A Protocol for Multicast Weather Data Distribution Over AX.25

Nick Luther, K9NL
1655 South Westhaven Drive
Oshkosh, Wisconsin 54904
nluther@ieee.org

Abstract

A protocol is described for use on top of AX.25 in order to form multicast, push-architecture, regulatory compliant, radio weather data links. This protocol is especially suited to disseminating NEXRAD weather surveillance radar data over 1200 baud AFSK VHF radio links. Further, a larger scale system for distributing weather data using various means, among them radio links using this protocol, is discussed along with the current status of its implementation. Using the defined protocol over a radio link along with other system components, it is possible to ingest NEXRAD data from fast, operational data sources and to relay that data to severe weather spotter resources in a timely fashion, even when those resources have no access to the Internet or other data network infrastructure.

Key Words

NEXRAD, AX.25, RDTP, NOAAPORT, SKYWARN

Introduction

In the United States many amateur radio hobbyists participate in the National Weather Service's SKYWARN™ program. Nationwide, this program has nearly 280,000 trained severe weather spotters [1]. While not all of these are amateur radio volunteers, it is evident that a large group of them in fact are. When considering the future direction of amateur digital communications, technologies supporting volunteer severe weather spotters should be seriously considered.

An architecture for a system which uses amateur packet radio as a physical link to disseminate real time NEXRAD weather surveillance radar data, as well as other relevant products, in support of SKYWARN operations was previously presented in [2]. Implementation challenges have since been resolved and detailed design has been completed. A protocol is now presented for use above AX.25 to support this data dissemination system. The protocol, called Radar data Datagram Transport Protocol over AX.25 (hereafter RDTP/AX.25), supports multicast or point-to-point topology, allows data to be pushed asynchronously as it is received from upstream sources, and maintains two-way communication for compliance with 47CFR97. Further, the status of this project is discussed with a focus on new developments since the publication of [2].

Design Inputs

AFSK and FSK modems with typical data rates of 1200 and 9600 baud are readily available in amateur equipment, especially when integrated into hardware Terminal Node Controllers (TNCs) or software AX.25 protocol implementations. Making use of this equipment simplifies the constructions of an RDTP/AX.25 station. However, the requirement that this protocol be used over AX.25 is introduced. Further, the maximum frame length is constrained.

The RDTP/AX.25 protocol then must meet these requirements:

- Be implemented on top of AX.25
- Operate at frame sizes less than 256 bytes
- Viably relay at least two 5 kB messages received every five minutes
- Support multicast operation to multiple receivers
- Maintain two-way communications
- Be designed to simplify implementation on both Windows®[1] and Linux®[2] operating systems

The protocol should be able to handle various types of weather data, as long as the individual message data lengths are reasonable. This should include radar and text data. The data flow is generally unidirectional, with a clear upstream and downstream side to any link. RDTP should recognize a server side, upstream with respect to data flow, and a client side, which is downstream with respect to data flow. The server will likely be a fixed radio station, and the client could be a fixed or mobile station. In a multicast situation, there could exist an arbitrarily large number of clients in a link.

Some additional discussion supporting these requirements is available in [2].

Protocol Overview

The previously stated requirements drive significant design decisions, which are discussed in this section.

The requirement to operate on top of AX.25 is established, however, the functional capabilities of AX.25 are not necessarily required, nor even desired. Thus the RDTP protocol is designed to make use of only AX.25 Unnumbered Information (UI) frames. Every frame transmitted includes the transmitting station call sign in the appropriate AX.25 field. This meets 47CFR97 requirements, but may generally be ignored by receiving software. The recipient call sign in the AX.25 frame is always stated as ASCII "RDTPC" for server-to-client frames and "RDTPS" for client-to-server frames. This convention resolves certain issues identified during implementation. The only other AX.25 feature functionally used by RDTP is the Frame Check Sequence (FCS). The current RDTP implementation assumes that the AX.25 stack will have dropped any corrupted data, and thus all data ingested by the RDTP software may be considered error-free. To increase compatibility with available AX.25 stacks, the AX.25 PID is always set to text, though it would ideally be set to a unique identifier for this protocol. For reference, the AX.25 protocol is specified in [3].

[1] Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries.
[2] Linux is the registered trademark of Linux Torvalds in the United States and other countries.

To realize maximum utility from low data rates, such as 1200 baud, compression is specified at the RDTP frame level, and is also available at the weather data level. The bzip2 algorithm was chosen as its freely available implantation, libbz2, was shown to perform very well in informal engineering confidence testing and is easily integrated into an RDTP software package on multiple operating systems. Additional compression details are presented in a later section of this paper. For more information on bzip2, see [4].

