8.3

A DISTRIBUTED ARCHITECTURE FOR THE WSR-88D (NEXRAD) RADAR PRODUCT GENERATOR (RPG)

Allen Zahrai[1], Zhongqi Jing[1,2], and Neil Peery[3]

[1]NOAA/ERL/National Severe Storms Laboratory,
Norman, Oklahoma

[2]Cooperative Institute for Mesoscale Meteorological Studies,
University of Oklahoma, Norman, Oklahoma

[3]NOAA/NWS/WSR-88D Operational Support Facility,
Norman, Oklahoma

1. INTRODUCTION

NEXRAD is a network of Doppler weather surveillance radars (WSR-88D) which is replacing an aging fleet of non-Doppler meteorological radars operated by the National Weather Service (NWS), the Air Weather Service (AWS) and the Federal Aviation Administration's (FAA). The WSR-88D features significant improvements compared with earlier meteorological radars both in technology and in meteorological measurements. As a fully coherent Doppler radar, it provides not only accurate reflectivity information, but also radial velocity and velocity dispersion. Many other useful meteorological products are also derived from base data and distributed to users.

As the WSR-88D network reaches fully operational status, users' demands for enhancements and new features are expected to continue and grow. However, architectural characteristics of the existing system make functional enhancements difficult to achieve. In fact, many of the improvements proposed by the NEXRAD Technical Advisory Committee (TAC) and user agencies are believed to be beyond the capabilities of the existing system.

In response to increasing user demand for additional capability and improved performance and to ensure long-term viability of the WSR-88D system, a new architecture based on distributed open system technology has been adopted. In contrast with the original proprietary architecture, the new system is based on commercial off-the-shelf (COTS) equipment and open system standards. The purpose of this paper is to briefly introduce the new architecture and describe it's general characteristics.

2. SYSTEM CHARACTERISTICS

The WSR-88D radar consists of three primary functional areas; Radar Data Acquisition (RDA), Radar Product Generator (RPG), and the Principal User Processors (PUPs). The RDA subsystem provides real-time monitoring and control of the antenna, the transmitter and the receiver. The RDA also contains a digital signal processor (DSP) subsystem for estimation of base data. The RPG ingests base data radials from the RDA, produces meteorological products, and distributes these products to the users. In addition to product generation and distribution, the RPG also provides overall system command and control through the Unit Control Position (UCP). Users can access and display WSR-88D products using a variety of equipment generally connected to the RPG by dedicated and dial-up serial lines. Raw radial data received from the RDA is also available to external users through special dedicated high-speed interfaces.

Architectural limitations of the existing system result from a design which is rigidly tied to a legacy computing platform. In this design, a single computer often performs many computationally dissimilar functions. For example, within the RPG functional area, a single computer performs large number of compute-intensive algorithms as well as extensive amount of input/output (I/O) operations.

Long-term viability of the WSR-88D system mandates transition to an open system architecture. Open system standards are international standards for hardware and software where products and modules from different vendors can interoperate. These standards were established to make modular replacement and incremental upgrades relatively simple and cost effective.

Although this paper presents the new architecture for the open system RPG (ORPG), system characteristics and concepts presented here are readily applicable to other functional areas.

3. GENERAL REQUIREMENTS

Many architectural goals for the WSR-88D system can be achieved by a suitable open system implementation. Other desired characteristics can be provided by using distributed processing concepts and client/server technology. These goals and characteristics can be summarized as follows:

- The architecture must be modular and scalable. It must readily accommodate modular replacement and incremental upgrades. The system must be able to evolve with new standards and technologies.
- Vendor independence. Avoid proprietary hardware and software environments. Application software must be designed to be portable at least at the source level.
- Protection against technological obsolescence. Application software can migrate to different hardware platforms, preserving the investment in software development.

Corresponding author address: Allen Zahrai, National Severe Storms Laboratory, 1313 Halley Circle, Norman, OK 73069-8480; e-mail <Allen.Zahrai@noaa.gov>

- Interoperability between dissimilar functional modules. Incremental upgrade and enhancement reduces the cost and risk associated with planned product improvements.
- Reduction of complexity to develop new or improved meteorological algorithms, allowing rapid deployment of enhancements.
- The new architecture must provide improved local and wide area connectivity for reliable and efficient delivery of products and base data to an increasing number of users.
- The architecture must accommodate fault tolerance and dynamic response to overload conditions.

4. OPEN SYSTEM ARCHITECTURE

An open system is defined as a system which implements widely adopted public specifications for interfaces, services, and supporting formats to enable properly engineered applications software to be ported across a wide range of systems with minimal changes, to interoperate with other applications on local and remote systems, and to interact with users in a style which facilitates user portability. An open system provides a set of protocols, interfaces, and services to enable users and applications transparent access to data, resources, and other services across a heterogeneous network.

