Comparing 2-Meter Packet Radio Data Unicasts and Multicasts

Paul D. Wiedemeier, Ph.D., KE5LKY
Assistant Professor of Computer Science
The University of Louisiana at Monroe
Computer Science and Computer Information Systems Department
College of Business Administration
700 University Avenue, Monroe, Louisiana 71209
318-342-1856 (Work) or 318-396-1101 (Fax)
wiedemeier@ulm.edu or KE5LKY@arrl.net

Abstract

Fairs, concerts, marathons, and festivals are public events where the local amateur radio club often assists the sponsoring organization(s) by providing communication services. These celebrations however, can be tarnished if a parent cannot find their child or a child cannot find their parent(s). During such situations, one site orally broadcasts the description of the child to all other sites participating a directed network. Unfortunately, oral descriptions can be ambiguous. A solution would be to transmit a digital image of the child along with the oral description. If all sites participating in a directed network have packet radio stations, then an optimal data transmission method would be for one site to transmit the digital image to all other sites simultaneously. This type of transmission is referred to as multicast and our data show that multicast file transfer tools should be used by the amateur radio community when transmitting digital data, imagery or otherwise.

Key Words

Packet Radio, Unicast, Multicast, Pure-FTP, Udpcast, Digital Image, and ASCII File.

Introduction

Many cities throughout the United States host annual festivals, etc. that hundreds or thousands of people from the community attend. Often, the local amateur radio club assists the sponsoring organization(s) by providing communication services. However, two potentially dangerous situations can occur during these celebrations; (1) a parent cannot find their child (i.e. missing child) or (2) a child cannot find their parent(s) (i.e. lost child). In either situation, it is imperative for network control (NC) to facilitate the quick dissemination the child's description to other sites participating in the directed network. Figure 1 shows a written description of a hypothetically missing or lost child and his associated image.

Male child; name Connor; age 3; light brown hair; hazel eyes; approximately 39 inches tall and 40 pounds in weight. Connor is wearing a blue baseball cap, a green shirt, a blue sweatshirt, blue jean shorts, white socks, and blue tennis shoes.

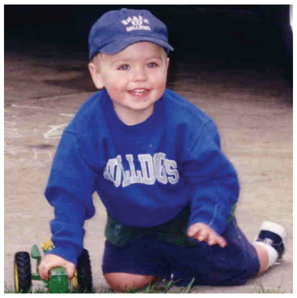


Figure 1: Written description of a hypothetically missing or lost child and associated digital image.

Unfortunately, the problem with any oral or written description is that it can be ambiguous. A solution would be to transmit a digital image of the missing or lost child along with the oral description, because, as is often said, "A picture is worth a thousand words." The implementation of this solution, however, requires that each site participating in the directed network have access to a packet radio station, consisting of a computer, a terminal node controller, and a transceiver.

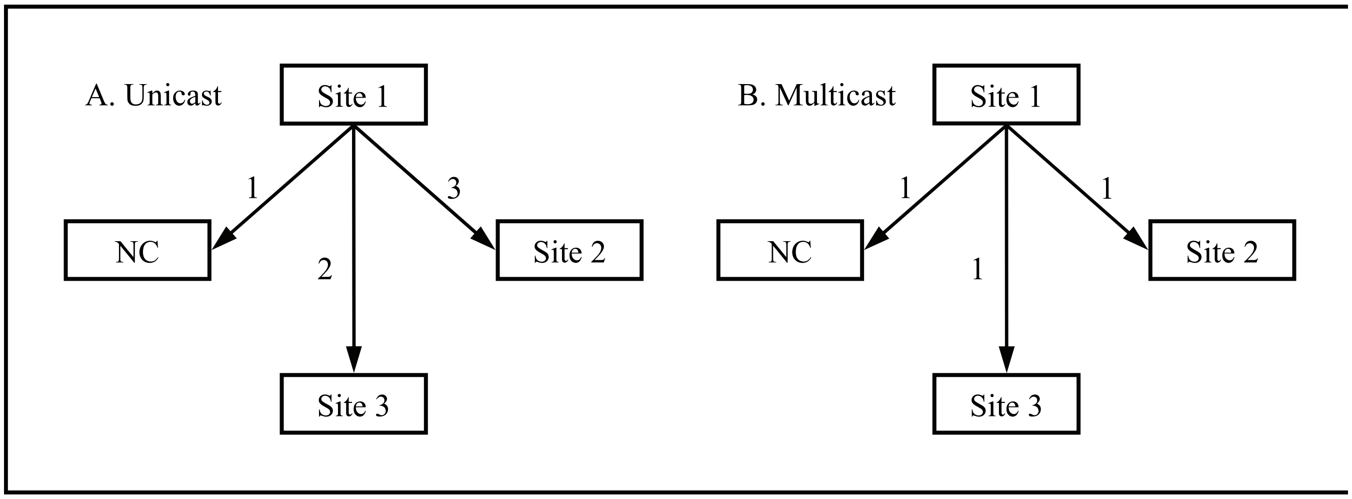


Figure 2: An assumed four site directed network.

If packet radio stations are available, then let us assume a directed network with four sites; "Site 1", "Site 2", "Site 3", and "NC". Let us also assume that a missing or lost child is reported at "Site 1".

Using traditional packet software (e.g. Winlink 2000 and AirMail), "Site 1" would transmit the digital image of the child to "NC", "Site 2", and "Site 3" in three separate transmissions. This type of transmission, shown in Figure 2A, is referred to as unicast (Wiedemeier, 2008) and each of the three unicast transmissions are labeled with a number that specifies the order of transmission. Here, the order of transmission is not important. What is important, however, is that three individual unicast transmissions are required before all sites receive a copy of the digital image.

An optimal data transmission method would be for "Site 1" to transmit the digital image of the child to "NC", "Site 2", and "Site 3" simultaneously. This type of transmission is referred to as multicast (Wiedemeier, 2008) and is shown in Figure 2B. Given that both unicast and multicast can be used to transmit the digital image of the child, the purpose of this paper is to compare the transmission of digital data using Pure-FTP, a unicast file transfer tool, and Udpcast, a multicast file transfer tool. Our data show that Udpcast performs better than Pure-FTP. Thus, we argue that multicast file transfer tools should be used by the amateur radio community when transmitting digital data, imagery or otherwise.

In the following "2-Meter Packet Radio Stations" section, we discuss how two 2-meter packet radio stations were configured and used to obtain unicast and multicast file transmission times. In the "Udpcast File Transfer Tool" section, we discuss how to multicast data using Udpcast. We also list the specific Udpcast command line arguments we used to obtain instantaneous bandwidths and how Microsoft Excel 2007 was used to compute multicast transmission times. In the "Pure-FTP File Transfer Tool" section, we discuss how we used Pure-FTP to obtain unicast transmission times.