With these fundamental design details explained the discussion may proceed to protocol semantics. To organize the available data on the server side, the concept of a named data stream is defined. Now posit there exists an RDTP server sourcing weather data from unknown upstream sources, and there further exist one or more RDTP clients, and that all of these stations share the same physical link. In other words, they are all tuned to the same frequency. The server organizes incoming data into logical data streams. The clients all desire data product which are members of one or more of these data streams. At the high level, the client server exchanges will follow this procedure:

- The client waits to be polled, as it shall not speak until spoken to. However, if an unspecified dead air timer elapses, as in initial link establishment, it immediately contacts a known RDTP server requesting one or more data streams.
- The server acknowledges and accepts or denies the client's request for data. Other clients on the channel hear the acknowledgement and realize that it is unnecessary to transmit their own request, unless said clients desire to request additional data streams for which no acknowledgement has yet been received.
- When new data becomes available from upstream, the server classifies this data into a data stream, checks for an active request for that data stream, and if one is found then transmits the data in a manner such that all clients sharing the same physical channel may receive it.
- After transmitting data from a data stream, the server initializes a timer, which once elapsed, will purge the active request for any new data from that data stream. This will cause the channel to shut down until another client request is received.
- To keep channel access organized for the next round of data stream requests, the server will poll clients in an orderly manner. Clients will only speak after being spoken to, as previously stated, and will transmit any pending data stream requests in response to the poll. The process then repeats.

In practice, when data is not flowing, neither the client nor server will transmit. This allows for multiplexed access to the physical channel, and more specifically allows someone else to use the frequency. To establish a link, a client detects dead air and transmits its request. When the last client in a group is taken off air, the server will eventually lose all active data stream requests, and will then stop transmitting. This is the ideal behavior for operation in the amateur radio service, and is most polite to all band users.

Detailed Protocol Specifications

The requirements and high-level design of the RDTP protocol have been discussed to the greatest extent reasonable within the scope of this paper. The low level details of the protocol will now be presented.

Conventions

These conventions are followed in this section:

- All indices are zero-based.
- Bits are numbered by order, with the most significant bit assigned the highest order number, and the least significant bit the lowest order number. The most significant bit is transmitted first. A numerical value for a byte should be represented over the physical channel following this convention.
- A string of bytes is numbered beginning with the first byte in the string, which is the first byte transmitted. This same notion for strings is used for data structures of numbered bytes.

Call Signs

RDTP call sign fields are filled as shown in Table 1, except for a deviation in the RDTP Frame Layer Header.

Table 1: RDTP Call Sign Field Format

Bytes		Bits	Description	Value	Notes
First	**Last**				
0	6	All	Call sign ASCII text	ASCII string, unused bytes filled with 0x00.	Left (byte 0) justified.
7	7	7-4	Reserved	0 bits	Always fill with zeroes.
		3-0	Station SSID	16-bit unsigned integer SSID value.	

Frame Layer

The RDTP Frame Layer exists directly on top of the AX.25 UI frame. It functionally replaces AX.25 connected mode framing with special framing designed for the RDTP application. The most important function of this layer is to allow messages to be broken into frames and then reassembled. To support this, a message sequence number and frame sequence number are defined. Each individual message is numbered and broken into frames. Each frame within that message is then numbered sequentially. These two numbers allow receiving software to group frames into their respective messages, and then to reassemble them in order, even if they were transmitted out of order or repeated later.

Also notable is that this is the level of RDTP that supports data compression and forward error correction. A compression code identifies the data as compressed or not compressed, and indicates the compression algorithm used. When the compression code indicates that compression is in use, only the data section of the frame is compressed, to exclude all of the headers. For forward error correction, simple N+1 parity of the data section only is specified. Setting the N+1 parity flag indicates that a frame is the parity frame for the message indicated by its message sequence number.

The RDTP Frame Layer specification is presented in Table 2.