Transition of the RPG to an open systems environment involves development of a suitable architecture and use of applicable standards. The RPG can be most effectively implemented as a distributed client/server system. Since the RPG is essentially an algorithm engine and a communications processor, separation of these two functions provides the primary foundation for the new distributed architecture. Figure 1 illustrates such a distributed architecture. In this environment product servers implemented in the processing elements ingest raw radial data from the RDA, execute algorithms and deliver the results to their clients. The communications server consists of dedicated communications processors which deliver base data from the RDA to internal and external users, retrieve products from product servers and distribute them to the users. A Local Area Network (LAN) provides communications medium between functional nodes. Processing elements consist of general purpose workstations and nodes with specialized functions, such as file servers, routers, display terminals, etc.

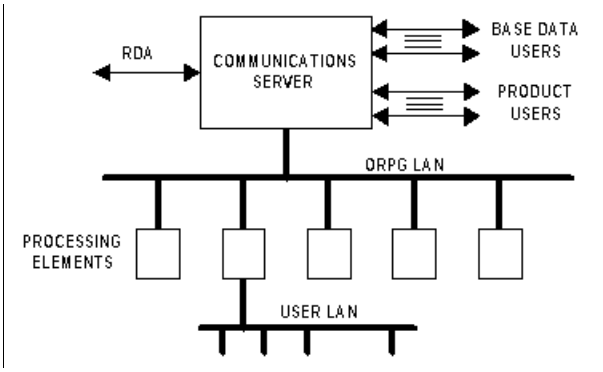

Figure 1: A Distributed ORPG Architecture

The communications server in Figure 1 must handle real-time base data traffic between the RDA and any number of internal and external clients. Internal clients are the processing elements on the LAN and external clients are base data users' equipment connected to the base data ports. The server also must handle all network traffic including interactive product distribution to product users connected through serial dedicated and dialup lines. Using a single communications server for both base data and product traffic not only requires high performance equipment, but it presents a single point of failure for the entire system. Figure 2 illustrates a refinement of the previous architecture, where base data and product distribution functions are separated and isolated. Narrowband communications is handled by an access server. This architecture provides improved modularity in each area as communication demands change. For example, base data distribution to external users may be handled by a server in the RDA functional area.

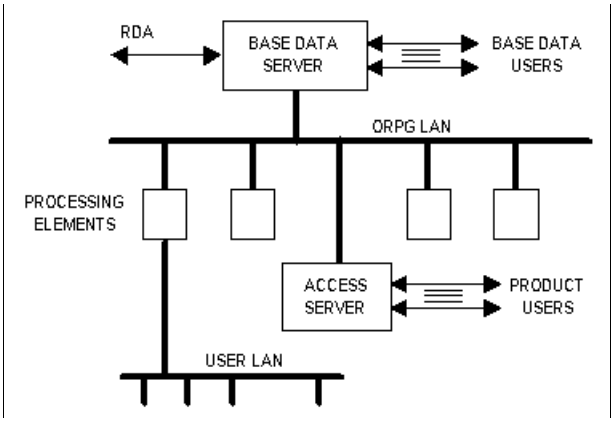

Figure 2: Separate Communications Server Functions

Within the distributed ORPG; system resources and user functions are distributed among computing nodes coupled through a LAN. Within this environment; application programs can be dynamically allocated to system nodes according to predefined application classification.

The use of COTS equipment, standard operating system, client/server, and networking technologies offer many advantages beyond improved maintainability and support. A distributed system allows dynamic load distribution and rapid performance enhancements. A networked architecture provides connectivity with other computer equipment as well as with remote users through an internetworked infrastructure. Figures 1 and 2 show the ORPG connected to another LAN segment (USER LAN) through a router or similar type of device. Improved connectivity facilitates remote maintenance, troubleshooting, and software updates. Encapsulation of functionality into independent application modules with a well-defined application programming interface (API) reduces the scope and side effects of software changes, and allows modular development and code reuse.

5. SOFTWARE MODEL

The ultimate goal of any open system implementation is application portability. To achieve this goal, application programs must be encapsulated and isolated from the underlying service layers. All interaction between the application program

and the system takes place through a set of functions established and defined in the API. This functional encapsulation will ease debugging and allow concurrent development. If addition of new algorithms require increased processing power, another processor can be added without redesigning the entire system. Closely related algorithms such as hydrometeorological algorithms can be grouped together and assigned to a dedicated server.

