

Evolution of the Commercial *ANTELOPE* Software

1. Background

At the end of the IRIS Joint Seismic Program (JSP), a decision was made to develop a new "Broadband Array" facility within the PASSCAL program using experience, equipment and technologies that were acquired during the life of the JSP. The central objectives of the new facility were to provide real time high resolution digital data from the field recorders directly to the desktops of the PIs or other interested parties and to provide the final SEED data volumes to the IRIS DMC as quickly as possible (within several days). It was hoped that the new Broadband Array facility could act as a prototype for future PASSCAL field deployments and start to give the research community experience with the new communication and data processing technologies that would become available with the new facility.

The specific software to support the Broadband Array, its required functions and its form as delivered to IRIS came about in an evolutionary manner. This software evolved from the Datascope Seismic Application Package (DSAP), a public-domain seismic data management/processing suite that was developed at the University of Colorado, Boulder with funding from the IRIS JSP. The main shortcoming of DSAP was also the main objective of the Broadband Array; real time data flow from the field sensors to the scientist. Through a series of meetings, system specifications, system design, prototype implementations and system evaluations, a new software facility, known as the Object Ring Buffer (ORB), was ultimately developed at the University of Colorado, Boulder to sup-

port the initial testing and operational phases of the PASSCAL Broadband Array. The first ORB software was tested with live data starting in the summer of 1996 from the initial test deployment of the Broadband Array equipment in Southern California. The first operational use of the ORB software started in the summer of 1997 after the deployment of the Broadband Array in Northwestern Colorado for the Lodore Array Project.

In the fall of 1996, Boulder Real Time Technologies, Inc. (BRTT) was formed as a Colorado corporation. BRTT's commercial mission was to provide technical expertise and computer software to support the operations of seismological networks worldwide. Shortly after its formation, BRTT entered into a contract with Kinemetrics, Inc. to support Kinemetrics' Saudi Arabia National Seismic Network project.

1. Early Development of the ARTS Software

At the start of the Kinemetrics/Saudi Arabia project BRTT conducted a review of the available software products that could be used in support of the Saudi Arabia Seismic Network. BRTT was interested in using the then public-domain Datascope/ORB software. However, the public-domain Datascope/ORB software was lacking many of the functions that were necessary for support of real time automated seismic network processing, including automated detection, phase picking, network triggering, association/location and magnitude estimation. Also lacking were ORB software modules that could communicate with Quanterra dataloggers, which were being used for the project. Another primary concern of BRTT was that the public-domain software had been developed to support a research mission and the BRTT staff knew that cost/reliability design decisions had been made based upon research mission objectives.

Because of the appealing design concepts embodied in the public-domain Datascope/ORB software, BRTT decided to initially adopt its

underlying data management and data flow concepts and to concentrate on building up the new client applications that were necessary to meet the requirements for automated real time seismic network processing. This collection of commercially developed client programs, including Quatterra acquisition/monitoring/control programs, detection/picking software, network triggering software, association/location software and magnitude estimation software, became the beginning of the Antelope Real Time System (ARTS), although in the early stages they were sitting on top of the public-domain Datascope and ORB software.

1. Mission Critical vs. Research Requirements

The software requirements imposed by the IRIS Broadband Array project were all derived from its basic research mission. Other than RAMP, there were no typical research projects that would require real time data flow to meet the basic research objectives. The real time requirement for the Broadband Array project was made instead to increase the efficiency and quality of the data flow into the permanent archive, relative to the efficiency and quality of data flow obtained from traditional portable-recorder type PASSCAL experiments. The operational environment of a Broadband Array experiment was traditional for an academic/experiment enterprise; 1) relatively short term (about one year), 2) many people to "babysit" the experiment/data flow, 3) not a lot of resources to invest in upfront development work, 4) emphasis on high rate of data recovery, 5) manual intervention ok during the experiment to keep the system running, 6) large data latencies ok as long as data was not lost.