Table 2: RDTP Frame Layer Header and Payload

Bytes		Bits	Description	Value	Notes
First	**Last**				
0	3	All	Protocol identifier sequence	Literal ASCII ‘R’ ‘D’ ‘T’ ‘P’	
4	4	All	Protocol version code	Literal 0x00	This field is intended to ease protocol version changes.
5	5	7	From call/SSID present flag	1: From/Transmitting station call sign present 0: From call sign not present	
		6	N+1 parity data flag	1: This frame is N+1 parity data 0: This frame is not N+1 parity data (a normal frame)	
		5-4	Reserved	0	Always fill with zeroes
		3-0	From station SSID	4-bit unsigned integer SSID value	Used only if byte 5, bit 7 is set
6	11	All	From call sign without SSID	Call sign as specified in Table 1 without SSID	Used only if byte 5, bit 7 is set
N-5 6/12	N-5 6/12	All	Message sequence number	Increments per message from 0x00 through 0xFF, then rolls around back to 0x00.	
N-4 7/13	N-4 7/13	All	Frame sequence number	Increments per frame from 0x00	Reset at 0x00 for each message
N-3 8/14	N-3 8/14	All	Frames in message	Total number of frames in this message minus one.	Subtracting one makes use of the value 0x00 and slightly expands the maximum data length RDTP is capable of handling
N-2 9/15	N-2 9/15	All	Compression code	0: No compression 2: Bzip2	Use of libbz2 is recommended. Compression applies only to the payload.
N-1 10/16	N-1 10/16	All	Frame layer payload length	The length in bytes of the payload section of this RDTP frame.	For example, 0x00 would indicate no data section present, but would likely never be used.

Bytes		Bits	Description	Value	Notes
N	N + data length		Frame layer payload (data section)	A segment of a sequence of RDTP logical entities. These segments will be concatenated by the receiver in the order of frame sequence number, lowest sequence number first.	This is the content of the RDTP message. It must consist of RDTP logical entity blocks.

Logical Entity Layer

The RDTP Frame Layer replaces certain AX.25 functionality to meet the requirements of this protocol. With this layer specified, it is now possible to split and frame messages for radio transmission, so the discussion can continue to the content of those messages. The RDTP Frame Layer's payload is one or more concatenated RDTP Logical Entity Blocks (LEB). The Logical Entity Blocks are the meaningful content of a message. In this section, critical blocks will be discussed and specified one at a time.

Table 3 presents the specification for the Data LEB. This LEB is used to transmit new weather data from a server to a client. The payload is the data message itself, in original format from its source or compliant to the same specifications if keeping the original format is not possible.

Table 3: Data LEB

Bytes		Bits	Description	Value	Notes
First	Last				
0	0	All	LEB ID	Literal 0x00	Identifier for this type of LEB
1	7	All	Data stream name	Free ASCII text, defined at the implementation or system level	System administrators will specify data stream names.
8	8	All	Compression code	0: No compression 2: Bzip2	Use of libbz2 is recommended. Compression applies only to the payload.
9	10	All	Length of data	16-bit unsigned big-endian integer indicating the length of the data payload in bytes.	
11	N = 11 + data length	All	Data payload	This is the actual weather data received from the information source.	

As previously discussed, a server cannot transmit data unless a client requests the data stream and the request is accepted. A client may send to a server a Data Request LEB to perform this function. This LEB is specified in Table 4.

Table 4: Data Request LEB

Bytes		Bits	Description	Value	Notes
First	**Last**				
0	0	All	LEB ID	Literal 0x01	Identifier for this type of LEB
1	7	All	Server call sign	RDTP formatted call sign of the server which the data stream(s) are being requested from	The AX.25 address fields are not used due to implementation complications.
8	8	All	Number of data stream names	Length of the array of data stream names in the payload	
9	9 + 7N	All	Data stream names	Array of seven-byte data stream names. Each is left justified and right-filled with 0x00 bytes as necessary.	This is a simple C array.

A server may accept and acknowledge or deny a Data Request LEB received from a client. The Request Ack LEB and Request Denied LEB are used for this purpose, respectively. They are defined in Table 5 and Table 6.

Table 5: Request Ack LEB

Bytes		Bits	Description	Value	Notes
First	**Last**				
0	0	All	LEB ID	Literal 0x07	Identifier for this type of LEB
1	7	All	Call sign of client being acknowledged	RDTP formatted call sign	
8	8	All	Number of data stream names	Length of the array of data stream names in the payload	
9	9 + 7N	All	Requested data stream names being acknowledged	Array of seven-byte data stream names. Each is left justified and right-filled with 0x00 bytes as necessary.	This is a simple C array.

Table 6: Request Denied LEB

Bytes		Bits	Description	Value	Notes
First	**Last**				
0	0	All	LEB ID	Literal 0x0C	Identifier for this type of LEB
1	7	All	Call sign of client being denied	RDTP formatted call sign	

Bytes		Bits	Description	Value	Notes
8	8	All	Number of data stream names	Length of the array of data stream names in the payload	
9	9 + 7N	All	Requested data stream names being denied	Array of seven-byte data stream names. Each is left justified and right-filled with 0x00 bytes as necessary.	This is a simple C array.

