmeister system

B. Dedicated IoT modules

There are DMR radio modules available for IoT purposes [8], with connectors and an application protocol through which text messages can be sent and voice communication can be set up. These units are relatively inexpensive (e.g., DMR-828 from NiceRF) but provides limited functionality and rather low transmission power (2 W). They are probably exposed to the numerous interoperability problems related to how radios transmit text messages, and have not been studied for the use in this experiment.

C. MMDVM radio modem

In the amateur radio community, equipment called *hotspots* are used to connect DMR radios to world wide repeater networks in order to provide global connectivity. Hotspots are found as simplex link nodes or as repeaters, but with a low transmission power suited for private use within a household. A hotspot is equipped with a simple DMR radio module with up to 20 mW transmission power, and an IP interface for connecting to the repeater network through Internet. The Internet connection carries all voice and data traffic to a server which dispatches the traffic to other hotspots (and public DMR repeaters) in order to connect to other radios. The principle of this arrangement is shown on Figure 4.

The hardware of a hotspot will most often consist of a Raspberry Pi single-card computer and a "hat" of a MMDVM (Multi-Mode Digital Voice Modem) unit based on the ADF7021 IC and a micro controller with firmware for the operation of the DMR protocol, as shown on Figure 3. The MMDVM is also available without a radio module, for wired connection to radios with better receivers and higher transmission power.

D. The advantages of a low-level control protocol

The protocol between the hotspot and the Internet servers is called *MMDVM Homebrew* (shown as green lines on Figure 4) and is partially documented and easily reverse engineered. For voice, text and data traffic sent and received, the "raw" data bursts in the time slots are transmitted (layer 2 PDUs) together with some metadata. In the experiments described in this paper, the component called "Master server" has been replaced with a "Controller" component which emulates the Master server interface to the MMDVM and provides a service for conversion to/from the MQTT protocol. The Controller software was developed by the author as a part of this experiment, and its design will be described in the next section.

Although there is relatively more work in coding and decoding layer 2 Protocol Data Units (PDUs), it also allows the interpretation and construction of messages used by different radio manufacturers. For the sake of better interoperability between radios from different manufacturers, using this low-level interface is an appealing alternative, also due to the low cost of the MMDVM module and the availability of related source code. One serious drawback with the Homebrew protocol is that it is UDP-based, which means that there is no easy way to throttle the traffic from a connected computer, which may lead to buffer overrun problems.

V. Controller software design

The chosen configuration, as shown in Figure 2, employs a Controller sitting between the MQTT broker service (Mosquitto) and the MMDVM unit as an adaption layer. Its responsibilities are:

- Emulate the Homebrew protocol to the MMDVM
- Construct and deconstruct the DMR bursts
- Route IPv4 packets between the MMDVM and the connected network
- Present an interface to the MQTT protocol and Mosquitto, the MQTT broker
- Construct text and voice messages from received MQTT publications
- Construct and publish MQTT publications from received text messages and GPS locations
- Record and playback voice messages

The chosen programming language is Python3, mostly due to the availability of library modules for operating on DMR PDU structures and interface to MQTT brokers, but also for its convenient programming style. The Controller is running a Linux OS, chosen for its easy access to raw sockets, threads and locks.

Figure 5 shows an interaction diagram with the flow of information between the different components. The color of the invocation lines indicates the protocol.

A. Coding and decoding

The process of DMR PDU coding and decoding is the most complicated part of the Controller program. There are Python libraries for parts of these processes (dmr_utils3), but they are mostly targeted towards voice communication and had to be modified for the purpose of constructing and deconstructing data traffic.

During data traffic, the IPv4 packet is split up in a series of blocks, each holding a 12 bytes fragment. A header block (cf. Figure 1) is prepended and the final data block is padded and given a 32 bit CRC value. The blocks also carry fields for synchronization and signalling, and the different parts are given different error correction and turbo coding. These tasks must be handled by the Controller, since the MMDVM simply passes the blocks on to the radio interface.

The construction of text messages uses much of the same code, since the messages are contained in UDP/IPv4 PDUs. As mentioned earlier, the UDP and IPv4 header details and the syntax of the UDP payload depend on the radio manufacturer so programming code must be adapted to gloss over interoperability problems.

B. Synchronisation and flow control

The Homebrew protocol is simple and does not offer any flow control mechanisms, since all messages are included in UDP segments and no receiver feedback is provided. Possible buffer overrun problems could be mitigated through a choking mechanisms in the Controller, with the risk of creating premature timeouts in communication endpoints in the connected networks.

For short text messages and voice clips sent to the MMDVM, buffer overrun may not be a frequent problem. When the MMDVM engages in IP routing, however, additional steps may have to be taken, which will be discussed in Section VI.