In the "Digital Files" section, we discuss the structure of the 4 KB, 8 KB, and 16 KB files utilized for our unicast and multicast research. The "Results" section compares the unicast transmission times obtained by Pure-FTP against the multicast transmission times computed using average instantaneous bandwidth generated by Udpcast. The "Conclusions" section re-caps what we discussed in the "Result" section and lists future research we intend to conduct.

2-Meter Packet Radio Stations

To conduct our research, we utilized two identically configured 2-meter packet radio stations located on the campus of The University of Louisiana at Monroe. Each packet radio station consisted of a Dell OptiPlex GX240 personal computer, a Kenwood TM-271A 2-meter transceiver, a Kantronics KPC-3 Plus terminal node controller, and a Diamond X30A antenna. Power was supplied by an Astron SS-30 power supply and distributed by a West Mountain Radio RigRunner 4005. See Figure 3.

We installed Fedora Linux Core 8 on both Dell OptiPlex GX240 personal computers and configured their AX.25 network interfaces (Tranter, 2001) (Jones, 1996) to transmit and receive data using the Amateur Packet Radio Network (i.e. AMPRNet) "testing" IP addresses 44.128.*.*. Specifically, we used the IP addresses 44.128.2.1 and 44.128.2.2. For a thorough discussion of how the personal computers used in this research were configured, please refer to (Wiedemeier, 2008). Specifically, Figures A1-A7 in the appendix of (Wiedemeier, 2008) show the Bourne shell scripts and Unix commands we used to configure the personal computers.

Additionally, we configured three commands within the Kantronics KPC-3 Plus terminal node controllers. The `paclen` command was set to 0 (e.g. 256 Bytes). The `frack` command was set to 7 and the `maxframe` command was set to 7. We refer the reader to the Kantronics KPC-3 Plus User

Guide (Kantronics, 2005), which provides a thorough discussion of all terminal node controller commands.

The Kenwood TM-271A transceivers were configured to transmit and receive data at the transmission rate of 1200 bits per second (bps) using the frequency 145.010 Megahertz (MHz) (Kenwood, 2009). Because both transceivers were located within ten to fifteen feet of each other in the same room, their transmit power levels were set to low.

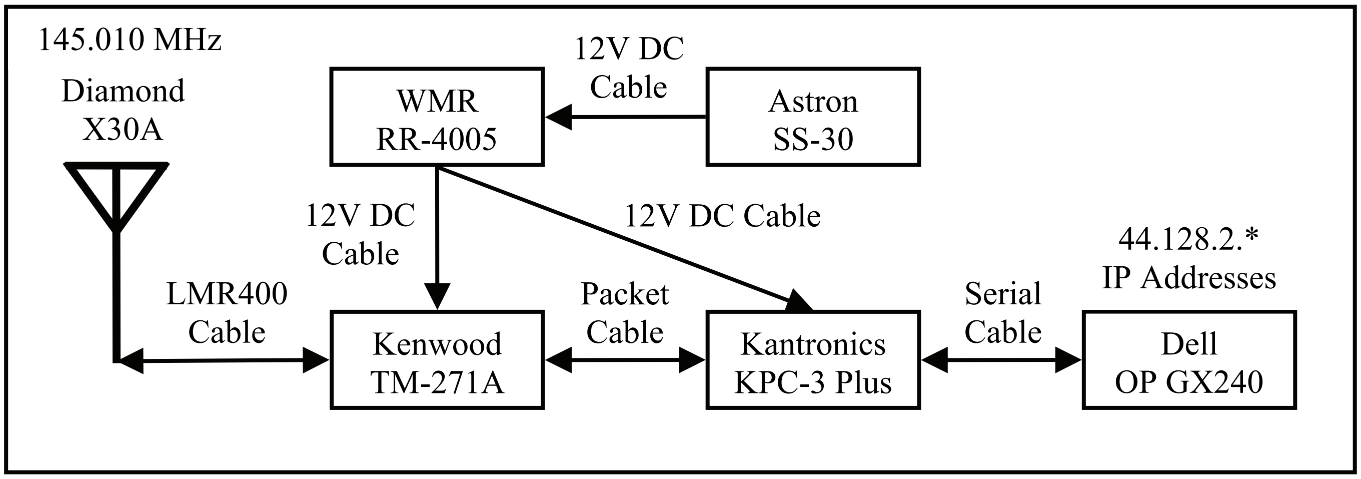


Figure 3: Configuration of a 2-meter packet radio station.

Udpcast File Transfer Tool

The Udpcast software is freely available and easily installed on personal computers running the Fedora Linux operating system using the administration software management tool. The Udpcast software can, however, be downloaded from the website `http://udpcast.linux.lu/` and installed on other Unix/Linux operating systems. A version of Udpcast also exists for the Microsoft Windows operating system. A thorough discussion of the Udpcast file transfer tool is provided by (Wiedemeier, 2008). For our research, we used the Udpcast sender and Udpcast receiver Unix commands shown in Figure 4.

```
udp-sender   --file FILETOSEND –-interface INTERFACE –b BLOCKSIZE
             --async –-fec IxR/S --max-bitrate BPS --log LOGFILE
             --bw-period SECONDS

udp-receiver --file FILETORECEIVE --nosync --interface INTERFACE
```

Figure 4: The Udpcast sender and Udpcast receiver Unix commands.

Because the Udpcast file transfer tool multicasts data asynchronously, the Udpcast sender adds forward error correction to the data stream. This allows the Udpcast receiver to correct any data that may be corrupted during transmission. The amount of forward error correction the Udpcast sender adds to each file transmission is based on five factors; (1) file size, (2) block size, (3) interleave (i.e. I – the number of stripes), (4) redundancy (i.e. R – the number of additional stripes to transmit), and (5) stripe size (i.e. S – measured in blocks). For specific information about the Udp sender `--fec` (forward error correction) command line argument the reader should refer to (Knaff, 2005 and 2006). In the following paragraphs, we will briefly describe how forward error correction is computed.

The smallest unit of data that the Udpcast sender transmits during file transmission is a slice and each file transmitted is divided into one or more slices. The size of a slice (i.e. slice size in blocks) is computed by multiplying the interleave by the stripe size (i.e. slice size = I x S blocks). Udpcast requires that the slice size be greater than or equal to 16 blocks but less than or equal to 1024 blocks (i.e. 16 blocks <= slice size = I x S <= 1024 blocks). Additionally, the interleave must be less than or equal to 8 (i.e. I <= 8) and the stripe size must be less than or equal to 128 (i.e. S <= 128).