It has been previously stated that RDTP clients shall not speak unless spoken to, unless an exception applies. In normal operation, polling is required. A Poll LEB is defined for an RDTP server to poll clients. Clients shall only respond when a Poll LEB is received giving the specific client permission to transmit. The Poll LEB is specified in Table 7.

Table 7: Poll LEB

Bytes		Bits	Description	Value	Notes
First	Last				
0	0	All	LEB ID	Literal 0x06	Identifier for this type of LEB
1	1	7-4	Poll type	0: Level poll 1: Call sign poll 2: Wide open poll	Any client may respond to a wide-open poll or level 0 poll. Level and call sign poll discussions are located with their respective data fields.
		3-0	Level	Unsigned 4-bit integer 0 for call sign or wide-open polls. Otherwise, an unsigned 4-bit integer level for a level poll. Only clients with this access level or higher may respond.	In practice, multiple level polls may be used, with the server stepping down from 15 to 0, likely skipping many steps at a time.
2	8	All	Call sign being polled	Call sign poll: RDTP formatted call sign Other polls: This field does not exist, and the LEB will only be two bytes long.	A call sign poll is used to poll a specific client. This functionality is useful to isolate clients that have been heard recently and prevent interference.

The Poll LEB introduces the concept of an access level. Certain systems may have multiple clients of different priorities. An example would be a system with a client station at an Emergency Operations Center, a station at another communications facility, one or more mobile stations for severe weather

spotters, and one or more stations operated by general interest amateur radio hobbyists that provide little or no operational benefit. The access level feature allows the requests of these stations to be prioritized.

Ideally, each client would be properly programmed with its access level, assigned by a system administrator. In practice, many erroneous configurations are anticipated, so a special LEB, called the Access Level Is LEB, is defined to advise clients of their access level. A client must comply with the assigned access level if it is capable of responding to level polls. This LEB is specified in Table 8.

Table 8: Access Level Is LEB

Bytes		**Bits**	**Description**	**Value**	**Notes**
First	**Last**				
0	0	All	LEB ID	Literal 0x09	Identifier for this type of LEB
1	7	All	Client call sign	RDTP formatted call sign	
8	8	7-4	Reserved	0	Always fill with zeroes. This is a 4-bit unsigned integer.
		3-0	Access level	4-bit access level of the client being addressed.	

Presentation of additional LEB specifications is beyond the scope of this paper. The LEB specifications presented herein establish the minimum functionality for an operational system. Additional LEB types are specified in a draft RDTP specification available at [5]. Readers are cautioned that this is a draft specification and will be revised in the future.

System Integration and Testing

The RDTP protocol was designed for integration into a system for ingesting weather data and distributing it to users who can realize benefits from that data. As such, some discussion of an integrated system is appropriate for this paper. With much assistance, the author has established a weather data distribution system [6] that is capable of:

- Ingesting real-time NEXRAD, GOES, text, and other data from the U.S. National Weather Service's NOAAPort distribution network. Fore more information on NOAAPort, see [7].
- Relaying that data over connections on top of TCP/IP or AX.25 (the subject of this paper).
- Building relay networks which may have multiple upstream data sources for redundancy
- Processing that data and presenting it in a meaningful way, displaying it to end users in a Graphical User Interface.
- Performing various other useful tasks such as: archiving received data, automatically generating mobile phone text message and other alerts from received National Weather Service statements, and driving Internet web sites with real time weather data.

RDTP has been used intermittently in this weather data distribution system for many years. Users have provided feedback that RDTP performance has been excellent. With reasonable processing, it has been found that a frame of NEXRAD data can be ingested from NOAAPort, stripped to the quarter of the frame which is of interest, relayed via a 1200 baud RDTP link, and processed all before that same frame

of NEXRAD data is available on Internet web sites. As more Internet weather sources are developed and the current sources advance this statement will likely no longer be true, but it has been informally tested and found to be true as recent as two years prior to the writing of this paper. More information is available in [2].

Project Status and Future Work

The RDTP protocol is one small part of an ongoing project to ingest, distribute, and process weather data in real time, with special interest to NEXRAD data. The author and colleagues originally referred to this project simply as the "Radar Project," but it has since outgrown that term. As previously discussed, a weather data distribution system has been established which is capable of ingesting many different types of weather data and relaying those data through complex networks consisting of Internet and AX.25 links to end users. To ease implementation, all of the software tools that compose this network are available for both the Windows® and Linux® operating systems. Indeed, this is a stated goal for the RDTP design. To the end user, a simply graphical application handles connections to Internet and AX.25 data sources and automates data processing. Deployment of this system to end users is nontrivial, it is also not very difficult.