The software requirements imposed by most permanent seismic networks are derived from several different basic missions. As with the Broadband Array, there is usually a research mission. However, unlike the Broadband Array, most seismic networks have critical "societal"

missions to reliably, accurately and quickly report information relating to damaging or felt earthquakes. These societal missions lead to "mission critical" software requirements that are in sharp contrast to those for the research mission. A typical seismic network operation 1) is long term, 2) employs a small permanent staff to oversee the operation, 3) is willing to invest upfront resources for development in order to, 4) achieve high data recovery, 5) minimize manual intervention to keep the system running, and 6) minimize data latencies.

1. Evaluation of the Public-Domain Datascope/ORB Software

The public-domain Datascope/ORB software was designed with the IRIS Broadband Array research mission in mind. As the software was tested with live data from the test deployment in Southern California, a number of problems were encountered. Each of these problems was either solved by changes in the public-domain software or operational "workarounds" were devised. By the time the Broadband Array started its first operational deployment in Northwestern Colorado, we felt confident that the public-domain Datascope/ORB software, along with a certain amount of vigilance by the people involved in the project, would meet the basic research mission.

During the winter of 1997 BRTT made a critical evaluation of the public-domain Datascope/ORB software as a candidate system for mission critical seismic network applications. In this evaluation BRTT was not looking at the concepts embodied in the Datascope/ORB system (BRTT was already convinced that the concepts were sound), but instead was looking at the specific public-domain implementation in order to identify weaknesses that would be unacceptable in a mission critical environment. This evaluation identified the following weaknesses.

1. **ORB server-client hangs.** Occasionally the TCP/IP socket between an ORB server and client would hang. Usually this was due to some communication hardware problem. This caused the read

on the client side to block indefinitely. Attempts were made in the public-domain implementation to get around this (non-blocking reads with timeouts in the application code), but all of these patches were ultimately unreliable and these hangs were never completely eliminated. We concluded that a fundamental new low level software utility needed to be developed and used throughout the ORB server-client communications to implement an automatic timeout/save state/reconnect/set state/continue mechanism. The workaround for the Broadband Array deployment was to monitor the connections and to restart the ORBs as necessary whenever they got hung. We found that this happened infrequently; about once a week on average.

2. **The ORB srcname table was static.** This is the table that identifies the list of packet source names within an ORB. This table was generated one time at **orbserver** startup. This could cause numerous problems whenever new packet source names appeared on an ORB and when an ORB was created for the first time. In these cases an ORB select would not see or return the new packets. This could in turn cause the **orbserver** to start chewing up CPU cycles (see item 3). We concluded that all ORB source name tables and indices had to be completely dynamic. Changing the public-domain implementation to work in this fashion would require substantial rewriting of the **orbserver**. The workaround for the Broadband Array deployment was to carefully startup the various ORB systems and their clients in a certain order so that clients who expected to read certain sources were not started until those sources appeared in the subject ORBs. This problem was not so critical in the Broadband Array deployment since the data flows were simple. However, in a fully functional seismic network, with all of the automated network processing enabled, this problem would be critical.

3. **The ORB select mechanism could cause serious performance problems for certain situations.** This was manifested by the **orbserver** using up huge amounts of CPU time. We identified the coding problems and decided that the ORB select mechanism needed to be rewritten. See item 2 for the workaround. We also decided that there needed to be a complimentary ORB reject mechanism, i.e. give me everything but ...
4. **The ORB "after" mechanism could cause serious performance problems for certain situations.** This was manifested by the **orbserver** using up huge amounts of CPU time. We identified the coding problems and decided that the ORB "after" mechanism needed to be rewritten. The ORB "after" function was not routinely used in the operation of the Broadband Array deployment. However, this function would be used routinely by the network processing clients in a seismic network processing environment.
5. **The ORB communication protocol was inefficient (especially for small packets) and prone to errors in inter-architecture situations (e.g. SUNS to PCs).** We decided that a new and more efficient protocol needed to be used along with a low level object serialization formalism to reduce protocol overhead and make inter-architecture connections transparent. This problem did not manifest itself with the Broadband Array deployment since the packet sizes were all relatively large (due to the REFTEK/UCSD data sources) and non-SUN architectures were not used.
6. **orbserver memory usage was increasing with time.** This implied **orbserver** memory leaks. The leaks seemed to be slow leaks. The workaround for the Broadband Array deployment was to occasionally restart the **orbserver**. We found any memory leaks to be unacceptable in a long term/minimum maintenance mission critical environment.

