PROJECT CRAFT

A Real-Time Delivery System for Nexrad Level II Data Via The Internet

BY KEVIN E. KELLEHER, KELVIN K. DROEGEMEIER, JASON J. LEVIT, CARL SINCLAIR, DAVID E. JAHN, SCOTT D. HILL, LORA MUELLER, GRANT QUALLEY, TIM D. CRUM, STEVEN D. SMITH, STEPHEN A. DEL GRECO, S. LAKSHMIVARAHAN, LINDA MILLER, MOHAN RAMAMURTHY, BEN DOMENICO, AND DAVID W. FULKER

This document is a supplement to "Project Craft: A Real-time Delivery System for Nexrad Level II Data Via The Internet," by Kevin E. Kelleher, Kelvin K. Droegemeier, Jason J. Levit, Carl Sinclair, David E. Jahn, Scott D. Hill, Lora Mueller, Grant Qualley, Tim D. Crum, Steven D. Smith, Stephen A. Del Greco, S. Lakshmivarahan, Linda Miller, Mohan Ramamurthy, Ben Domenico, and David W. Fulker (*Bull. Amer. Meteor. Soc.,* **88,** 1045–1057) • ©2007 American Meteorological Society
• *Corresponding author:* Kevin E. Kelleher, National Severe Storms Laboratory, 120 David L. Boren Blvd., Norman, OK 73072 • E-mail: Kevin.Kelleher@noaa.gov • DOI: 10.1175/BAMS-88-7-Kelleher.

COMPRESSION OF CRAFT DATA. In general, compression schemes work well when the input data exhibit properties for which the techniques are designed to compress. Specific knowledge of the structure of the radar data was needed to objectively determine which technique(s) to use. To understand the structure of the radar data, a comprehensive characterization of frequency distribution was compiled (Smith et al. 2002; Kelleher et al. 2003) for both individual fields (reflectivity, Doppler velocity, spectrum width) and combined fields (level II data transmitted with the three moments interlaced).

Several different loss-less compression techniques were investigated, including dictionary- and probability-based schemes (Lelewer and Hirschberg 1987). Huffman coding and arithmetic coding (Witten et al. 1987), both probability distribution–based coding schemes, were considered. The primary performance measure for all the compression algorithms examined in this study was compression ratio (CR), defined as the input data size divided by the output data size. The larger the CR, the more compression achieved. Other measures of algorithm performance address the time and space complexity of compression and decompression. These are equally important as CR in measuring algorithm performance since an algorithm that achieves superior CR may take a prohibitively long time (CPU cycles) to compress (decompress) the data or may exceed the memory or disk requirements of the target computing machine. Algorithms that exhibit relatively large space and time complexity were not consistent with the need for real-time transmission of these data and were not considered. A 0-order Huffman, an arithmetic, and a UNIX compression algorithm were selected for testing based upon the level II data frequency distribution information and space and time complexity requirements.

In the original Collaborative Radar Acquisition Field Test (CRAFT) prototype, the freeware loss-less compression algorithm *bzip2* (www.bzip.org) was implemented by Harry Edmon of the University of Washington and it worked well. It was important, from a software cost and support view, to choose an existing algorithm that was freely available in the

public domain (i.e., off the shelf with a nonrestrictive copyright agreement) and one that could be easy to integrate into the Next-Generation Weather Radar (NEXRAD) software build cycle. Because *bzip2* is a hybrid algorithm that not only incorporated many of the attributes found in the Huffman, arithmetic, and UNIX compress algorithms, but outperformed them, an effort was made to find a better algorithm than *bzip2* or find a preprocessing or postprocessing techniques (e.g., data transformations) for *bzip2* that would improve the efficiency with respect to compression, real-time performance, space and time complexity, while satisfying the nonrestrictive software usage constraint.

Like the weather, weather radar data have spatial and temporal variation. Consequently, CR statistics were computed for archived level II radar data taken from different radar sites, at different times of year, under different types of weather conditions, with different radar scanning strategies using *bzip2,* UNIX compress, and 0-order Huffman algorithms. In all cases, *bzip2* had the largest CR (Kelleher et al. 2003), which is not surprising given *bzip2* is a hybrid algorithm. Consequently, the remaining discussion will focus on the *bzip2* results.

