A proposal for xNTD computing

T.J. Cornwell, ATNF

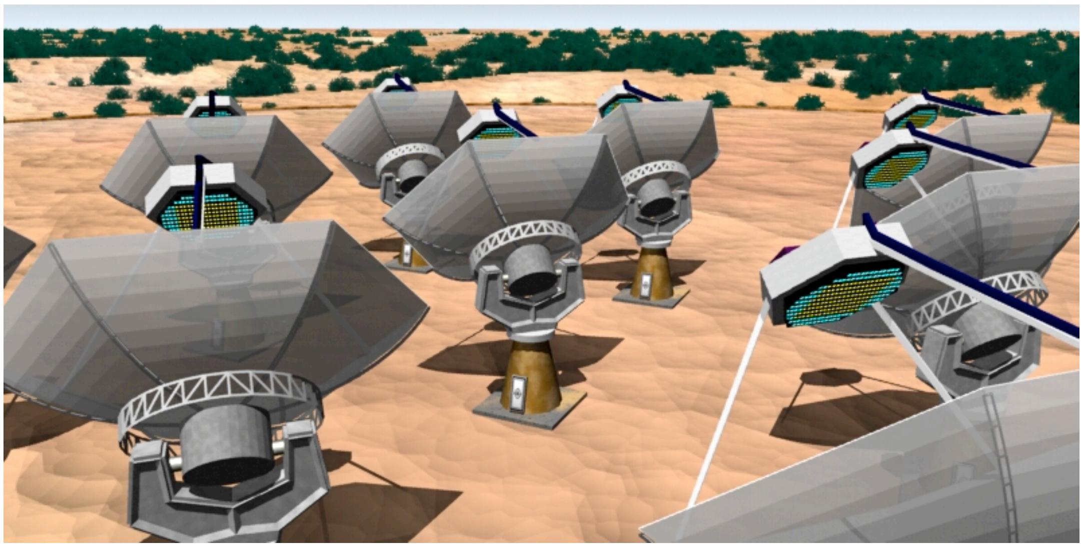
Tim.Cornwell@csiro.au

10/2/06

Abstract: *I identify the key challenges in xNTD computing as the large scale in data and processing, the use of focal plane arrays in radio synthesis, and the need for streamlined scientific operations. System complexity must be limited if the software costs are to be constrained. I propose an approach to xNTD computing designed to provide robust, well-documented scientific capabilities with a limited software budget. In this approach, the complexity is constrained at the top level by only supporting a limited number of observing modes. These modes are implemented by "Software Instruments". As we gain experience, we can deliver more Software Instruments.*

1. xNTD Goals and challenges

The xNTD has two main purposes – to act as a demonstrator for the Square Kilometer Array and to do great science. The xNTD will be a novel type of radio telescope – consisting of about 30 – 50 conventional synthesis arrays operating in parallel.



Focal plane arrays on each of twenty antennas will deliver multiple parallel streams of digitized signals that must be transported, correlated, calibrated, and transformed into one image of a region of the sky. Since the telescope is parallel, the peak data rate is high (a few TB per hour – see Cornwell, 2006) as is the necessary peak computing (about 20Tflops). The first computing challenge is therefore of **large scale in both data and processing**. Cornwell shows that parallel processing with thousands of processors will be necessary to handle full spectral line observations. Fortunately, much of this processing is independent per channel – embarrassingly parallel in the terminology of high performance computing – and can thus be spread across multiple processors with less effort than would be required for continuum processing.

Although radio synthesis is well understood and quite advanced, until now only single feeds have been used, and those have been well-behaved systems such as horns. xNTD will synthesize feeds by summing multiple elements in a focal plane array. We currently know little about the performance of such systems in an astronomical context. Novel calibration and imaging techniques will certainly be necessary. This constitutes the second challenge: **use of focal plane arrays in radio synthesis**. Experience with the NTD will give insight on what the problems are likely to be but we already know that calibration of the FPA is likely to be an important factor.