C. The relation between DMR TGs and PubSub topics

On the matter of interfacing a DMR system to a PubSub distributed systems, the mapping between DMR Talk Groups (TG) and PubSub Topics must be addressed. PubSub topics in the case of the MQTT protocol are organized as a taxonomy. Clients can subscribe to a single topic or to a subtree of topics. Clients can thus choose to subscribe to all information in a general area of topics, or to specific topics with a narrow focus. Likewise, clients can publish information using any topic to indicate the nature and specificity of the information. A topic taxonomy which efficiently reflects the business logic is a core element of PubSub system design.

In a DMR system, the concept of Talk Groups can be used to indicate a topic of a voice or text transmission, but TGs are not hierarchically organized. Besides, the typical design of a DMR radio does not allow a radio to associate with different TGs for voice and text. Given that the radio is being used also for spoken communication in an operational environment, the TG does not appear as a useful tool to specify groups of receivers for publications. Individual transmissions to each radio, based on their DMR-id, will only be defensible on a small scale of operation due to the scarcity of communication capacity.

On the other hand, voice and text messages sent from a DMR radio can be annotated with any TG, which would be mapped into a MQTT topic before rewriting it to a common publication syntax and passed on to the PubSub broker. The radio is equipped with an address book where names can be given to TGs, which allows the operator to relate to meaningful topic names rather than TG numbers.

D. Voice recording and playback

In the absence of a proprietary software codec for AMBE+2, it is still possible to use the AMBE+2 codec embedded in the DMR radio handset for the recording and playback of voice messages.

All voice traffic will be sent from the MMDVM to the Controller over the Homebrew protocol as DMR bursts and can be stored there in the AMBE+2 coded form. Likewise,

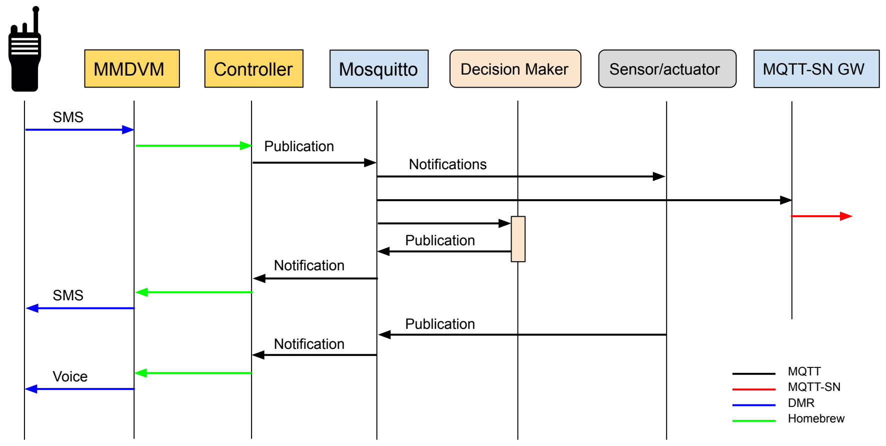

Figure 5. Protocol interaction diagram. The coloring of the invocation lines indicate the protocol used.

stored voice bursts can be sent from the Controller to the MMDVM which will be transmitted and received by the radios within range.

This arrangement does not allow for voice traffic across technology domains, but still is considered a useful addition to the otherwise text-centric information exchange. A set of voice messages can be recorded and assigned an index as a part of the system configuration. The voice clips can be sent to a radio unit or a TG as responses to MQTT notifications. The present implementation allows any radio to record and index voice clips, which allows for interesting voice information exchange: Voice clips can represent the most recent status or observation recorded by a field operator, which can be announced to other operators on a variety of conditions, including interactive queries.

VI. IP ROUTED ACROSS A DMR LINK

For the final part of this presentation, the use of a DMR link for general IP routing will be explained and demonstrated. The principle for such an arrangement is quite simply explained: The Controller uses a raw socket to take in full Ethernet frames, and decides if they should be routed across the DMR link. This decision is made on the basis of both MAC- and IP addresses. Frames addressed to the Controller IP interface are not routed, nor is any multicast or broadcast frames. The IP packet is split into a series of DMR data blocks as explained in Section II-B which are sent to the MMDVM using the Homebrew protocol. The receiving MMDVM restores the IP packet from the DMR data blocks and sends it through a raw socket interface.

Several problems were expected to appear when a simplex DMR channel is used to support two-way IP traffic, possible several two-way streams. No media access protocol is defined in DMR other than a “polite” mechanism which is a primitive form of Carrier Sense. No collision avoidance protocol is in place, nor any backoff method, etc.

Moreover, the equipment used for the experiment activated the transmitter even if the receiver was actively receiving a signal (like how Push-to-talk (PTT) on a radio would override the receiver). Slow turnaround in the radios also caused problems for the reception.

TABLE I
RESULTING RTT AND BITRATE OVER SIMPLEX AND DUPLEX DMR LINKS

Link type	RTT (secs)	Bitrate (bps)
Simplex link	3.6	430
Duplex link	2.4	1100

Even though a simple `ping` command would successfully operate on the link, it became clear that a duplex link is necessary, which could operate in both directions simultaneously. Two way communication can be accomplished using the two time slots, or using two frequencies. The MMDVM units may have two independent radios, but their firmware does not support such configuration. The choice was made to use two pairs of MMDVMs to make two independent DMR links on different frequencies, and to modify the Controller software to use the two connected MMDVMs for traffic in opposite directions. This arrangement is illustrated in Figure 6.