A total of eight datasets representing seven weather events were examined. Four of the eight datasets represented precipitation events; one was a squall line event, one a hurricane event, and the other two were widespread stratiform precipitation events. The remaining four datasets represented clear-air events. Two of the clear-air datasets represented the same low-level jet event, but with data collected from two separate radars using different volume coverage patterns and radar operating characteristics. In addition to being selected for the weather they represent, the datasets were chosen such that there were two datasets for each of the operational volume coverage patterns (VCPs): 11, 21, 31, and 32 (OFCM 2006) in use at that time. Descriptions and characterizations of the eight datasets examined in this paper are given in Smith et al. (2002).

Calculations showed that for the precipitation events analyzed, the *bzip2* CR range was between 6:1 and 11:1. For the clear-air events analyzed, the CR range was between 10:1 and 25:1. One important finding was that the CR is improved by simply preprocessing the data before it is compressed and transmitted. More specifically, preprocessing radial messages by setting the Channel Terminal Manager header and trailer to zeroes, removing all data located higher than 70,000 ft above the radar, padding the end of the message record with 0s, and transforming/compressing the message metadata offered incremental improvement in overall compression by 10%–30% with no changes required to the software on the user end.

Although results showed *bzip2* performed well overall, it occasionally had difficulty meeting bandwidth limitations (i.e., 56 Kb s^{-1} leased phone lines) when processing data collected during squall line events. There were several possible explanations for this observation. One explanation was the tendency of the National Weather Service (NWS) to operate the radar in VCP 11, since this VCP has the fastest update rate and, consequently, produces the largest data rate. Another explanation was that squall lines are characterized by large gradients in the moment data. Large gradients limit the effectiveness of compression techniques that rely on spatial correlation. Regardless of the reason, four solutions to this problem were considered: 1) increase the bandwidth of the communications line beyond 56 Kb s^{-1}, 2) decrease the data rate over the communications line (possibly resulting in unacceptable latencies), 3) improve the compression for the existing data rates, or 4) some combination of the above. The problem was solved by the NWS during national implementation of CRAFT in 2004 by increasing the available bandwidth for local data manager (LDM) transmission of level II data to 128 Kb s^{-1} per radar and preprocessing the radar data in the Base Data Distribution System (BDDS) as described above.

EXAMINATION OF NETWORK PROPERTIES. *Bandwidth.* When the radar generates data at a rate exceeding the link's capacity, the data are stored temporarily in a local queue by the LDM software. The data generation rate is based heavily upon the weather conditions and volume coverage patterns being used. On average, the rate was approximately 40 Kb s^{-1}. In highly convective situations, the rate approached 64 Kb s^{-1}, a common commercially available telecommunications bandwidth. In rare cases, the bandwidth exceeded 64 Kb s^{-1} and approached 77 Kb s^{-1}, resulting in buffering and increased latencies.

Figure S1 shows the capacity of the communications links for the original CRAFT network in Oklahoma at the time of the simulation. Network provider OneNet was connected to the Abilene backbone (2.4 Gb s^{-1}) using a T3 line (45 Mb s^{-1}). Given the aggregated data transfer for CRAFT data were only a few hundred kilobits per second, bandwidth was not an issue for either connecting to Abilene (T3) or transmitting over Abilene (2.4 Gb s^{-1}).

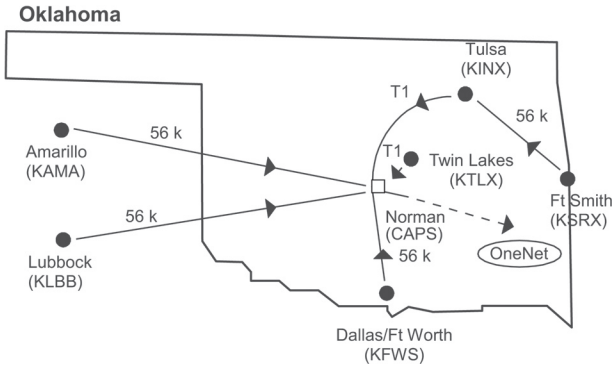

Fig. S1. The original CRAFT network as of January 1999. The six radars, shown as circles, transmitted compressed level II data in real time over dedicated T1 and 56 Kb s^{-1} phone lines simultaneously to ingest computers to Norman, OK. The data were sent to NCDC for archival via a T3 line provided by OneNet.

However, for those radars connected using 56 Kb s^{-1} phone lines, network congestion occurred when data were generated at rates higher than 56 Kb s^{-1} as observed, for example, during squall line events. Therefore, for squall line events, the local buffer was needed to temporarily store the data for eventual transmission during periods of lower bandwidth usage.

