Earth Sci Inform
DOI 10.1007/s12145-010-0043-6

RESEARCH ARTICLE

A flexible streaming software architecture for scientific instruments

Martin Grill · Keith Barratt · Farideh Honary

Received: 24 August 2009 / Accepted: 11 January 2010

Abstract The recently completed prototyping efforts for a new type of riometer, the Advanced Rio-Imaging Experiment in Scandinavia (ARIES), required the development of a uniquely flexible software architecture to deal with what in software engineering terms is referred to as a 'Wicked System:' Source, volume and type of data as well as required processing are only very loosely defined at the outset of the project. Speed, reconfigurability, remote control and data provenance are of major importance for the success of the project both during development and during operation of the deployed prototype. Details of the Advanced Riometer Components (ARCOM) component-based software architecture are presented. The software architecture is not specific to ARIES, and ARCOM components can readily be re-used in other, similar instruments.

Keywords Componentry · Geospace · Pipelining · Provenance · Software architecture · Streaming

Communicated by T. Narock

Part of this work was funded by the UK's Particle Physics and Astronomy Research Council (PPARC), now the Science and Technology Facilities Council (STFC).

M. Grill (✉)
SRI International,
Menlo Park, CA, USA
e-mail: martin.grill@sri.com

K. Barratt · F. Honary
Lancaster University,
Lancaster, UK

Introduction

Riometers (Relative Ionospheric Opacity Meter using Extra Terrestrial Electromagnetic Radiation) (Little and Leinbach 1959; Detrick and Rosenberg 1990) measure to what extent cosmic background noise is being absorbed by the ionosphere. The largest 16×16 (256-element) phased array imaging riometer at Poker Flat, Alaska (Murayama et al. 1997) achieves an angular resolution of 6° at zenith which translates into an area of about 11×11 km at a height of 90 km. The need for higher spatial resolution provides motivation to prototype riometers based on the Mills Cross technique (Nielsen and Hagfors 1997; Mills and Little 1953; Nielsen 2001). Since increasing the number of antenna elements becomes increasingly impractical, such riometers employ digital receivers and more complex data processing combined with a sparse antenna array to achieve higher spatial resolution. A drawback of a Mills Cross type system, compared to a filled phased array system of similar resolution, is that the required integration time is worse due to the reduced aperture of the antenna array. This issue has been investigated in (Nielsen et al. 2004) and (Hagfors et al. 2003), and concludes that this effect is sufficiently counterbalanced by advancements in receiver technology over the last decades.

Operating software to support prototyping of the first Mills Cross based riometer needed to be flexible enough to support engineers and scientists during all phases of the development, ranging from initial single beam experiments and signal processing algorithm evaluations to final continuous operation of the fully deployed instrument.

In addition to actual absorption signals ("the data"), there is a strong need for collecting engineering and housekeeping data. This is particularly useful during instrument development, but remains of significant importance for potential troubleshooting during routine operation.

Published online: 13 February 2010

Having engineering data embedded in a (the) single stream of "data" allows for unambiguous reconstruction of the chain of events that resulted in any given data point in question ("provenance"). In the authors' experience, the ability to "replay" data has been an invaluable tool in engineering and troubleshooting the ARIES prototype. The ability to add new data to the stream (for example, adding a new temperature sensor or feedback from the ventilation system) without having to worry about separate log files and data merging, and without breaking any existing code, has turned out to be a big time saver.

Digital signal processing

A Mill Cross type riometer is based on the principle of cross-correlating signals from the two perpendicular fan beams formed by two perpendicular linear phased arrays as first conceived and used by (Mills 1952) and (Christiansen and Mathewson 1958), albeit not for riometry purposes. The beam forming process for a Mills Cross is illustrated in Fig. 1. The two small panels on the left show an example of a fan beam formed by a linear array of antennas along the y-axis (top panel) and along the x-axis (bottom panel), respectively. The small inset in the upper right hand corner of each panel shows an idealized top-down view of several fan beams generated by the arm in question, with the shown fan beam highlighted in green.

With this type of antenna arrangement, high spatial resolution can be achieved with significantly fewer antenna elements when compared to a filled array antenna: to achieve n times the resolution, the number of antenna elements required now only increases linearly with n and not n^2.