This arrangement removed the channel access and collision problems, as well as problems related to radio turnaround time. Remaining problem were:

- Premature timeouts due to low transmission rate, both on the TCP protocol and certain application protocols.
- Buffer overrun in the sending MMDVM, due to lack of synchronized flow between the Controller and the MMDVM.

These problems affected the ability to use the `iperf` utility to measure communication rate. The familiar `ping` command combined with a stopwatch allowed for the estimation of Round Trip Time (RTT) and bit rate:

The average RTT for default `ping` message size (80 bytes), and the average bit rate when using packets of 240 bytes are shown in Table I. With a bit of patience, it was possible to fetch websites and make `ssh` connections over this duplex link, but in practice it would be better suited for short signalling

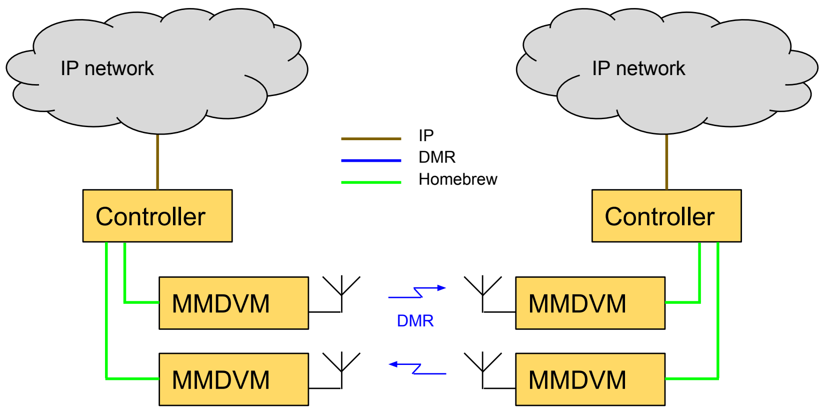

Figure 6. IP router arrangement using separate DMR links for opposite direction.

and short messages in a stop-and-wait type protocol. Any streaming application will fail due to the lack of MMDVM-Controller synchronization mentioned above.

VII. Conclusion

This paper has demonstrated a proof-of-concept effort on the integration of DMR technology with a sensor network using Publish-Subscribe technology and the MQTT protocol. The integration arrangement allows sensor readings to be reported to individual radios or to talk groups, using text or voice. It also allows the radio operators to send text messages and GPS position reports to the MQTT broker as publications. Additionally, the experiment demonstrates a voice recording service as well as IP routing across DMR links.

The combination of these services offers an improved situation awareness to field operations, better utilization of the handheld radios, and access to human observations into a sensor networks.

The use of MQTT protocol is used as an example for information dissemination in an IP network, other forms for distributed middleware will surely work in a similar manner.

For this laboratory experiment to evolve into a more realistic field study, there will be a need to use equipment better suited for the task than the MMDVM firmware. Modification to the firmware could be one option, or using different hardware, e.g., dedicated IoT/DMR modules. Voice traffic exchange between DMR and VoIP systems requires the availability of a AMBE+2 codec and is an appealing extension to the present arrangement, which would also allow text-to-speech services to be implemented. Further studies will also look into the configuration and information presentation in a realistic field operation scenario, as well as usability aspects and interoperability issues.

References

[1] “The DMR Standards,” https://www.dmrassociation.org/dmr-standards.html, European Telecommunications Standards Institute, [Online: Accessed 29.07.2022].

[2] “MQTT Specification,” https://mqtt.org/mqtt-specification/, OASIS, [Online: Accessed 29.07.2022].

[3] “ADF7021,” https://www.analog.com/en/products/adf7021.html, Analog Devices, [Online: Accessed 29.07.2022].

[4] S. Campos Neto, F. Corcoran, J. Phipps, and S. Dimolitsas, “Performance assessment of 4.8 kbit/s ambe coding under aeronautical environmental conditions,” in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, 1996, pp. 499–502 vol. 1.

[5] “AMBE+2,” https://www.dvsinc.com/soft_products/ambe_p2.shtml, Digital Voice Systems, Inc., [Online: Accessed 29.07.2022].

[6] R. Pyndiah, “Near-optimum decoding of product codes: block turbo codes,” *IEEE Transactions on Communications*, vol. 46, no. 8, pp. 1003–1010, 1998.

[7] L. Li, Q. Huang, and W. Wan, “Design of converting low-rate speech codec between ambe and g.729,” in *IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013)*, 2013, pp. 185–188.

[8] I. Ganchev, Z. Ji, and M. O’Droma, “Designing a low-cost dmr module for use in m2m/iot applications,” in *2018 2nd URSI Atlantic Radio Science Meeting (AT-RASC)*, 2018, pp. 1–2.