As of this writing, two NOAAPort satellite receive sites are in operation and are redundantly linked via the Internet through a backbone node. Additional fan-out Internet distribution nodes are available to remove loading from the backbone. An RDTP server is operated in the author's hometown every summer during critical time periods, where excellent system performance has been observed.

Most relevant to this paper, future work on this project will further define the RDTP protocol and add useful features. Many additional LEBs have been defined for anticipated features but resources have not yet become available to implement and experiment with those envisioned features. Among the highest priority desired RDTP features is a more advanced forward error correction scheme, such as Reed Solomon coding. Additional work will focus on weather data processing software to expand the types of data capable of being processed beyond simple text, radar, and satellite data into graphical depiction of model output, radiosonde observations, additional types of radar and satellite data, and any other data types of interest. Further, future work will yield additional features for various components of the system to perform useful tasks. Severe weather alerting has been identified as a specific area where additional features would be valuable.

Most importantly it must be stated that all of the core development on this project has been completed, and that a much larger, more reliable weather data distribution system may be created without any further development work. The future work of the highest priority is recruiting new volunteers to implement NOAAPort satellite receive sites and RDTP servers to grow the network infrastructure. This provides substantial benefits to the project, notably including a large body of troubleshooting resources for future development, and increasing morale and providing greater value to volunteer software developers.

Acknowledgements

The author would like to extend special gratitude to Aaron Heise, KB9QWC, for his extensive contributions over the life of this project. These contributions include, among other things, assistance

developing the specification of the RDTP protocol, development of software tools for integration into the larger scale weather data distribution system, troubleshooting of every weather software tool the author has developed, and in general providing extensive engineering insight concerning all aspects of this project.

The author would also like to thank Stephen Williams, KB9RLF, and Gerry Parks, N9OEW for their dedicated contributions of equipment and time to the mission of building a weather data distribution system and also for substantial software troubleshooting assistance, including especially the author's RDTP implementation software package.

Conclusions

The RDTP protocol and other system components developed by the author and colleagues present a complete weather data delivery solution for severe weather spotters and other weather enthusiasts. This solution overcomes the limits of the reach of the Internet and may be implemented at minimal cost. Indeed, a network already exists implementing the concepts discussed. As the software tools in use support redundant operation, the network reliability will increase as new volunteers establish and link new infrastructure. Therefore, interested parties are strongly encouraged to follow references in this paper and contact the author for further discussion. With network growth and continued development effort a private, volunteer-operated weather distribution network is possible which will rival not only gratis but also costly commercial solutions that are currently available. This is the ultimate goal and future vision of the author's long-standing project to develop weather data distribution tools.

A GPL-licensed implementation of the RDTP/AX.25 protocol is available from the author.

References

[1] NOAA: National Weather Service, Office of Climate, Water, and Weather Services, "What is SKYWARN?," National Weather Service web site, Available http://www.weather.gov/skywarn/, Accessed 28/Jul./2008.
[2] N. Luther, "Disseminating NEXRAD Data via Packet," *QST*, vol. 88 (no. 9), p. 81, Sep. 2004.
[3] W.A. Beech, D.E. Nielsen, and J. Taylor, "AX.25 Link Access Protocol for Amateur Packet Radio,' Tucson Amateur Packet Radio Corporation web site, version 2.2, Jul. 1998, Available http://www.tapr.org/pdf/AX25.2.2.pdf, Accessed 29/Jul./2008.
[4] J. Seward, bzip2 web site, see http://www.bzip.org/.
[5] N. Luther and A. Heise, "RDTP/AX.25 Protocol Specification," DRAFT version 0.3, released Apr. 2006, Available from Winnebago County, Wisconsin ARES®[3]/RACES web site [online], follow link from http://www.ecwec.org/noaaport.shtml.
[6] N. Luther, "NOAAPort Dissemination Network," Winnebago County, Wisconsin ARES®/RACES web site, Available http://www.ecwec.org/noaaport.shtml.
[7] NOAA: National Weather Service, Office of Operational Systems, "NOAAPort User's Page," National Weather Service web site, Available http://www.weather.gov/noaaport/html/noaaport.shtm

[3] ARES is a registered trademark and a program of the American Radio Relay League, Incorporated.