7. **We suspected a number of unidentified bugs within the orb-server and/or the liborb and libpkt used by all ORB clients.** This was manifested by mysterious crashes and system hangs. These seemed to be exacerbated by large ORBs with many clients and many sources. The workaround for the Broadband Array deployment was to use an executive script that would automatically restart programs when they died along with sufficient vigilance by the project personnel.

8. **Datascope database tables could only be written/updated by a single user/program at a time.** In the Broadband Array deployment, the only database table that was being updated continuously was the *wfdisc* table, that references the waveform files on disk, and this table was being updated by a single program. Therefore, in the Broadband Array deployment, we did not encounter problems related to this. However, for seismic networks with a full suite of processing programs running continuously, we knew that the output relational database would be updated by many different programs and analysts in a dynamic fashion. We concluded that we needed to embed some kind of automatic table locking mechanism within Datascope.

In spite of these problems/weaknesses, the public-domain Datascope/ORB software worked very well in support of the first Broadband Array deployment in Northwestern Colorado. The people involved in the project were vigilant to spot problems and respond quickly and they found the problems related to the software to be the easiest and quickest to fix (as opposed to hardware problems in the field). It was deemed that this software implementation was acceptable for running the Broadband Array and met the basic requirements of the project. In addition to support-

ing the Broadband Array project, the public-domain Datascope/ORB software has also been successfully used by the Alaska Network, at the Geophysical Institute, University of Alaska, Fairbanks, and by the University of California, San Diego/California Institute of Technology to facilitate automatic real time data sharing between the Anza Network and the CalTech/USGS Southern California Network.

1. Development of the Commercial Antelope Software

Based upon its evaluation, BRTT was uncomfortable with using the public-domain Datascope/ORB implementation, or a slightly modified version, for a local/regional permanent seismic network with mission critical operational responsibilities. BRTT's opinion regarding environmental monitoring software for mission critical applications was that the software should work reliably and accurately with the need for little or no human monitoring/intervention over very long periods of time. Certainly the basic data acquisition and archiving should work flawlessly regardless of the circumstances.

Consequently, in the winter of 1997 BRTT decided to start from scratch and, in a deliberate manner, from the bottom up, address all of the problems/weaknesses that were uncovered in the evaluation of the public-domain implementation. It was determined that the most efficient way to do this would be a complete rewrite of all of the ORB software. For the Datascope software, the table locking was made as a modification to the public-domain implementation. In order to do this right, BRTT developed a formal ORB test suite that was used to flog the system as it was developed. BRTT also started taking live data feeds from several networks that were using the public-domain Datascope/ORB software. This uncovered bugs and other system performance problems that were sorted out as the new ORB was developed. The basic concepts and the software interfaces in the public-domain Datascope/ORB were, for the most part, preserved in the commercial version.

By late summer 1997 a working prototype of the new commercial ARTS system was being tested. This new system included completely rewritten ORB software, revised Datascope software and a suite of new ORB and Datascope client-application programs to support automated real time seismic network processing. At about the same time BRTT met with IRIS and proposed to use the commercial Antelope software to run the Broadband Array deployment in Northwestern Colorado. In fall of 1997 BRTT and IRIS entered into a license agreement which provided IRIS limited use of the commercial Antelope software. The IRIS Broadband Array deployment in Northwestern Colorado started using the commercial Antelope software in the fall of 1997. Up until this time there had been two development tracks for the Datascope/ORB software; the public-domain track and the commercial track. After fall of 1997, development of the public-domain version of the Datascope/ORB software ceased (with the exception of several PASSCAL related programs for dealing with SEED data). The final version of the public-domain Datascope/ORB software was delivered to IRIS in June 1998 and is available, including all source code, on IRIS' anonymous ftp site at <ftp.iris.edu>.

The commercial ARTS software has been under continuous development since fall 1997. In summer 1998 the IRIS PASSCAL program contracted with BRTT to continue providing the commercial Antelope software for support of all PASSCAL experiments, including the experiments using the Broadband Array facilities.