In order to take advantage of the distributed architecture, every functional element within the ORPG must be isolated and encapsulated within an application module. An application module is characterized by a well-defined functionality, and interacts with the rest of the system through a standard API. Application programs so designed, can be dynamically allocated on any qualified node either by the operating system or user command. Figure 3 illustrates a typical set of functions for the ORPG.

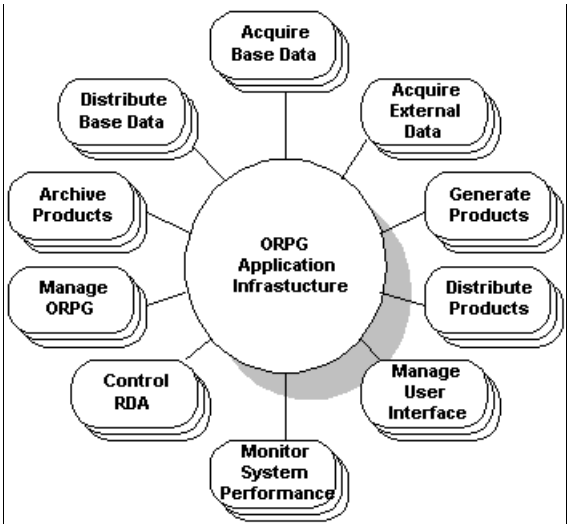

Figure 3: Typical ORPG Functions

An application module consists of one or more interrelated processes. Processes within an application module use local Inter Process Communication (IPC) facilities of the operating system. An application, however, can only communicate with another application or other system resources using a standard set of procedures defined by the API specifications. Isolating application programs from the underlying details allows dynamic allocation and load distribution.

A layered programming model is essential in the design of portable software. This model delineates the programming environment into functional layers, where each layer is characterized by the services it provides to the adjoining layers through a set of standardized calls. The RPG can be described as a data driven system; all processes in general, respond to events consisting of either request for or availability of a data item. Figure 4 shows a software model for a typical ORPG node. In this figure, the network layer includes network access, internet, and host-to-host layers of the TCP/IP protocol. The middleware between the network and application layers represents the ORPG application infrastructure layers. The interface between the application programs and the operating system is completely defined in the API specifications and is highlighted in this figure. It is important to note that other system services not directly related to ORPG functions are also represented by application modules.

The ORPG Application Programming Library (APL) implements API functions for application modules. Application programs utilize a set of standard calls to access system services. Library functions interpret and transform these calls for lower layers. For example, when an application module needs a particular data item, it simply requests the item by some logical name. The application does not know or care where the data are located nor how it will be retrieved. Functions in the library locate the data and retrieve it with the aid of lower level services. User applications are completely isolated from the underlying implementation details.

Dynamic binding is essential in a distributed environment in order to facilitate dynamic resource allocation. That is, applications must not refer to data items and files by their physical attributes, because either the application or the data or both may be relocated to another location by the system in response to some condition. Within the ORPG, all data which are not local to an application, are globally defined by well-known logical descriptors. The system maintains mapping tables to translate logical names to physical locations.

Data distribution in the ORPG environment is derived from information modeling concepts, where global data are maintained and distributed through well-known data storage areas. This method decouples application modules from each other such that one module's response or failure does not affect the other. Actual nature and characteristics of the data stores are defined by the system designers, and hidden from the application programs.

Events generally refer to asynchronous phenomena requiring immediate system intervention. Events occur either from exceptional conditions such as component failures or user interaction. Events are well-known global entities and processes register for sourcing or receipt of specific events. Event queues and signaling are managed by a system level process on each ORPG node.

Data distribution, event notification, and other distributed system services are represented in Figure 4 as the distributed application services layer. Functions in this layer build upon and extend standard UNIX (POSIX) system calls to provide a robust and efficient distributed processing environment.

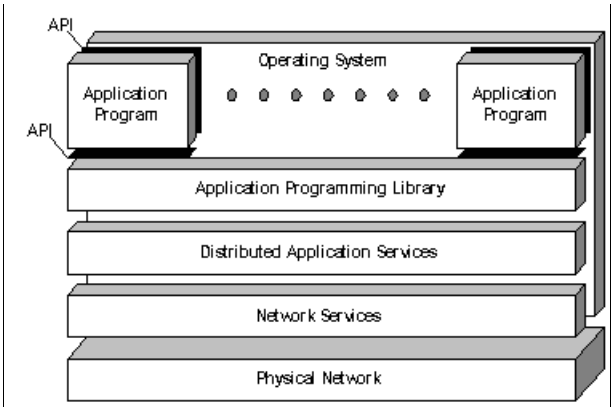

Figure 4: Software Model for a Typical ORPG Node.