Given that the smallest slice size the Udpcast sender will transmit is 16 blocks and the block size we use for our research is 256 Bytes (i.e. from the terminal node controller `paclen` command), then the smallest file the Udpcast sender could transmit is 4 Kilobytes (KB). That is, the Udpcast sender transmits a single slice of size 16 blocks. In this instance, the product of the interleave and stripe size (i.e. I x S) must equal 16.

Table 1: Udpcast command line argument associated values.

File Size	Block Size	Band Width	FEC (IxR/S)	Number of Blocks	Slices Transmitted	Percent File Redundancy	FEC in Bytes	Bytes Transmitted
4 KB	256 Bytes	1200 bps	1x1/16	16	1	6.25%	256	4,353
			1x2/16			12.5%	512	4,608
			1x4/16			25%	1,024	5,120
			1x8/16			50%	2,048	6,144
			1x16/16			100%	4,096	8,192
8 KB		1200 bps	1x1/16	32	2	6.25%	512	8,704
			1x2/16			12.5%	1,024	9,216
			1x4/16			25%	2,048	10,240
			1x8/16			50%	4,096	12,288
			1x16/16			100%	8,192	16,384
16 KB		900 bps	1x1/16	64	4	6.25%	1,024	17,408
			1x2/16			12.5%	2,048	18,432
			1x4/16			25%	4,096	20,480
			1x8/16			50%	8,192	24,576
			1x16/16			100%	16,384	32,768

For this research, we chose to transmit files of size 4 KB, 8 KB, and 16 KB. A 4 KB file was selected because it represents the minimum size that can be transmitted by the Udpcast sender given a 256 Byte block size. The 16 KB file was selected because it is four times the size of the 4 KB file and we desired to minimize the overall transmission time to around 10 minutes. We selected the 8 KB file because it is twice the size of the 4 KB file, but half the size of the 16 KB file.

The first four columns in Table 1 show the file size, block size, bandwidths, and forward error correction (FEC) we used for the research presented in this paper. The values shown were used as arguments to the `--file`, `-b`, `--max-bitrate`, and `--fec` Udp sender command line arguments shown in Figure 4. What is not shown in Figure 4 are the values for the three Udp sender command line arguments `--interface`, `--log`, and `--bw-period`, which were `ax0`, `logfile.txt`, and `5` (e.g. 5 seconds) respectively.

As stated earlier, the file sizes we used were 4 KB, 8 KB, and 16 KB. The block size we used was 256 Bytes because this represented the maximum `paclen` value that could be set for the terminal node controllers. The maximum bandwidth we used to transmit the 4 KB and 8 KB files was 1200 bps because, as was stated earlier, the transceivers were set to transmit data at 1200 bps. We found, however, through trial and error, that the maximum bandwidth that would support a 16 KB file multicast was 900 bps. Due to the asynchronous nature of Udpcast, we hypothesize that multicast transmissions are more vulnerable when large files are transmitted using a low transmission rate of 1200 bps, regardless of the amount of forward error correction added to the transmission. More research into this anomaly, however, is needed.

The interleave, redundancy, and stripe size values that we used to multicast data were 1x1/16, 1x2/16, 1x4/16, 1x8/16, and 1x16/16. We could have used the interleave, redundancy, and stripe size values 2x1/8 instead of 1x2/16, as both represent a 12.5% file redundancy. However, we found that computed multicast transmission times using interleave, redundancy, and stripe size values 2x1/8 and 1x2/16 for all file sizes were similar to the whole part of a decimal number. Additionally, by using a value of 16 for stripe size (i.e. S) we were able to obtain a 6.25% file redundancy (e.g. with interleave, redundancy, and stripe size values 1x1/16).

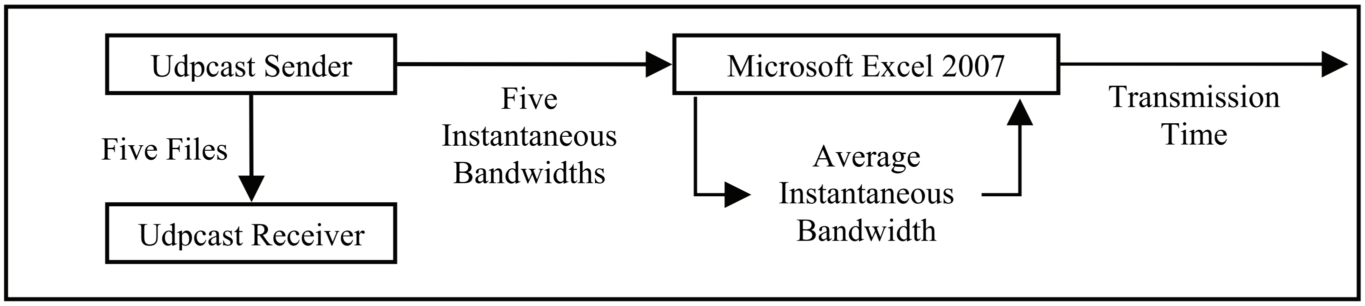


Figure 5: Our method for obtaining data multicast transmission times.

In Table 1, the last five columns, starting with column “Number of Blocks”, display values that are computed using values from the first four columns. The “Number of Block” column displays the number of blocks associated with a specific file size and is the quotient of a file size numerator and a block size denominator. The “Slices Transmitted” column displays the number of slices transmitted for a particular file size and is the quotient of a number of blocks numerator and a denominator of I x S (i.e. the product of interleave and stripe size). The “Percent File Redundancy” column displays the amount of forward error correction added to the transmission as a percentage of the file size and is a quotient of a redundancy (i.e. R) numerator and a stripe size (i.e. S) denominator. The “FEC in Bytes” displays the number of bytes of forward error correction that will be transmitted and is the product of file size and percent file redundancy. The "Bytes Transmitted" column displays the total number of bytes transmitted for a give file size and is the sum of file size and forward error correction.

Overall, using Udpcast, we transmitted the 4 KB, 8 KB, and 16 KB files five times for each of the interleave, redundancy, and stripe size combinations shown in Table 1. Because Udpcast is an asynchronous data multicast tool (Wiedemeier, 2008), neither the Udpcast sender nor the Udpcast receiver generate a file transmission time. However, the Udpcast sender will record an instantaneous bandwidth value in a log file if the `--log LOGFILE` command line argument is specified. See Figure 4.

$$\boxed{TransmissionTime = \left[\left(\frac{BlockSize * 8}{Bandwidth}\right) + Delay\right] * \left(\frac{FileSize + FEC}{BlockSize}\right)}$$

Figure 6: Mathematical equation used to compute Udpcast file transmission times.