During initial prototyping, the first beam forming stage (the Butler Matrix forming fan-shaped beams) was carried out in analog electronics, and ARCOM software dealt with continuous cross-correlation and post-integration. Once the full bank of receivers had become available, forming of the now 32+32=64 fan beams was still carried out using analog electronics as shown in Fig. 2a. However, in this incarnation, cross-correlation was handed off to a field-programmable gate array (FPGA), as standard PC hardware cannot currently be easily interfaced with the 64 serial input lines delivering a total input data rate of 2.6 Gbits/s produced by the 64 simultaneously sampling receivers. Over time, the signal processing functionality has expanded its scope to include the Butler Matrix processing block and this is shown in Fig. 2b. With both beam forming stages being carried out by digital signal processing, a number of significant additional capabilities have become possible including programmable tapering of the array, improved temperature stability due to the reduction in analog electronics, and a better real-time diagnostic capability of the receiver boards.

ARIES operating software

ARIES exhibits many properties of a 'Wicked System,' defined as a system whose properties are poorly understood at the outset, and are likely to change significantly as

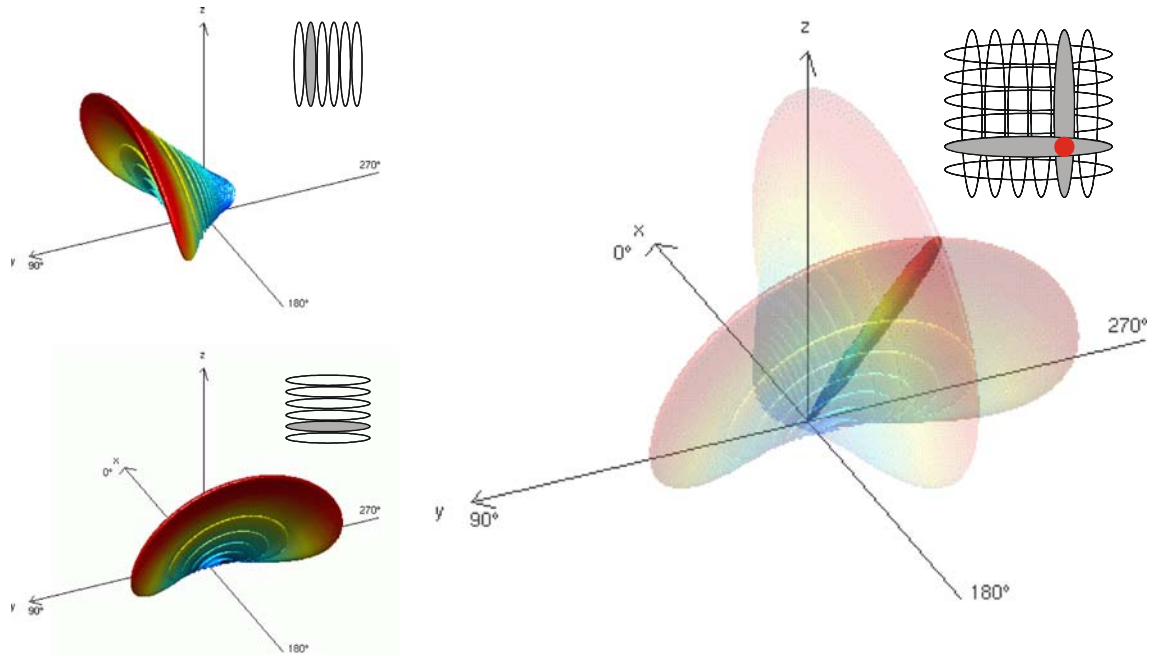


Fig. 1 Beam forming for a Mills Cross

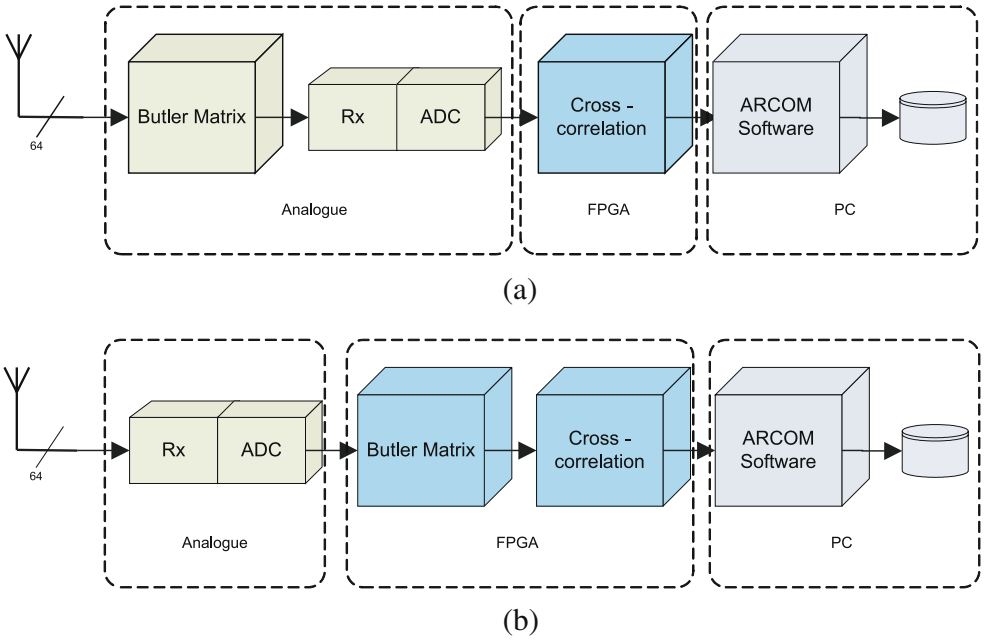


Fig. 2 ARIES System: **a** Original design using analog Butler Matrices for stage 1 beam forming; **b** Current implementation with digital beam forming

system development progresses from prototype to prototype (Sommerville 2004). Software design for such systems provides unique challenges to the software engineer seeking to support system evolution to the most flexible extent possible. The ARIES operating software is the result of a structured approach of defining goals, deriving the overall architecture and finally designing, implementing and integrating the individual (software) building blocks that make up the system.

Software goals

The ARIES operating software ARCOM (Advanced Riometer Components) is designed around a flexible component-based pipelining architecture derived from the following goals:

(i) Speed. On average, the system needs to keep up with the incoming data stream from the receivers. The phrasing 'on average' is appropriate, because even though we must not lose any data, we need not guarantee that all data is processed immediately. In engineering terms, this system can be referred to as a 'soft real-time' system.

(ii) Run-time reconfigurability. To a certain extent, we want to be able to reconfigure the system while it is running. For example we want to continuously integrate an incoming signal, and then later on add functionality for logging the incoming signal to a file. In this example, the logging should not affect the operation of the integrator.

(iii) Expandability. We want to be able to develop new system functionality in the future. This new functionality should integrate seamlessly with any existing parts of the system.

(iv) On-line status information. For diagnostic purposes, we want to be able to query the current internal status of the system in sufficient detail at any time, and without affecting the operation of the system.

(v) Remote control. Due to the physical remoteness of the site, all system functionality should be accessible from off-site and through potentially slow (56K modem) communication links.

ARCOM architecture

Figure 3 gives a layer-oriented overview of how ARCOM fits in with the other parts of an advanced riometer system. This section discusses major aspects of the ARCOM architecture.

Component-based

This approach allows for easy insertion and removal of components at runtime. Common Object Request Broker Architecture (CORBA) is used for controller-to-component communication, e.g. to start, stop or query the status of any given component. A set of three principal (meta-)components was designed. Every component in the ARCOM control software behaves like (is derived from) one of these principal components. The core ARIES control software is therefore made up of only three structurally different components, and even those have strong commonalities as far as inter-component communication is concerned. Recorder components stand at the beginning of a processing

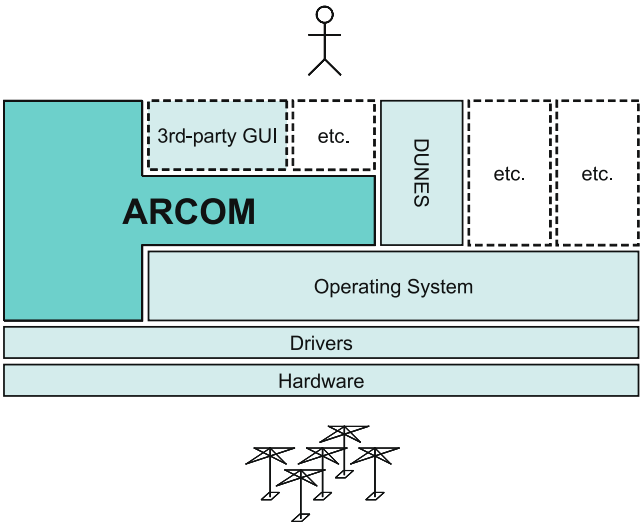


Fig. 3 Multi-layer view of ARCOM and its operating environment. DUNES is an independent dial-up networking module implemented by the author and is not discussed further in this paper