1. Differences Between the Commercial Antelope Software and the Public-Domain Datascope/ORB Software

The commercial Antelope software is substantially different from the public-domain Datascope/ORB software that is available from IRIS' anonymous ftp site. The Antelope **orbserver** and the standard ORB client interface, **liborb**, have been completely rewritten. There are many new Antelope ORB client programs that do not exist in the public-

domain release. The ARTS executive program, **rtexec**, is also a complete rewrite of the public-domain version and is much more robust and versatile. New tcl/tk ORB interfaces have been written for Antelope that do not exist in the public-domain release. Substantial changes, improvements and new modules can also be found in the Antelope version of Datascope and the Datascope applications. These include automatic table locking, a dynamic event map display module, substantial enhancements of **dbloc2** (interactive location/review program), a new program for computing local magnitude, **dbml**, substantial rewriting of **dbe** (database editor GUI), new Datascope perl interfaces, and several new programs for automatically performing database backup/cleaning/migration tasks. Following is a breakdown of the differences between the current commercial Antelope software and the public-domain Datascope/ORB software.

orbserver differences:

1. **Dynamic srcname index table.** The internal ORB index table that associates source names with packets was completely redesigned so that it is dynamic and changes as new packets are written. In addition to new source names appearing in the table dynamically, stale source names are removed dynamically.
2. **Dynamic srcname based selections.** The internal ORB mechanism for performing a source name based packet selection for ORB client sockets was completely rewritten. The new implementation supports dynamic changes in the **srcname** index table and performs selections more efficiently than the public-domain implementation. This eliminates the problem of having to wait to start an ORB client program until the desired sources are available on the ORB and the **orbserver** CPU usage problem that was associated with certain types of selections.

3. **New ORB "after" implementation.** Time based requests for packets (give me the first packet with a time stamp after xxx) were part of the original ORB API (application programmer interface). The original implementation was inefficient and caused the **orbserver** to use large amounts of CPU time in certain situations. A new index table and lookup mechanism was implemented within the **orbserver** to specifically support time based packet access methods. This has resulted in significant improvements in performance for time based packet requests.
4. **No observable memory leaks.** A large effort was made to eliminate all memory leaks from the **orbserver**. Currently, we know of no existing memory leaks that are observable or detectable with the CASE tools we have used to analyze the software.
5. **No unexpected orbserver hangs or crashes.** We have taken a bulldog mentality about **orbserver** crashes or hangs. In the early stages of the commercial **orbserver** development, we promptly tracked down and fixed all problems that caused the **orbserver** to crash or hang. We have not seen such **orbserver** problems in the commercial implementation since winter 1998. BRTT has a zero tolerance policy toward **orbserver** crashes or hangs and will vigorously pursue and fix any reported occurrences.

Differences in the **orbserver**-client communication software:

1. **Implementation of a dynamic and reliable TCP/IP socket re-connection protocol.** After extensive testing to identify the nature of TCP/IP socket communications and how they fail, BRTT designed and implemented a smart TCP/IP socket re-connection protocol. This is embedded within a low level software library that is used by the **orbserver** and all ORB client programs. The protocol is transparent to the user and the application programmer. The protocol insures reliable and complete TCP/IP based

communications between the **orbserver** and a client application as long as there is sufficient buffer space within the ORB stores to cover physical down time of the communication channel. The protocol is implemented as a timeout / save state / close socket / look for reconnect / reconnect and open new socket / set state / continue sequence of events. This new protocol has completely eliminated **orbserver** and ORB client program hangs due to communications interruptions and has provided completely reliable inter-ORB data streams.

2. **Implementation of a protocol compiler with embedded low level object serialization.** This has provided a generic method for describing byte-stream protocols and automatically generating standard source code that implements reliable and efficient object serialization/de-serialization. The protocol compiler generated methods recognize local machine architecture and automatically produces data objects in the correct byte order. This has produced more efficient packing of the surrounding ORB protocol information around the raw data packets, resulting in reduced ORB to ORB byte flow rates especially for data streams with small raw packet sizes. This also has made it easier to pass ORB packet data across differing architectures, such as SUN to INTEL.

New ORB client programs:

1. **rtexec** - New ARTS executive program for controlling all of the other ARTS programs. There was a public-domain predecessor to **rtexec**, but the new program is a complete re-write of the public-domain version and it has significant improvements in reliability and functionality.
2. **rtm** - X-windows GUI front end to **rtexec**.