Figure 5 shows the method we used to multicast five files, generate five instantaneous bandwidths, compute an average instantaneous bandwidth, and compute a transmission time. Specifically, the Udpcast sender transmits, using the interleave, redundancy, and stripe size values shown column 4 in Table 1, a 4 KB, 8 KB, or 16 KB file to the Udpcast receiver five times. We then used the `wc` (e.g. file word count) and `diff` (e.g. file difference) Unix command on the Udpcast receiver personal computer to verify that the file received was correct. The five instantaneous bandwidths were then recorded in a Microsoft Excel 2007 spreadsheet. Using the five instantaneous bandwidths, an average instantaneous bandwidth was computed. The associated transmission time was computed using the equation shown in Figure 6. The Bandwidth variable from the equation shown in Figure 6 represents the average instantaneous bandwidth computed by the Microsoft Excel 2007 spreadsheet shown in Figure 5.

Pure-FTP File Transfer Tool

Similar to Udpcast, Pure-FTP is freely available software and easily installed on personal computers running the Fedora Linux operating system using the administrative software management tool. The Pure-FTP software can be downloaded from the website `http://www.pureftpd.org/` and installed on other Unix/Linux operating systems. Additionally, a version of Pure-FTP exists for the Microsoft Windows operating system. Note that FTP is a standard computing acronym for File Transfer Protocol.

Pure-FTP, as are all FTP file transfer tools, is comprised of two parts: a client and a server. To connect to a remote FTP server and upload or download files, one would use the Pure-FTP client. Using the FTP client, users who have accounts on the FTP server can then upload files to or download files from the server. Likewise, an anonymous FTP server can be configured to allow anonymous (e.g. guest) users to upload files to or download files from the anonymous FTP server.

After installing and configuring Pure-FTP servers on the personal computers associated with our packet radio stations, we used the Pure-FTP client on one personal computer to connect anonymously to the Pure-FTP server on the other personal computer using the AX.25 network interface discussed earlier.

We then retrieved each of the 4 KB, 8 KB, and 16 KB files ten times. During file retrieval, the Pure-FTP server displays the clock time at which each file retrieval occurred and the total time in seconds required to retrieve the file. For each file retrieved by the FTP client, the file was verified to be correct by comparing it against a “correct” file within the FTP client’s file system using the `wc` and `diff` Unix commands.

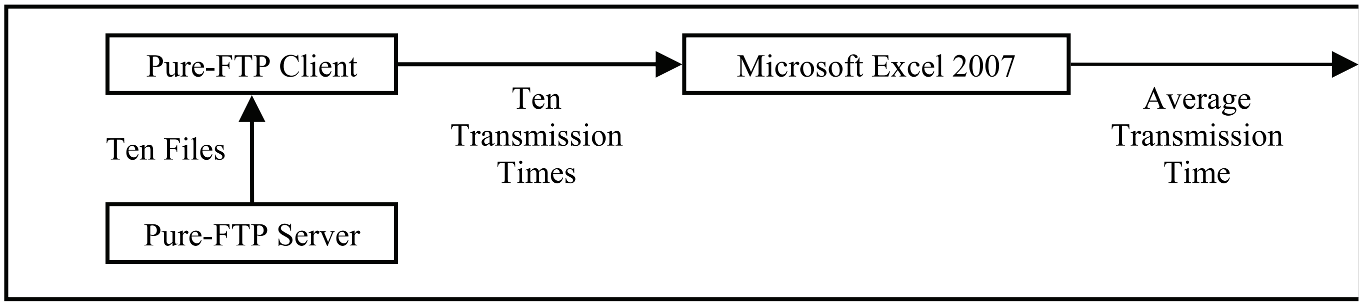


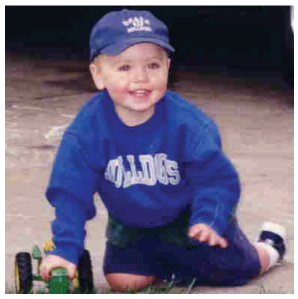
Figure 7: Our method for obtaining data unicast average transmission times.

The ten transmission times generated by Pure-FTP when transmitting the 4 KB file were recorded in a Microsoft Excel 2007 spreadsheet. An average transmission time was then calculated. The ten 8 KB and 16 KB files transmission times generated by Pure-FTP were also recorded in the same Microsoft Excel 2007 spreadsheet and then averaged. See Figure 7.

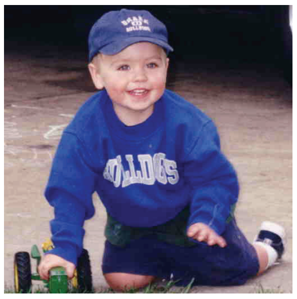
Digital Files

As discussed earlier in this paper, we chose to conduct our unicast and multicast research using files of size 4 KB, 8 KB, and 16 KB. If we use the digital image of the child shown in Figure 1 and adjust the number of pixels in the image and the overall image quality, then we generate the three images shown in Figure 8. Unfortunately, none of the three images shown in Figure 8 is exact with respect to the stated size. All are slightly smaller than the size indicated. They are, however, representative of what one would expect a digital image to represent using a specific file size and pixel size.

A. 4 KB File Size
256x256 Pixels



B. 8 KB File Size
384x384 Pixels



C. 16 KB File Size
512x512 Pixels

Figure 8: Digital images of a missing or lost child. Each is smaller than the listed size.

To obtain files with the "exact" sizes of 4 KB, 8 KB, and 16 KB for our research, we created three ASCII text files, each containing 4,096, 8,192, and 16,384 ASCII characters respectively. Because of their exact sizes, we use the three ASCII files instead of digital images when conducting our research.

Results

The times generated by Udpcast and Pure-FTP, when each transmitted the 4 KB, 8 KB, and 16 KB files, are shown in Tables 2 and 3 respectively and plotted in Figure 9. From the data, we see that Udpcast transmits the 4 KB, 8 KB, and 16 KB files, using 6.25% - 50% file redundancies, in less time than Pure-FTP for 1, 2, and 3 file transmissions. Specifically, we see from Tables 4, 5, and 6, and Figures 10, 11, and 12 that Udpcast, using 50% file redundancies, transmits the 4 KB and 8 KB files 45% faster than Pure-FTP for 1 file transmission. Udpcast, using 50% file redundancy, transmits the 16 KB file 25% faster than Pure-FTP for 1 file transmission. When the percent file redundancies decrease (i.e. 25%, 12.5%, and 6.25%), Udpcast performs even better than Pure-FTP for 1 file transmission. To compute the percentage amount that Udpcast is faster than Pure-FTP we subtracted the Udpcast transmission time from the Pure-FTP transmission time and then divided the difference by the Pure-FTP transmission time.