chain (pipeline). They collect data from some (hardware) device (for example the ARIES FPGA or a temperature sensor) and transfer it to the standardized shared memory interface. Processor components manipulate incoming data. Examples are post-integration, absorption calculation, or cropping (reducing the field of view). Processors can also be data sinks. The ALogger component, for example, takes data and writes it to disk. Finally, Adaptors simply provide a consistent CORBA interface to different hard- and software components supplied by third parties, following the 'façade' design pattern (Gamma et al. 1995; Deacon 2002). That way, third party components like, for example, the control software for an uninterruptible power supply (UPS), can be accessed through a standard CORBA interface, just like any other ARCOM component. Adaptors do not access shared memory interfaces.

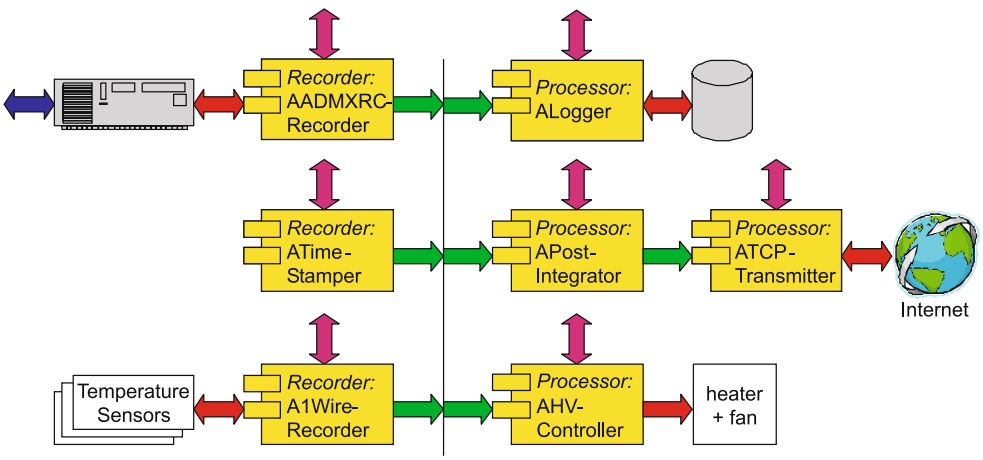
Pipelined

A pipeline architecture follows the natural notion of data flowing through the system, being processed as it does so. This model is not limited to one input and one output, several streams of data may enter the system simultaneously, to be processed independently or combined together. Results may be written to external disk straight away, or processed by components further down the pipeline. A pipelining architecture supports reuse of transformations, evolving the system by means of adding new transformations is straightforward, and concurrent systems (many processing paths processed simultaneously) are readily supported. Figure 4 is an example of several components arranged in a multi-pipeline setup. This depicts the ARCOM configuration for our current FPGA-based ARIES implementation. All recorder components feed into the same shared memory interface. The resulting data stream gets stored to disk by an ALogger component. Selected packets can be post-integrated and transmitted over the network in real time. Additional components can be added and removed at run-time, and command-line tools or additional components can be used to tap into any of the shared memory interfaces. The major disadvantage of a pipelining model is that each transformation needs to agree on a common input and output format in order to be able to tie in with the other transformations along the processing pipeline. This was turned to an advantage in ARCOM, see section below on ARCOM's versatile streaming data format.

Shared memory interface

In addition to the CORBA interfaces, which are used for infrequent control tasks such as starting and stopping components, all components that can take part in pipeline-

Fig. 4 ARCOM configuration for current FPGA-based design



based processing are glued together through blocks of shared memory with strictly unidirectional data flow. Establishing a shared memory interface for data flow between components in the processing pipeline allows for maximum speed as data is transferred between components, essentially only limited by memory throughput of the processing hardware. ARCOM shared memory interfaces are block-based with flexible block sizes. They are multi-client and multi-master and allow for multiple simultaneous read and write operations at any given time. Real-time diagnostics are supported to fine-tune the operating parameters of any given interface through functions to retrieve and visualize current allocation information. Figure 5 shows one example snapshot of the internal state of an ARCOM shared memory interface. Tall (green) arrows indicate head pointer position, short (black) arrows indicate tail pointer positions. Masters 0 and 2 are connected and idle. Master 1 is currently writing to the shared memory interface. Client 0 is waiting for Master 1 to finish. Client 1 has recently connected and is waiting for new data packets. Client 2 is disconnected. Note particularly how multiple masters can hold a write lock on their part of the data stream simultaneously, and how clients consume new data in sequence as masters finish writing out their respective packets.