It is important to note that functionality provided by the Distributed Application Services layer is not specific to the ORPG.

6. STANDARDS

The success of any open system implementation depends to a large degree upon adoption and adherence to applicable public standards. Such standards directly contribute to an application development framework established to ensure software portability and code reuse. A comprehensive Open System Environment (OSE) standards profile is currently being developed for the ORPG through a technical consensus process in consonance with other modernization efforts in the NWS. A brief summary of applicable standards will be presented here for completeness.

IEEE Standard 1003.1-1988 was the first of a group of proposed standards collectively known as the Portable Operating System Interface for Computer Environments (POSIX). In 1990 POSIX was adopted by the ISO and designated ISO/IEC 9945-1:1990. POSIX is a rich set of standards covering a wide range of operating system services. A conforming POSIX application can move from system to system with a very high confidence of low maintenance and correct operation. Therefore, POSIX conformance is a primary requirement for all ORPG application programs. Although POSIX does not specify or require a specific operating system, it is based on the UNIX operating system. Therefore, many UNIX platforms offer POSIX conformance.

The primary programming language is C (FIPS 160 C, ANSI/ISO 9899:1992). C is a general purpose high-level language designed for use in all levels of software. Fortran (FIPS 69-1, ISO 1539:1980, ANSI X3.9-1978) is a high-level language used primarily in scientific applications. Use of Fortran shall be restricted to; existing algorithms, and under special circumstances to new algorithms.

The Open Systems Interconnection (OSI) protocols define connectivity standards for both local (LAN) and wide (WAN) area networking. The existing RPG uses an OSI protocol known as X.25 for communicating with the RDA and the external users. Open system RPG shall provide X.25 support for existing users and TCP/IP protocol support for network connectivity. Remote connections to the RPG in the future may use either X.25 or TCP/IP protocols as requirements dictate.

The user interacts with the ORPG through the master system control function (MSCF) interface. In the distributed environment these functions may be accessed from different nodes depending on local configuration. Therefore, a device independent graphic user interface (GUI) is required for the ORPG environment. X Window System originally developed at MIT to fulfill a need for a distributed, hardware independent user interface, has long been adopted by a consortium of hardware and software vendors and the government as standard base for user interfaces. X provides a client/server infrastructure for user interface. X does not define any particular user interface style; i.e. “look and feel”. Open Software Foundation (OSF) has developed a standard user interface toolkit for X known as Motif. The standard user interface for ORPG shall be based on X Window System and Motif style.

Computer Aided Software Engineering (CASE) tools shall be used in forward and reverse engineering processes to produce design documentation and facilitate maintenance.

7. ACKNOWLEDGMENT

The WSR-88D is jointly maintained and operated by the National Weather Service (NWS), a component of the National Oceanic and Atmospheric Administration (NOAA), in the Department of Commerce (DOC), the Air Force’s Air Weather Service (AWS), in the Department of Defense (DOD), and the Federal Aviation Administration (FAA) in the Department of Transportation (DOT). The primary support organization established by the three agencies is the WSR-88D Operational Support Facility (OSF). The authors also wish to acknowledge the NWS Office of Systems Development (OSD) which provides project management and support for the WSR-88D evolution. The National Severe Storms Laboratory (NSSL) of the Environmental Research Laboratories (ERL), a component of NOAA in the DOC along with the OSF and the OSD are jointly responsible for the development and testing of many enhancements for the WSR-88D. The NSSL along with other ERL laboratories provide technical leadership and guidance in further understanding and utilization of the WSR-88D system.

8. REFERENCES

Saffle R.E., L.D. Johnson: *NEXRAD Product Improvement Overview*, Preprints 13th IIPS, Longbeach, CA., Amer. Meteor. Soc., paper 8.1

Institute of Electrical and Electronic Engineers, *IEEE Std 1003.0-1995, Guide to the POSIX Open System Environment,* IEEE Standards Board, Piscataway, New Jersey

International Standards Organization/International Electrotechnical Commision (ISO/IEC), 1989, *Information Processing Systems - Open Systems Interconnection - Basic Reference Model,* ISO/IEC 7498-4, Geneva, Switzerland

NOAA, *NEXRAD Technical Requirements*, R400-SP401A, Joint Systems Program Office, Silver Spring, MD

NOAA, *Tri-Agency Requirements for Operational Use of Weather Radar Data*, Draft Document May 31, 1996, NWS Office of Systems Development, Silver Spring, MD

NOAA, *Federal Meteorological Handbook, No. 11,* Federal Coordinator for Meteorological Services and Supporting Research, National Oceanic and Atmospheric Administration, US Department of Commerce, Rockville, MD, 1989