Table 2: Udpcast average instantaneous bandwidths and associated modeled transmission times.

File Size	FEC (IxR/S)	Percent File Redundancy	Bytes Transmitted	Average Instantaneous Bandwidth	Transmission Time in Seconds
4 KB	1x1/16	6.25%	4,353	1021	34.087
	1x2/16	12.5%	4,608	962	38.320
	1x4/16	25%	5,120	861	47.573
	1x8/16	50%	6,144	712	69.034
	1x16/16	100%	8,192	528	124.121
8 KB	1x1/16	6.25%	8,704	993	70.137
	1x2/16	12.5%	9,216	937	78.710
	1x4/16	25%	10,240	840	97.559
	1x8/16	50%	12,288	697	141.019
	1x16/16	100%	16,384	520	252.256
16 KB	1x1/16	6.25%	17,408	732	190.186
	1x2/16	12.5%	18,432	692	213.241
	1x4/16	25%	20,480	622	263.536
	1x8/16	50%	24,576	516	380.839
	1x16/16	100%	32,768	386	679.042

From the data, we see also that Udpcast transmits the 4 KB and 8 KB files in less time, using 100% file redundancy, than Pure-FTP for 1, 2 and 3 file transmission. Again, we see from Tables 4, 5, and 6, and Figures 10, 11, and 12 that Udpcast, using 100% file redundancies, transmits the 4 KB file 1.73%, 50.86%, and 69.24% faster than Pure-FTP for 1, 2, and 3 file transmissions respectively. Udpcast, using 100% file redundancies, transmits the 8 KB file 2.11%, 51.06%, and 67.37% faster than Pure-FTP for 1, 2, and 3 file transmissions respectively.

Udpcast also transmits a 16 KB file, using 100% file redundancy, in less time than Pure-FTP for 2 and 3 file transmission. We see from Tables 4, 5, and 6, and Figures 10, 11, and R4 that Udpcast, using 100% file redundancies, transmits the 16 KB file 33.58%, and 55.72% faster than Pure-FTP for 2, and 3 file transmissions respectively.

However, Pure-FTP transmits a 16 KB file in a single transmission in less time than Udpcast, using 100% file redundancy. Specifically, from Tables 4, 5, and 6, and Figures 10, 11, and 12 we see that Pure-FTP transmits a 16 KB file in a single transmission 24.72% faster than Udpcast. To compute the percentage amount that Pure-FTP is faster than Udpcast we subtracted the Pure-FTP transmission time from the Udpcast transmission time and then divided difference by the Udpcast transmission time.

We hypothesize the reason why Pure-FTP transmits a 16 KB file in a single transmission in less time than Udpcast, using 100% file redundancy, is based on the actual number of bytes each file transfer tool transmits. From Table 3, we see that Pure-FTP transmits 18,944 Bytes for each 16 KB file transmitted once. This Byte count represents sixty-four 256 Byte data packets (i.e. blocks) and their associated 40 Byte acknowledgement packets. Udpcast, using 100% file redundancy, transmits 32,768 Bytes for the same 16 KB file (i.e. 16 KB file and 16 KB forward error correction). See Table 2. Essentially, Udpcast transmits the 16 KB file twice and overall transmits 13,824 more Bytes than does Pure-FTP.

Table 3: Pure-FTP average transmission times.

File Size	Number of File Transmissions	Bytes Transmitted	Average Transmission Time in Seconds
4 KB	1	4,736	126.300
	2	9,472	256.600
	3	14,208	378.900
8 KB	1	9,472	257.700
	2	18,944	515.400
	3	28,416	773.110
16 KB	1	18,944	511.200
	2	37,888	1,022.400
	3	56,832	1,533.600