The xNTD will be operated as a national facility (rather than, for example, an experimental system), and although the system will be complex, it must allow effective scientific use. This is clearly the third challenge: **streamlined scientific operation**. There are a number of models for how to do this but most require substantial software resources. For example, ALMA is spending about 430 FTE-years to make a telescope that can be used easily by radio astronomy neophytes.

Thus, to summarize, the key challenges that must be overcome are:

- Large scale in data and processing
- Use of focal plane arrays
- Streamlined scientific operation

These three challenges will also be present for the SKA as well. Any proposed approach for xNTD or SKA computing must therefore be capable of meeting these challenges.

2. Science requirements

Johnston (2005) has summarized the xNTD science requirements, drawing on a workshop held at the ATNF in April 2005. Here we distill only the core performance requirements. We also show the key parameters of the xNTD, along with the corresponding values for the KAT and the SKA.

Table 1 Summary of possible xNTD science

Topic	***Requirements***
Extragalactic HI emission surveys	300 day all sky survey. Ultra deep, 100 day integration in one direction.
Extragalactic HI absorption surveys	Large scale survey, tens of days, to 0.01 optical depth
Survey for OH masers and mega masers	100 day survey over 1000 square degrees
Continuum surveys	NVSS equivalent every day. 1 day gives 400uJy noise (confusion limit), can see polarization variability deeper. Variability at 2% for 100mJy sources, 20% for 10mJy, daily.
Galactic HI surveys	100 day survey, 600 square degrees at 1K, 40 arcsec. Deep imaging of mid-latitudes for HVCs: 100mK at 3 arcmin
Pulsar surveys	120 day all sky survey, adding all collecting area, pixelizing the primary beam
Polarization and Cosmic Magnetism	All sky survey to 1% across the field. Faraday tomography: slices of 100MHz across entire band
VLBI	Wide field of view, ionospheric calibration

Table 2 xNTD, KAT, and SKA telescope parameters

	xNTD	***KAT***	***SKA***
Frequency Range	0.8 – 1.7 GHz	0.7 – 1.7 GHz	0.1 – 30 GHz
Polarizations	2	2	2
Bandwidth	256MHz	250 MHz	25%
Frequency resolution	5kHz	4kHz	
Spectral Channels per beam	65536	65536	65536
Spectral Resolution	*NYS*	~2 km/s	*NYS*
A_{eff}/T_{sys}	3584/50 = 76	45	20000
Field of view	40 deg^2	40 deg^2	1 – 200 deg^2

3. Resources and Assets

Suppose that the computing budget of xNTD was to be \$10M? What would this buy? Splitting equally between hardware and software, we could afford computing hardware of \$5M – about right according to the preliminary analysis by Cornwell (2006). The other \$5M buys about 50 FTE-years of software effort. This is about one tenth that planned for ALMA, half of that for LOFAR, and close to that for KAT. Large analysis packages such as AIPS, IRAF, and AIPS++ typically consume hundreds of FTE-years to reach a level of maturity sufficient of supporting a range of science. Put another way, at our typical costs and levels of quality, 50 FTE-years buys about 300,000 to 400,000 lines of debugged, tested, documented code. Hence the conclusion has to be that reuse of existing software packages is essential. Examples of packages available to us internally are AIPS++, MIRIAD, ATOMS, SCHED, LiveData, the ATCA pipeline, ASAP, RVS.

We also have an option to collaborate closely with the KAT group to share computing costs. This seems very advisable given the limited resources available to both groups, the similarity of the telescopes, and the shared imperative to demonstrate international collaboration in the lead up to SKA. The KAT computing group expects to consume a similar number of FTE-years. The group is strong in non-astronomical software development, particularly so in commercial web-based solutions. This is complementary to the skill-base of ATNF, which is mainly in astronomy, telescope monitor and control, and the development of astronomical packages such as those listed above. In addition, there are areas in which neither group has significant existing expertise, such as high performance computing and very large data storage. Collaboration between the xNTD and KAT computing groups would double the nominal resources available, though with some increase in overhead due to the geographic and organizational splits between the two groups.