Local buffer. It was important to know under what conditions (e.g., transmission circuit failure, stormy weather) the local storage buffer becomes engaged indicating data latencies will begin to increase. For the early CRAFT network, overflows occurred during squall line events in which data were generated at rates exceeding 56 Kb s^{-1} bandwidth. Note that the total bandwidth needed is dependent upon the data rate of the compressed radar data plus the bandwidth needed to support both the TCP/IP transmission and LDM. If data rates only exceeded the maximum circuit bandwidth for a short period of time, then the latencies due to buffering were small as the buffer was quickly flushed and LDM caught back up to transmitting the data in real time. However, if the high data rates persisted, or during periods of extended circuit failure, then it was possible to exceed the storage capacity of the local buffer and, therefore, data were permanently lost from the buffer and latencies increased to the point that the real-time benefits of these data were lost. Note that LDM does have a feature that allows data to be written to disk and recovered manually via FTP after the event (e.g., an extended circuit failure), but these data are not automatically retransmitted.

To examine the effects of high data rates or circuit failure, a simulation was conducted using a buffer size of 100 mb (Kelleher et al. 2002). Figure S2 shows the amount of time (in hours) before the 100-mb buffer reaches its capacity for data rates exceeding 56 Kb s^{-1}. In the extreme case of a total circuit failure, a 100-mb buffer will fill in approximately 4 h assuming average data rates (40 Kb s^{-1}). Once the buffer reaches its capacity and overflows, the data will no longer be automatically transmitted. To prevent a buffer overflow, it was tempting to increase the buffer size. However, if the buffer was made too large it would again have resulted in losing the real-time benefits of the data since the LDM software flushes the older data stored in the buffer first (when bandwidth becomes available), before sending the most recent data. The result is that it could take hours or days for the buffer to flush and for LDM to catch up to real-time transmission if only a fraction of the 56 Kb s^{-1} circuit is available (i.e., the difference between the new data still arriving and the 56 Kb s^{-1} circuit is small). As a result, the most recent data become old, aging in the long buffer queue.

From Fig. S2, it can be seen that it will take approximately 35 h before a buffer overflow occurs for a data rate of 64 Kb s^{-1} (i.e., the data rate exceeds the circuit capacity by 9 Kb s^{-1} = 64 Kb s^{-1} – 56 Kb s^{-1}) and only 10 h at 77 Kb s^{-1}, the maximum data rates observed. Since the data generation rate fluctuates under real-time operational conditions, the exact time of data overflow can only be estimated.

Figure S3 shows the theoretical minimum amount of time needed to drain different size buffers for a 56 Kb s^{-1} circuit. Note that this assumes that the entire 56 Kb s^{-1} circuit will be available to flush the buffer, which is unrealistic since the new level II radar data will continue to compete for bandwidth over the same circuit, effectively reducing the available bandwidth to flush the buffer.

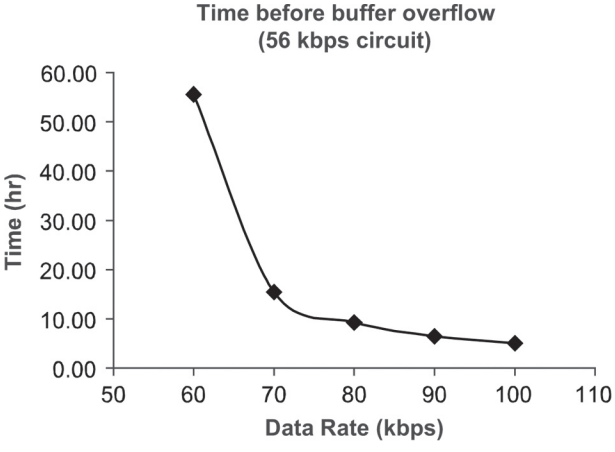

Fig. S2. Time needed to overflow a 100-mb buffer using different data rates assuming a 56 Kb s^{-1} circuit.

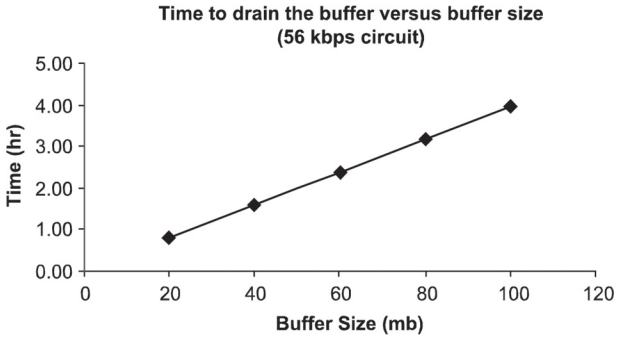

Fig. S3. Theoretical minimum amount of time required to empty the local buffer for various buffer sizes (assumes entire 56 Kb s^{-1} capacity circuit available to empty buffer).