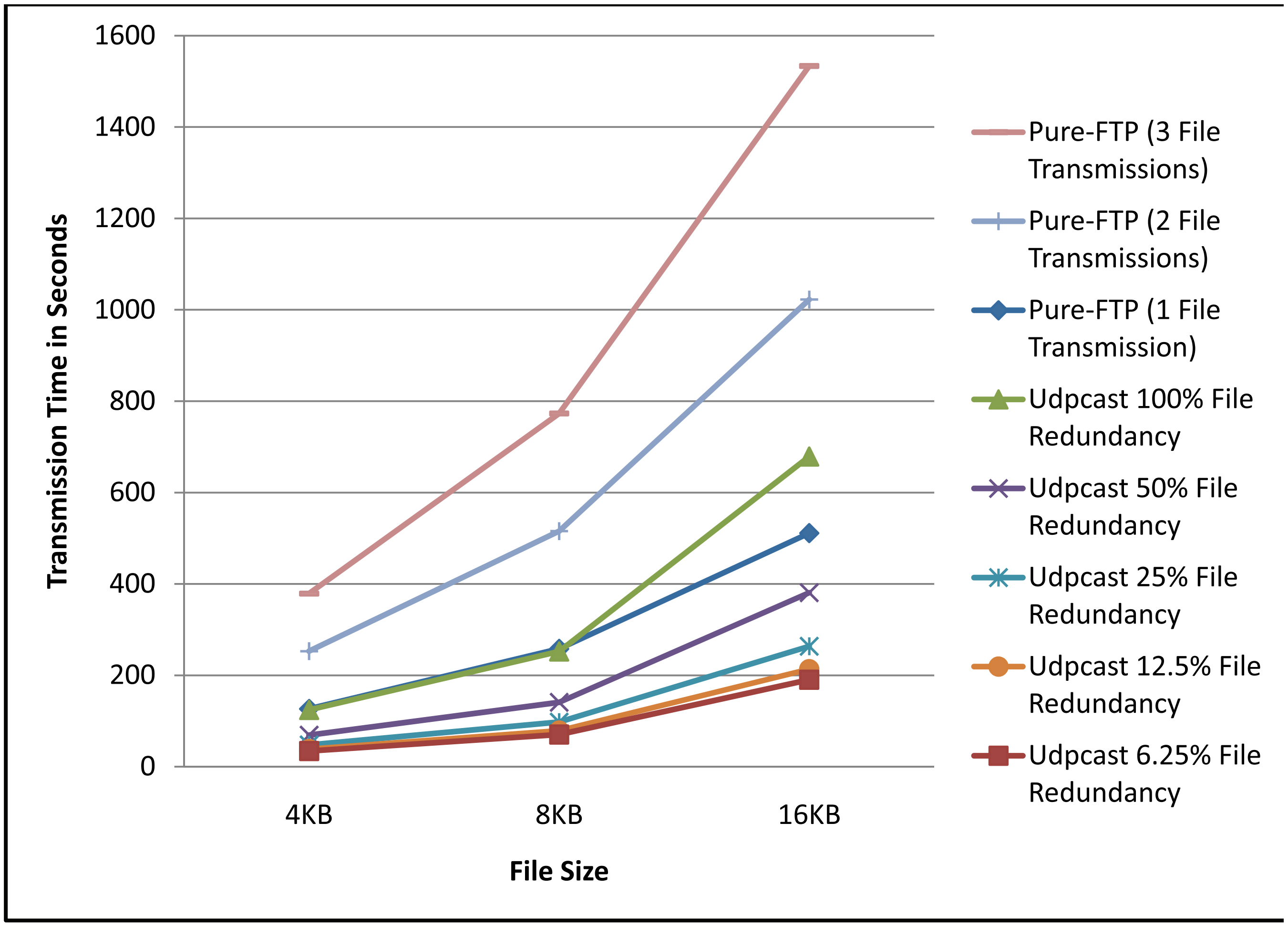


Figure 9: Udpcast and Pure-FTP file transmission times.

Table 4: Udpcast percent faster than Pure-FTP for the 4 KB file.

File Size	Udpcast Percent File Redundancy	Pure-FTP (1 File Transmission)	Pure-FTP (2 File Transmissions)	Pure-FTP (3 File Transmissions)
4 KB	6.25%	73.01%	86.51%	90.00%
	12.5%	69.66%	84.83%	89.89%
	25%	62.33%	81.17%	87.44%
	50%	45.34%	72.67%	81.78%
	100%	1.73%	50.86%	67.24%

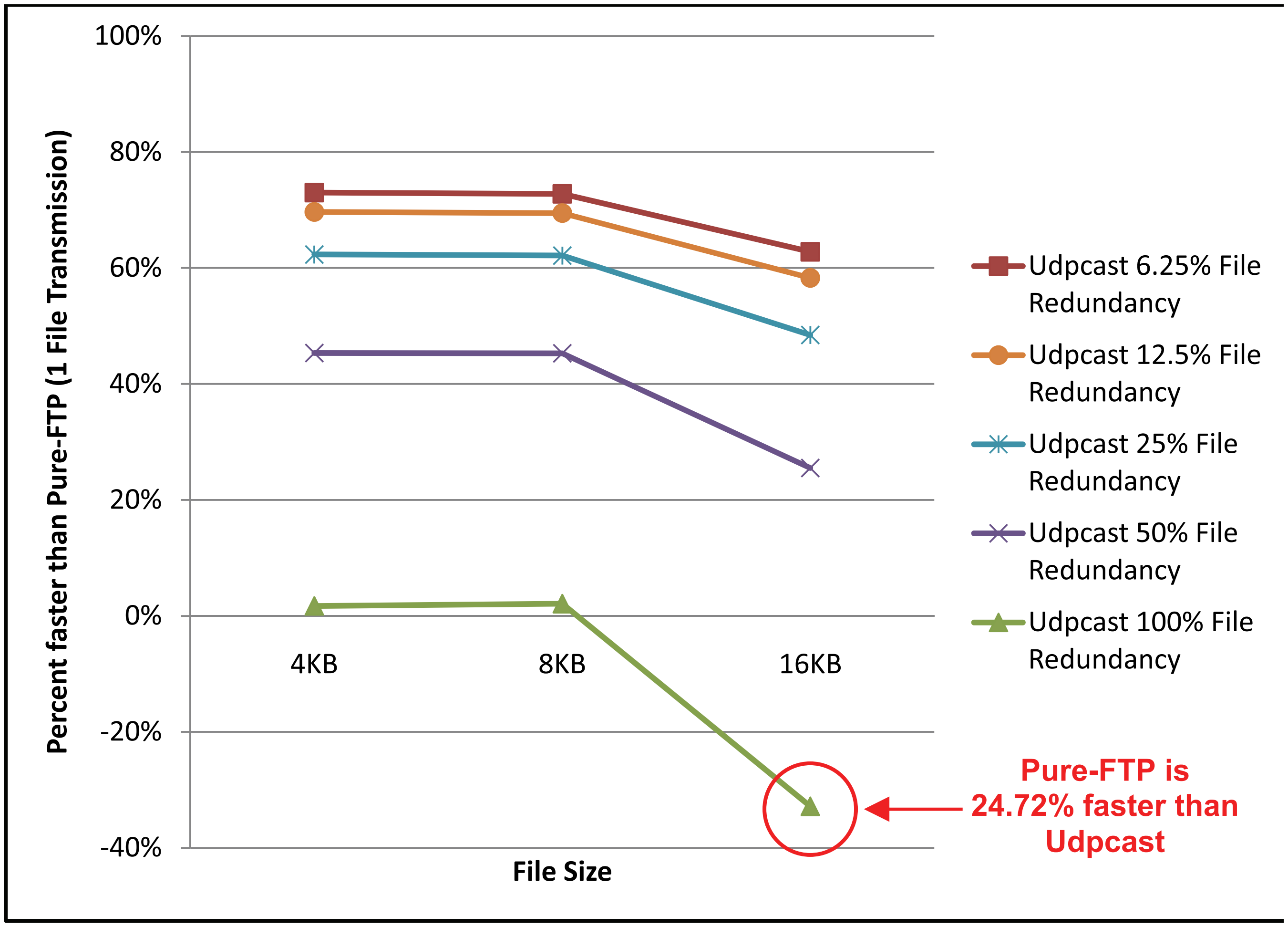


Figure 10: Udpcast percent faster than Pure-FTP (1 File Transmission).

Table 5: Udpcast percent faster than Pure-FTP for the 8 KB file.