3. **orb2orb** - Replaces the public-domain **orbcp** program for making ORB to ORB copies.
4. **orb2pf** - Dump out parameter file objects from an ORB.
5. **orb2stream** - Dump out raw ORB packets (for demo/simulation/debugging replay).
6. **orbcapture** - Grab packet series from **orbserver** triggered by new database row.
7. **orblatency** - Accumulate latency statistics from a network.
8. **orbmsg** - Put a message onto an ORB.
9. **orbpftrigger** - ORB parameter object trigger of a program execution.
10. **orb2db, orb2dbt** - **orb2db** is a rewrite of the public-domain **orb2db**. **orb2dbt** implements only database row archiving (as opposed to waveform archiving).
11. **orbassoc** - Spatial grid search based real time earthquake associator/locator.
12. **orbdetect** - Multiple frequency STA/LTA real time detector plus phase picker.
13. **orbmag** - Real time local magnitude computation.
14. **orbtrigger** - Real time network trigger.
15. **cs2orb** - Quanterra COMSERV to ORB.
16. **qt2orb** - Quanterra datalogger to ORB.

17. **qtdownload, qtmassrecenter, qtping, qtset, qtshell, qtupload** - Programs for remotely commanding a Quanterra datalogger through ORB interfaces including 1) download a file from a Quanterra datalogger, 2) issue a mass recenter, 3) ping the datalogger and **qt2orb**, 4) set **qt2orb** parameters, 5) cause execution of a shell script on a Quanterra datalogger, 6) upload a file to a Quanterra datalogger.

18. **qtmon** - X-window GUI front end for **qt2orb**.

New ORB/system interfaces:

1. Tcl ORB interface implemented as a dynamic run-time loadable library (tcl v. 8.0).
2. Tcl interface to system performance functions (per process CPU, memory, etc.) implemented as a dynamic run-time loadable library (tcl v. 8.0).

Changes to Datascope database management software:

1. Implementation of a new automatic table locking mechanism. Multiple processes/users can now safely update a database at the same time.
2. Significant expansion of the Datascope perl interfaces.

New Datascope application programs in support of Antelope real time operations:

1. **qedd** - Daemon process for bringing in QED events into a Datascope external catalog database. This works with any of the normal "finger quake" event data sets.

2. **dbassoc_rt** - Real time association of events in a database from an external catalog.
3. **dbevents** - X-window GUI for making map displays of events from a dynamically changing database.
4. **rtdbclean** - Real time automated database and disk cleaner.
5. **rtbackup** - Real time automated tape archive program for database and waveforms.

New and upgraded Datascope programs in support of off-line processing (analyst review):

1. **dbloc2** - Significant enhancements have been made to **dbloc2** including, phase synchronization between **dbloc2** and **dbpick**, automatic regrouping, automatic detection of "dirty" associations (associations for which the arrival parameters have been modified since the event was located/associated), automatic recalculation of magnitude after an event has been relocated, enhanced quality control, review status marking of events.
2. **dbml** - New program to compute local magnitude from database input data.

1. Current Status of the Commercial Antelope Software

The commercial Antelope software has been under continuous testing for the last eighteen months involving live data feeds from hundreds of remote stations and dozens of ORB servers all over the world. Although

the commercial Antelope software is substantially different in its inner workings from the public-domain Datascope/ORB, to the outside user and application programmer, the commercial software does not appear to be very different from the public-domain software, with the exception of the added new functionality. BRTT purposefully designed it that way. The big differences come in reliability, functionality and performance. The commercial Antelope Real Time System never hangs, never dies mysteriously and clearly performs better than the public-domain version.

So far, we have seen no failures with the commercial ORB system for at least the last twelve months on any of the networks that have been using the software. We can run orbserver-client connections for indefinite time periods with no hangs (but many automatic and seamless reconnects). Designing a system like this, one in which large quantities of data are continuously transferred and processes can run safely for months at a time, so that all memory and pointer bugs have been eliminated is a tedious process that requires many months of continuous testing. We think that we are there with our commercial Antelope product and we hope that seismic network operators around the world will take advantage of this new product.