ARCOM streaming data format

As noted above, any pipelining architecture requires its individual processing stages (components) to agree on a common data format for passing data along the pipeline. The flexibility requirements faced by ARCOM compared to existing riometer systems are unique in that the data format should accommodate for both existing (well-)defined data sources such as existing FPGA implementations of the ARIES cross-correlator, as well as data produced by yet-to-be-defined-and-implemented future components. Generic multi-purpose data formats have come a long way, with XML (eXtensible Markup Language) being the most prominent of these formats in recent years (Bray et al. 2006). XML is uniquely flexible in that it is an inherently extensible format that can be customized to represent any possible dataset. The major drawback of XML when it comes to inter-component communication between, in this sense closely coupled, components in a processing pipeline is its text-based format. Parsing XML data, which allows both variable-length tags and content, is processing intensive and therefore slow. It turns out that flexibility and speed requirements very similar to ARCOM's exist in quite a different area, namely digital television. Equipment compliant with the Digital Video Broadcast (DVB) standard uses pipeline architectures to multiplex, transmit, receive and de-multiplex streams of images, audio, subtitles and many additional, quite unrelated, datasets such as teletext or IP (Internet Protocol) packets. The underlying data format is the MPEG transport stream, see (ETSI 1997) and references therein. Inspired by this, ARCOM defines a common ARCOM streaming data format that, on the highest level, consists of a stream of 'packets,' each packet in turn containing one or many 'descriptors.' This format provides high flexibility as follows: Data in this format is future-proof (unknown packets/descriptors are simply skipped over by processing tools), generic (a basic set of tools can deal with arbitrary data streams), hardware-friendly (arbitrary external hardware data sources can be made to 'speak' ARCOM simply by encapsulating their data output into appropriate ARCOM packets, an approach very similar to 'tunneling' in, for example, TCP/IP networking) and fast to process, access, and archive. Grill (2007) gives details on all currently defined ARCOM packet and descriptor types. Figure 6 shows the definition of an ARCOM_FPGAPACKET packet as an example of a complex ARCOM streaming data packet. Note that although the figure shows the packet exactly as it is issued by version 4.0.1.2 of the FPGA firmware, ARCOM does at no stage rely on the detailed knowledge of descriptor offsets, order, or even existence of any particular descriptor. Instead, because the packet is made up of 'descriptor' entities, the packet can be scanned for descriptors of interest (as identified by their respective descriptor header), and the relevant data is extracted/processed, whilst unknown/uninteresting descriptors and packets that are known not to contain any descriptors of interest for the task at hand are simply skipped over. This is in contrast to fixed data structures that rely on every processing tool's knowledge of the precise location of any given data point in the structure and require updates to all tools to accommodate extra data.

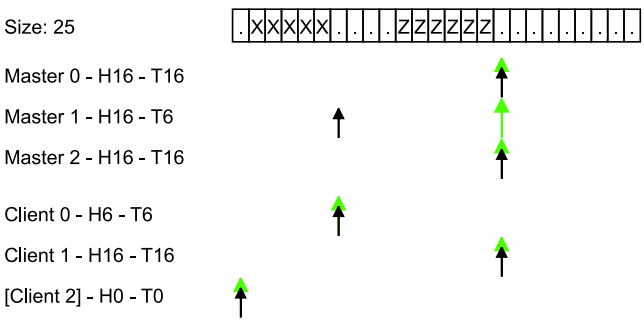


Fig. 5 An ARCOM shared memory interface in use

The authors would like to acknowledge that ARCOM's proprietary (although open and documented) data model generally requires at least an additional conversion step before the data can be fed into existing standard visualization/processing tools. While deemed an acceptable trade-off for our application at the time, this may be a serious commitment to ask from potential users.