On the hardware side, an investment of $5M represents a substantial increment to Australia's total capacity in scientific computing. Since the xNTD computing throughput needs will ebb and flow with the science to be done, some form of cost sharing within CSIRO via HPSC or in the larger community via APAC may be possible.

4. Operational Model

In this section, we turn to a discussion of the scientific operations. Conventionally, radio synthesis arrays are operated in a free-form way. Observations are specified at a fine level by using a program such as SCHED to issue sequences of commands such as "observe this source at this frequency for a few minutes". A complete observation may consist of hundreds of such commands. The observer is normally responsible for the subsequent data processing in which the inevitable complexity arising from the fine specifications must be deal with. The input complexity of the telescope therefore translates directly into complexity for the observer and for the software developer.

The conventional architecture for a synthesis telescope is therefore quite flat (see Figure 1), requiring the observer to interact directly with the telescope, archive, and the reduction packages. Even in a sophisticated and complex telescope like ALMA, this layering is used.

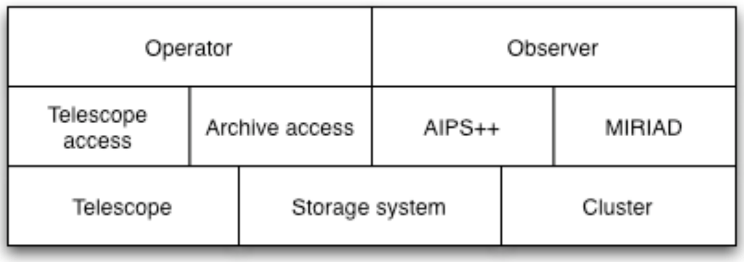


Figure 1 Conceptual layering of software in conventional design for a synthesis telescope.

The ALMA project has as a high priority the goal of making the telescope usable by radio astronomy neophytes. To do this, ALMA is taking the approach of developing a software system capable of tracking and dealing with the complexity at the second layer. The ALMA software will aid the observer in setting up the schedule, and then deal automatically with the necessary processing during observing and data processing. Whether this is a viable approach has yet to be demonstrated by the successful operation of ALMA in this mode. As noted above, the ALMA software budget exceeds that of xNTD by about an order of magnitude.

An alternative approach, similar to that used in the Parkes Multibeam system (Barnes, 1997), is to limit complexity at the front end. Only certain types of observing are to be supported and these are describable by a limited set of parameters. Ensuring that the processing engine can support the observing is then much easier. The KAT project has a similar concept in which the back end software is called a Software Instrument. I suggest we adopt the same terminology. A Software Instrument presents a simple interface aimed at supporting one type of observation. For example, one SI would support continuum polarimetric imaging and another would support HI surveys.

The proposed conceptual layering of xNTD is represented in Figure 2. In the proposed approach, the presentation layer and the software instruments are layered on top of the data reduction packages and thus mediate the complexity.

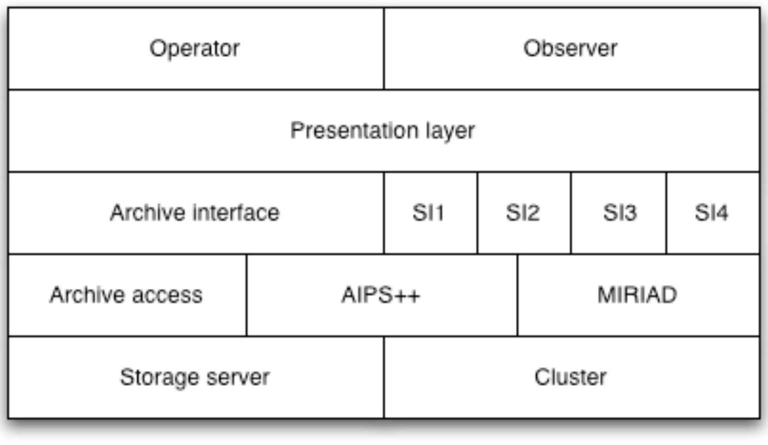


Figure 2 A possible conceptual layering showing multiple software instruments SI1, SI2, SI3, SI4