In summary, a compromise was needed in the choice of a buffer size. The buffer size needed to be large enough to avoid unnecessary data loss due to temporary circuit failure (to maximize data archival success), yet small enough to avoid undesirable aging of the most recent level II data in the queue while the buffer was being flushed. Timeliness of the radar data was a primary concern of operational forecasters in both the public and private sector. The choice of a buffer size of 100 MB seemed reasonable in that it allowed the capture and automatic transmission of several hours of data when a circuit temporarily failed, while allowing acceptable latencies caused by transient high data rates from extreme weather events.

Latency. Ensuring data reliability was considered critical during the early design phases of CRAFT, but equally important was data latency. Low latency was important when consideration was given to the needs of the researchers at NSSL developing warning decision support systems for the operations (e.g., NWS forecasters) and for private sector weather companies engaged in providing live data for clients, as mentioned in the previous section. A study funded by the National Oceanic and Atmospheric Administration's High Performance Computing and Communications (HPCC) program was conducted to investigate the causes and reduce the size of the latencies observed in CRAFT.

The latency between two nodes may be calculated as follows:

Latency = Transmit Time + Link Propagation Time + Queue Delay
Transmit Time = Data Size/Bandwidth
Link Propagation Time = Distance/Speed of Light.

In the real world, available bandwidth can vary, so transmit time can vary and be nontrivial. Under the ideal conditions of a simulation, transmit time and link propagation time are fixed for a given network topology (e.g., a 56 Kb s^{-1} circuit) and given data rate. Latency, under these assumptions, is primarily a function of the queue delay. For the purposes of studying the CRAFT network, the queue delay was assumed to include both the delay at the local buffer and the delay associated with the routers during the transfer through the Internet.

Isolating the portion of the queue delay attributable to routers was difficult since the Internet is highly dynamic. Isolating the delay due to the local buffer was easier since it is strictly the transfer rate of data from the PC buffer onto the network. Therefore, the local buffer delay could be predicted quite accurately for given input and output data rates.

For example, if the data are generated at 100 Kb s^{-1} for 6 h ($x = 6$), the average queue delay is calculated to be 1.3 h. However, for the same situation, the delays due to insufficient bandwidth would exceed 16 h. This can be seen from Fig. S2 that shows at 100 Kb s^{-1} the entire 100-MB buffer fills, while Fig. S3 shows the time to drain a 100-MB buffer using all 56 Kb s^{-1} bandwidth is 4 h. If we have only 14 Kb s^{-1} bandwidth (the difference between an average data rate of 42 and 56 Kb s^{-1}) or one-fourth the bandwidth, it will take 4 times longer to empty the buffer.

In summary, network simulation has shown that the latency attributable to the queue delay is small when compared with the latency attributable to insufficient bandwidth (transmit time). Insufficient bandwidth causes local data buffering as previously described and causes large delays in transmission of the data. These delays can be significant for prolonged stormy weather events or when caused by network outages. Consequently, it was clear that a well-designed real-time network must avoid these latency delays by increasing the transmission circuit to greater than 56 Kb s^{-1}. The final NWS implementation used 128 Kb s^{-1} circuits, which essentially eliminated latencies due to insufficient bandwidth.

DETAILS OF DATA MINING LEVEL II DATA. A study was conducted to detect and classify mesocyclone signatures in Weather Surveillance Radar-1988 Doppler (WSR-88D) radar data using mining techniques (Xiang et al. 2004). Two approaches were chosen, one by the National Severe Storms Laboratory (NSSL) and one by the University of Alabama in Huntsville (UAH), but both involved Mesocyclone Detection Algorithms (MDA). NSSL's approach was to use a computationally optimized version of the existing NSSL MDA (Stumpf et al. 1998). NSSL MDA is an enhancement to the Build 9 WSR-88D Mesocyclone Algorithm (B9MA; Zrnic et al. 1985). Compared to B9MA, NSSL MDA identifies a broader spectrum of circulations and has an improved probability of mesocyclone feature detection. The latest version of

the NSSL MDA also includes a neural network classifier designed to diagnose which circulations detected by the NSSL MDA yield tornadoes (Marzban and Stumpf 1996). UAH's approach (called UAH MDA) was to use image processing techniques to identify mesocyclone signatures. However, it was necessary to create an embedded form of NSSL's MDA, based upon the original NSSL algorithm, to work with the UAH image processing techniques. The approach consisted of two components. The first step by each group was to ensure the validity of the mesocyclone detections using a historical dataset previously collected and manually verified by NSSL researchers. The second step was for each group to develop a mining algorithm fast enough to efficiently process large volumes of archived WSR-88D data.