Fig. 6 Example of an ARCOM_FPGAPACKET

Word Index	Packet Description				
0	magic (0xABDE)			packet type 0x0011	
1	total packet size in bytes – 0x0000_403C				
2	checksum (ignored) 0x0000_0000				
3	Xcorr Data Descriptor	Descriptor Header	Did 0x0a	0x4004	0x00
4 - 67		Real Outputs	Y0*X(0 to 31)	LSBs	0x0000 & MSBs
68 - 131			Y1*X(0 to 31)	LSBs	0x0000 & MSBs
132 - 195			Y2*X(0 to 31)	LSBs	0x0000 & MSBs
: :			: :	: :	
1988-2051			Y31*X(0 to 31)	LSBs	0x0000 & MSBs
2052-2115		Imaginary Outputs	Y0*X(0 to 31)	LSBs	0x0000 & MSBs
2116-2179			Y1*X(0 to 31)	LSBs	0x0000 & MSBs
2180-2243			Y2*X(0 to 31)	LSBs	0x0000 & MSBs
: :			: :	: :	
4035-4099			Y31*X(0 to 31)	LSBs	0x0000 & MSBs
4100	Timestamp Descriptor	Descriptor Header	Did 0x08	0x0014	0x00
4101		Hopf GPS Time	Hours(tens) & Hours(units) & Week day & status		
4102			Secs(tens) & Secs(units) & Mins(tens) & Mins(units)		
4103			Month(tens) & Month(units) & Day(tens) & Day(units)		
4104			0x0000 & Year(tens) & Year(units)		
4105	Information Descriptor	Descriptor Header	Did 0x09	0x0014	0x00
4106		Info 1	Number of Integrated Values for this packet		
4107		Info 2	Number of ADC conversions between PPS (Rising Edges)		
4108		Info 3	Upper 16 bits are noise word1; lowest bit is noise mode; bit 12 is the actual noise_en fed to Host		
4109		Info 4	Noise word 2		
4110	Version Descriptor	Descriptor Header	Did 0x10	0x0004	Version 0x??

Low-level support tools

A number of command-line based tools enable day-to-day operation and maintenance of ARCOM based systems and are part of the ARCOM distribution. These include the ARCOM command-line CORBA message dispatcher (sendcmd) to interact (i.e. start, stop, etc.) with running ARCOM components through their CORBA interface, a graphical user interface (gui1.tcl) to graphically control the state of ARCOM components, the ARCOM executor (executor.pl) to automatically start up and shut down a complete ARCOM-based system consisting of many components and shared memory interfaces between them, as well as several 'ARCOM packet tools' to spy on, retrieve, display the content of, filter, extract data from ARCOM packet streams and live shared memory interfaces.

Scope and limitations

It needs to be emphasized that the ARCOM architecture does not attempt to solve data streaming, routing and processing issues arising in (potentially wireless and/or ad-hoc) distributed sensor network applications (Akyildiz et al. 2002). The ARIES riometer instrument design still follows the more traditional design paradigm of many "dumb" sensors being attached to a small number of processing nodes. The primary goal here is to seamlessly integrate data from a variety of inputs into one common data "stream," route this stream to a (small) number of co-located and/or remote nodes for processing/archival, and do all this in a way that supports the goals presented in the section on "Software goals" above.

Neither does ARCOM (intend to) provide a solution for distributed query processing in sensor networks as

described, for example, by Gehrke and Madden (2004), or even an architecture for collective, distributed reasoning (Strohbach et al. 2004). Rather, ARCOM's pipelining architecture is inherently good for in-sequence processing of contiguous time series of data. Typical such processing tasks in the field of riometry are on-the-fly (real-time) visualization, beam forming, post-integration, filtering, and quiet-day curve derivation and subtraction.

Data dissemination to users for non-real-time purposes happens by the traditional means of transferring archived stream data (files) as generated, for example, by the ALogger component, for the time period of interest. A commonly used method in the field is to make these files available on a file transfer protocol (FTP) server. This is in line with current practice for resident archives (RA), but eliminates the need to archive multiple separate file types for any given instrument as all data is embedded into one single data stream.