Under this model, operations proceed as follows. The observer interacts with a given software instrument. Using the SI, the telescope is placed into a well-determined observing mode, specified by a limited number of options. We can call this a use case. Data flow through the software instrument, which executes in the known computer hardware environment available, and the results are stored in the archive for access by the observer. Opportunities for interaction by the observer are limited – the existing observing can be aborted and another configuration invoked. As an example, consider a survey of a given region for HI. The observer should specify the field center, observing frequency, velocity resolution, and integration time. In addition, some defaults for calibration (amplitude, phase, bandpass) can be examined and overridden if necessary. Data are collected and flow through to the archive. A Software Instrument (SI) monitors the data accumulation and processes the data accordingly. The observing context and sequence of observations match the heuristics built into the SI so relatively little intelligence is needed. Furthermore, the SI is matched to the computing hardware available (probably a large cluster) and so computing times (such as that taken for a bandpass solution) are known. This makes the internal logic of the processing significantly easier to design.

As said earlier, this key to this approach is to limit complexity at the telescope front end. The scientific capabilities of the telescope are limited explicitly by the use cases supported. Consequently, we can ensure that for early supported use case, there is a corresponding SI available and that it is well tested and documented. Thus this approach also should increase quality by constraining the data pathways that must be tested prior to release of a new capability. As we gain more experience in operations, we can make the telescope support more use cases by the development, testing, and deployment of the corresponding software instruments.

The science topics listed in Table 1 translate to a limited number of use cases. Strategic decisions can be made to develop these use cases in a known order to match the overall science goals for the telescope. All the activities of the xNTD team can be focused on developing one use case at a time.

Different software instruments may be developed using whatever tools are most suitable. Both AIPS++ and MIRIAD are currently possibilities and others may come along. Decisions can be made at the time of implementation based on the needs of a given software instrument. Parallel processing will be needed to match the data processing load, especially for spectral line observations. As discussed by Cornwell (2006), this is probably embarrassingly parallel and could therefore be implemented at a high, data level, perhaps via a scripting layer layered on top of MIRIAD. Calibration and imaging of continuum observations is likely to be much more demanding algorithmically and computationally. Data level parallelization will not be sufficient and instead algorithmic parallelization either by MPI or perhaps OpenMP will be necessary. For this AIPS++ may be more suitable.

5. Implications

The disadvantages of this proposed scheme are:

- Scientific capabilities are limited to those explicitly supported by an SI.

The advantages are:

- The telescope will be easier to use than with a conventional approach.
- The optimum observing and data reduction strategies can be encapsulated in software instead of cookbooks and informal knowledge.
- Scientific capabilities are released when ready, in a robust, well documented, computationally optimized scientific instrument.
- Versioning of scientific instruments simplifies support.
- New capabilities can be added incrementally, with known resource requirements. Thus the development of an SI could be the subject of an ATNF Project, managed by the usual CSIRO project management apparatus.
- Software costs will be better known, more constrained, and lower than for a conventional approach.

Finally, the software instruments need not be the only means for getting access to the telescope. For groups with the requisite software skills, we can provide open interfaces to various parts of the telescope. This allows the addition of new capabilities to the telescope without requiring the aid of the core team. Google Earth provides a nice example of the ingenious uses people can make of open interfaces. Examples of possible open interfaces in xNTD are:

- TCP interface to the beamformer and correlator data streams (as used by ATA)
- Software instrument - presentation layer
- Archive access

Opening the interfaces in this way will allow other groups *with the requisite software* *expertise* to access the telescope at a deeper level than allowed by the software instruments.

Acknowledgements

Many of the ideas presented here arose from discussions with a wide range of people. I'd particularly like to acknowledge Jasper Horrell and Simon Ratcliffe of the KAT Computing Group.

References

Barnes, D.G., 1998, *"Realtime, Object-oriented Reduction of Parkes Multibeam Data using AIPS++"*, ADASS VII, ASP Conference series vol 145. See http://www.adass.org/adass/proceedings/adass97/barnesd1.html.