NSSL researchers optimized the MDA code and achieved a speed up of approximately 20%. This result was not considered significant, although other areas for possible improvement in the future were identified. UAH, however, had notable success in their approach. Although the UAH algorithm differed somewhat in MDA detection from the NSSL algorithm, the results were reasonable.

The primary difference between the NSSL and UAH MDA algorithms was in the technique used to segment the two-dimensional mesocyclone signatures. The UAH MDA used a region growing technique that removed the restriction due to the shape of the feature. The UAH researchers assigned true or false labels to the mesocyclone features generated by the MDA by comparing with a truth set derived by an NSSL expert using the NSSL algorithm. This labeled feature dataset was then used in a series of analysis experiments.

The performance of the two algorithms was compared for two cases, 6 May 1994 and 11 May 1992 from Norman and Tulsa, Oklahoma, respectively, using the Critical Success Index (CSI) to evaluate the classifier performance (Stumpf et al. 1998). In general, the spatial coverage and feature distribution of identified mesocyclones from the two MDAs for the two cases were found to be very similar. The UAH researchers took the identified mesocyclone signatures for the two cases and applied them to two classifiers using three different feature selection methods to reach the best classification performance. They found the most important features were the vertical description of the signatures and the strength of the rotation.

In summary, the UAH MDA algorithm was about twice as fast as a variant of the computationally optimized full NSSL MDA. However, the best NSSL MDA algorithm CSI scores were higher than the best UAH MDA CSI scores, 0.56 and 0.61 versus 0.31 and 0.41 for the 6 May and 11 May case, respectively. Both the NSSL and UAH algorithms appeared to identify similar mesocyclone features and feature locations. Although additional analysis was recommended to ensure the UAH MDA correctly identified important mesocyclone signatures, it appeared from the first study that the UAH MDA could be considered for use as a tool to mine the mesocyclones from large volumes of archived radar data (Xiang et al. 2004).

REFERENCES

Kelleher, K. E., S. Y. Low, and S. Lakshmivarahan, 2002: Project CRAFT: Network topology and initial analysis of network loads. National Severe Storms Laboratory Tech. Memo. 107, 31 pp.

——, S. D. Smith, and S. Lakshmivarahan, 2003: Compression of NEXRAD (WSR-88D) radar data for real time Internet dissemination. National Severe Storms Laboratory Tech. Memo. 108, 33 pp.

Lelewer, D. A., and D. S. Hirschberg, 1987: Data compression. *ACM Comput. Surv.,* **19,** 261–296.

Marzban, C., and G. J. Stumpf, 1996: A neural network for tornado prediction based on Doppler radar-derived attributes. *J. Appl. Meteor.,* **35,** 617–626.

OFCM, 2006: Doppler radar meteorological observations, Part A. System concepts, responsibilities, and procedures. Federal Meteorological Handbook No. 11, FCM-H11A-2006, 50 pp. [Available online at www.ofcm.gov/fmh11/fmh11.htm.]

Smith, S. D., K. E. Kelleher, and S. Lakshmivarahan, 2002: Compression of NEXRAD (WSR-88D) radar data using Burrows–Wheeler algorithm. Preprints, *18th Conf. on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology,* Orlando, FL, Amer. Meteor. Soc., 133–135.

Stumpf, G. J., A. Witt, E. D. Mitchell, P. L. Spencer, J. T. Johnson, M. D. Eilts, K. W. Thomas, and D. W. Burgess, 1998: The National Severe Storms Laboratory mesocyclone detection algorithm for the WSR-88D. *Wea. Forecasting,* **13,** 304–326.

Witten, I., R. Neal, and J. Cleary, 1987: Arithmetic Coding for data compression. *Comm. ACM,* **30,** 525–540.

Xiang, L., and Coauthors, 2004: Mining NEXRAD radar data: An investigative study. Preprints, *20th Conf. on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology,* Seattle, WA, Amer. Meteor. Soc.

Zrnic, D. S., D. W. Burgess, and L. D. Hennington, 1985: Automatic detection of mesocyclonic shear with Doppler radar. *J. Atmos. Oceanic Technol.,* **2,** 425–438.