Summary and outlook

The ARCOM software architecture provides a versatile run-time environment for scientific instruments and can readily be tailored to support a wide range of data acquisition and processing tasks. ARCOM is not limited to ARIES, or even riometers. The component-based approach, together with high-speed pipelining through dedicated shared memory interfaces, allows for unprecedented flexibility and run-time reconfigurability of an ARCOM-based instrument, thus successfully solving the 'Wicked Problem' of ever-evolving systems. The ARCOM architecture and associated data structures such as the ARCOM packet format are well documented and various tools support the user during setup and day-to-day operation. ARCOM is discussed in-depth in (Grill 2007). Since its inception, ARCOM has also been deployed for the Advanced Imaging Riometer for Ionospheric Studies (AIRIS). Other instruments have also benefited from ARCOM concepts, for instance the new high-speed photometer for optical emission measurements (SPARKLE) developed by the author, which employs a packet-based streaming data format very similar to the one used by ARCOM. The Center for Geospace Studies at SRI International has developed a distributed data processing and transport architecture called 'Data Transport Network' (Valentic 2002). This is used in many large-scale projects, such as the Advanced Modular Incoherent Scatter Radar (AMISR) at Poker Flat, Alaska (Heinselman and Nicolls 2008). Future efforts will be undertaken towards combining Data Transport Network's excellent performance in distributed systems and over unreliable links with ARCOM's real-time streaming capability required for 'instant-feedback' type applications such as interactive telescope control.

References

Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E (2002) Wireless sensor networks: a survey. Comput Netw 38(4):393–422. doi:10.1016/S1389-1286(01)00302-4

Bray T, Paoli J, Sperberg-McQueen CM, Maier E, Yergeau F, Cowan J (2006) Extensible markup language (XML) 1.1. Technical report, The World Wide Web Consortium (W3C). Available from http://www.w3.org/TR/xml11/

Christiansen WN, Mathewson DS (1958) Scanning the sun with a highly directional array. Proceedings of the Institution of Radio Engineers, 46:127ff

Deacon J (2002) Design patterns. Technical report, Matrice, London, UK

Detrick DL, Rosenberg TJ (1990) A phased-array radiowave imager for studies of cosmic noise absorption. Radio Science, 25(4):325–338

ETSI (1997) Digital video broadcasting (DVB); a guideline for the use of DVB specifications and standards. TR 101 200. Technical report, European Telecommunications Standards Institute. Available from http://www.etsi.fr

Gamma E, Helm R, Johnson R, Vissides J (1995) Design patterns. Addison-Wesley, Reading

Gehrke J, Madden S (2004) Query processing in sensor networks. IEEE Pervasive Computing 1:46. doi:10.1109/MPRV.2004.1269131

Grill M (2007) Technological Advances in Imaging Riometry. PhD thesis, University of Lancaster

Hagfors T, Grill M, Honary F (2003) Performance comparison of cross correlation and filled aperture imaging riometers. Radio Science, 38(6):17/1–17/5. doi:10.1029/2003RS002958.

Heinselman CJ, Nicolls MJ (2008) A Bayesian approach to electric field and E-region neutral wind estimation with the Poker Flat Advanced Modular Incoherent Scatter Radar. Radio Science

Little CG, Leinbach H (1959) The riometer—a device for the continuous measurement of ionospheric absorption. Proc IRE 47:315–320

Mills BY (1952) The distribution of the discrete sources of cosmic radio radiation. Aust J Sci Res 5(2):266–287

Mills BY, Little AG (1953) A high-resolution aerial system of a new type. Aust J Phys 6:272–278

Murayama Y, Mori H, Kainuma S, Ishi M, Nishimuta I, Igarashi K, Yamagishi H, Nishino M (1997) Development of a high-resolution imaging riometer for the middle and upper atmosphere observation program at Poker Flat, Alaska. J Atmos Sol Terr Phys 59(8):925–937

Nielsen E (2001) Antenna system for a high resolution imaging riometer. Technical Report MPAE-W-03-01-04, Max-Planck-Institut für Aeronomie, Lindau, Germany

Nielsen E, Hagfors T (1997) Plans for a new rio-imager experiment in Northern Scandinavia. J Atmos Sol Terr Phys 59(8):939–949

Nielsen E, Honary F, Grill M (2004) Time resolution of cosmic noise observations with a correlation experiment. Annales Geophysicae—Atmospheres, Hydrospheres and Space Sciences 22(5):1687–1689, SRef- ID:1432- 0576/ag/2004-22-1687

Sommerville I (2004) Software engineering, 7th edition. Addison Wesley

Strohbach M, Gellersen H-W, Kortuem G, Kray C (2004) Cooperative artefacts: assessing real world situations with embedded technology. In Ubicomp 2004: Ubiquitous Computing, Springer

Valentic T (2002) A novel approach to data retrieval and instrumentation control at remote field sites using Python and Network News, February 2002. Available from http://transport.sri.com/TransportDevel/Sections/Documentation/Python10