File Size	Udpcast Percent File Redundancy	Pure-FTP (1 File Transmission)	Pure-FTP (2 File Transmissions)	Pure-FTP (3 File Transmissions)
8 KB	6.25%	72.78%	86.39%	90.93%
	12.5%	69.46%	84.73%	89.82%
	25%	62.14%	81.07%	87.38%
	50%	45.28%	72.64%	81.76%
	100%	2.11%	51.06%	67.37%

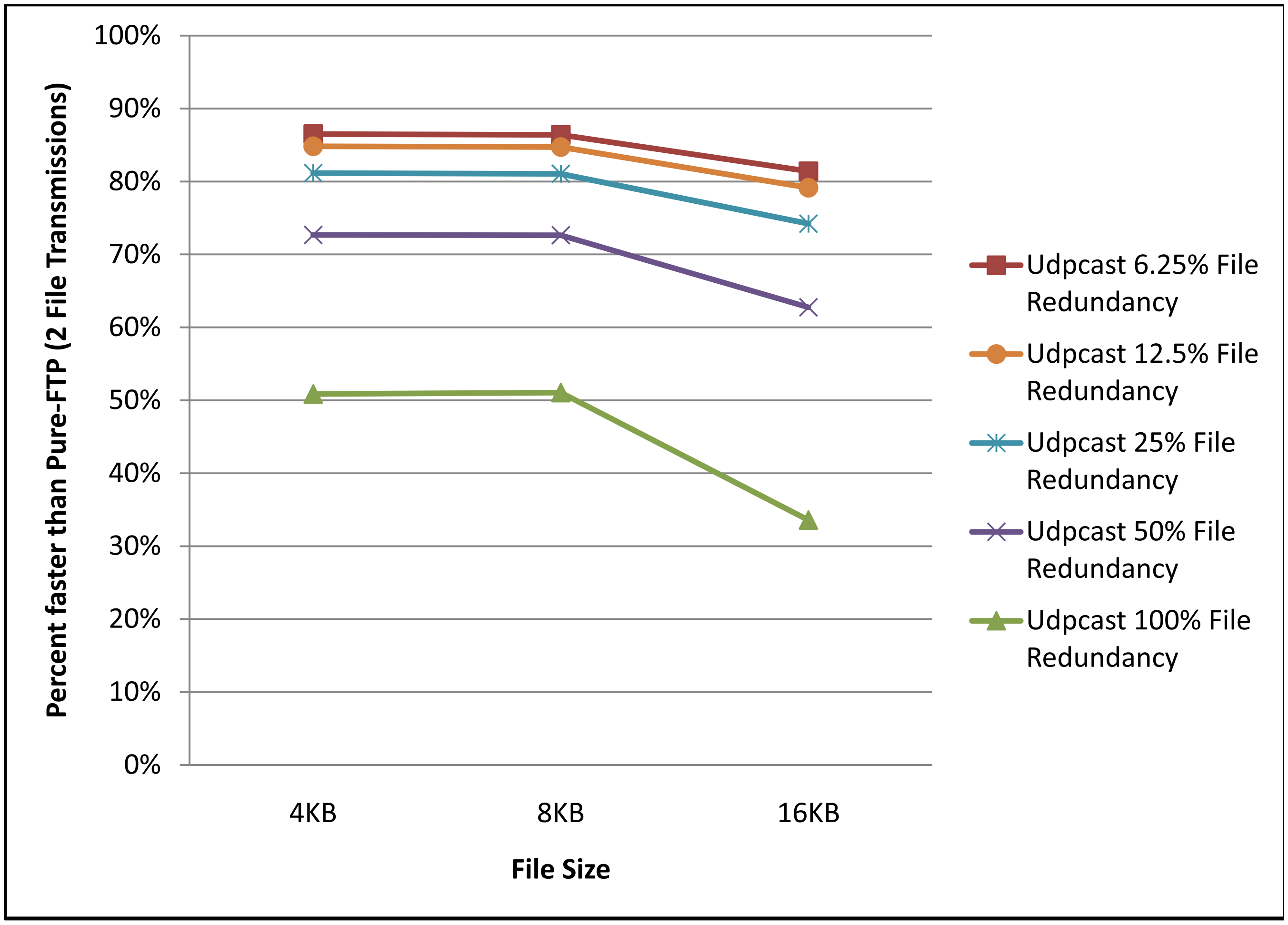


Figure 11: Udpcast percent faster than Pure-FTP (2 File Transmissions).

Table 6: Udpcast percent faster than Pure-FTP for the 16 KB file.

File Size	Udpcast Percent File Redundancy	Pure-FTP (1 File Transmission)	Pure-FTP (2 File Transmissions)	Pure-FTP (3 File Transmissions)
16 KB	6.25%	62.80%	81.40%	87.60%
	12.5%	58.29%	79.14%	86.10%
	25%	48.45%	74.22%	82.82%
	50%	25.50%	62.75%	75.17%
	100%	-32.83%	33.58%	55.72%

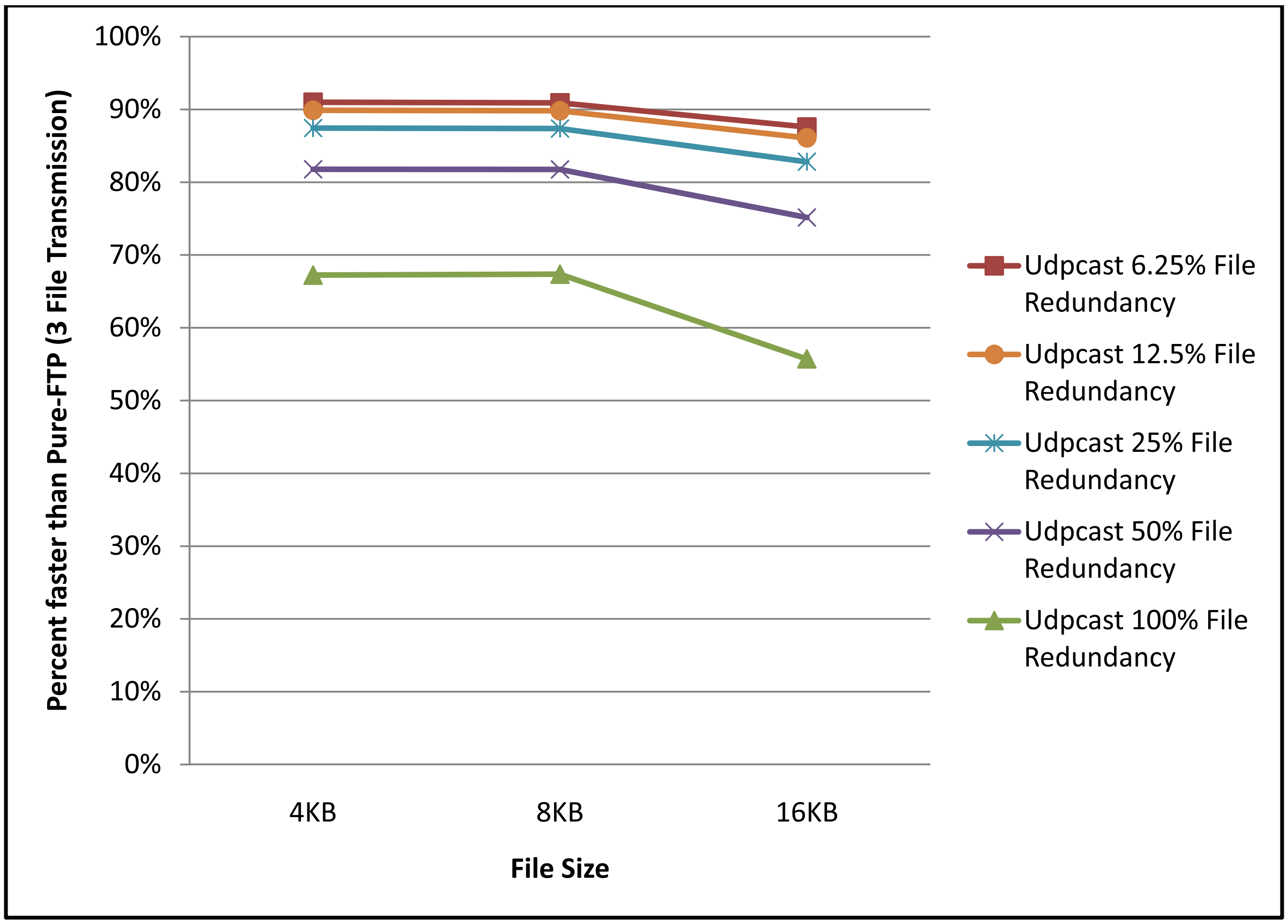


Figure 12: Udpcast percent faster than Pure-FTP (3 File Transmissions).

Conclusions

From the data shown in Tables 2 and 3 and plotted in Figure 9, it is clear that Udpcast, when using 100% file redundancy or less, transmits the 4 KB, 8 KB, and 16 KB files in less time than Pure-FTP for 2 or greater file transmission. We argue that that if one must transmit a file to two or more locales, it is best to use Udpcast. Even if you must transmit a file to one locale, Udpcast transmits the 4 KB, 8 KB, and 16 KB files in less time than Pure-FTP for file redundancies 6.25% - 50%. That is, if the communication channel is relatively error free, then Udpcast performs better than Pure-FTP in all instances.

We should note that the transmission times obtained using Pure-FTP, and Udpcast for that matter, occurred when no other local packet traffic was present. That is, the overall bit error rate associated with the communication channel would be considered zero during transmission. If packet traffic, other than ours (e.g. competing packet traffic), were present when we attempted to unicast data using Pure-FTP, then the resulting transmission times would have been larger than those shown in Table 3. This is because Pure-FTP utilized the “connection-oriented” Transmission Control Protocol (TCP), which uses acknowledgement packet to signify that a packet arrives correctly and retransmission when a packet arrives corrupt. As such, we would expect that Pure-FTP transmission times would increase when channel competition occurs because the number of packets retransmitted would also increase.

Thus, if channel competition were to occur, Pure-FTP would generate transmission times worse than those shown in Table 3. How much more the transmission times generated by Pure-FTP would increase depends on the bit error rate when the transmission occurred.

In contrast to Pure-FTP, Udpcast utilizes the "connection-less" User Datagram Protocol (UDP) and incorporates forward error correction to overcome any instances where channel competition generates corrupt packets. Thus, if channel packet competition exists, Udpcast transmission times would remain the same because the amount of forward error correction added by Udpcast is independent of the communication channel's bit error rate. As such, the transmission times shown in Table 2 would not change, even if the communication channel's bit error rate was non-zero.

We have identified three future research projects associated with this research. First, we intend to investigate the anomaly discussed in "Udpcast File Transfer Tool" section of this paper. Specifically, we will investigate how much Udpcast sender instantaneous bandwidth decreases as file size increases. Second, we intend to investigate the transmission of files using unicast and multicast, but substitute 70-centimeter transceivers in place of the 2-meter transceivers. Given that the 70-centimeter transceivers will allow us to transmit data at a maximum rate of 9600 bps, we also intend to transmit files larger than 16 KB.

Last, using the data discussed in this paper and the data generated from the project discussed in the previous paragraph, we will attempt to define a mathematical equation that can be used estimate Udpcast instantaneous bandwidths for faster transmission rates (e.g. fast packet transmission rates 19,200 bps, 38,400 bps, 76,800 bps, and up). By using this new mathematical equation and the one shown in Figure 6, we will be able to model data multicast over proposed fast packet radio networks.

Acknowledgements

The author graciously thanks Allison M.D. Wiedemeier, Ph.D., for reading the first draft of this paper. The author's research is supported through funds provided by the Clarke M. Williams, Jr. Endowed Professorship in Computer Science sponsored by The University of Louisiana at Monroe Foundation, and The University of Louisiana at Monroe Digital Communication Research Laboratory.

References

Jones, Greg (Ed.) (WD5IVD). (1996). *Packet Radio: What? Why? How? Articles and Information on General Packet Radio Topics*. Tucson, AZ: Tucson Amateur Packet Radio Corporation Publisher.

Kantronics, Incorporated. (2005, September 9). *KPC-3 Plus User Guide*. Retrieved July 28, 2009, from http://www.kantronics.com/documents/kpc-3plus_manual_RevD.pdf.

Kenwood USA, Incorporated. (2009). *Kenwood TM-271A Instruction Manual*. Retrieved July 28, 2009, from http://inform3.kenwoodusa.com/Manuals%5CTM-271.pdf.

Knaff, Alain. (2005, May 14). *Udpcast Commandline Options*. Retrieved July 28, 2009, from http://udpcast.linux.lu/cmd.html.

Knaff, Alain. (2006, March 20). "Multicasting over satellite". *Udpcast Forums*. Retrieved July 28, 2009, from http://udpcast.linux.lu/pipermail/udpcast/2006-March/000493.html.

Tranter, Jeff (VE3ICH). (2001, September 19). *Linux Amateur Radio AX.25 HOWTO*. Retrieved July 21, 2009, from http://tldp.org/HOWTO/AX25-HOWTO/.

Wiedemeier, Paul (KE5LKY). (2008, September). “Using Udpcast to IP Multicast Data over Packet Radio Networks”. (pp. 94-105). *Proceedings of the 27th ARRL and TAPR Digital Communications Conference*.

Biography

Dr. Paul D. Wiedemeier is an Assistant Professor of Computer Science at The University of Louisiana at Monroe (ULM) and the principle investigator of the ULM Digital Communication Research Laboratory. Dr. Wiedemeier obtained a Ph.D. in Computer Engineering and Computer Science from the University of Missouri – Columbia, a M.S. in Computer Science from Michigan Technological University, and a B.S. in Computer Science from Drake University. He is a member of the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, the Amateur Radio Relay League, the Tucson Amateur Packet Radio Corporation, the Consortium for Computing Sciences in Colleges, and the Louisiana Academy of Sciences. Dr. Wiedemeier also holds a Technicians Amateur Radio License (KE5LKY) issued by the United States of America Federal Communications Commission.