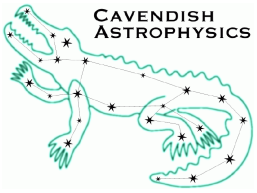
# MROI Dataflow architecture

## MROI System Data Flow Discussion Document

**INT-409-ENG-0105**

**The Cambridge Delay Line Team**

rev 1.0
16 July 2009




Cavendish Laboratory
Madingley Road
Cambridge CB3 0HE
UK

## Change Record

| Revision | Date | Authors | Changes |
|---|---|---|---|
| 0.1 | 2009-07-09 | EBS | First draft. |
| 0.2 | 2009-07-13 | EBS DFB | Added discussion, updated illustrations, added DFB's data model. |
| 0.3 | 2009-07-14 | EBS DFB CAH | Modified main text according to CAH's suggestions, replaced appendix with DFB's new version. |
| 0.4 | 2009-07-14 | CAH | Checked EBS corrections and adjusted text. Mods to formatting and title. |
| 1.0 alpha 1 | 2009-07-15 | EBS | Changes to fringe instrument data rates, miscellaneous edits suggested by CAH, DFB, JSY. |
| 1.0 alpha 2 | 2009-07-15 | CAH | Adjustments to intro sections and subsystem descriptions. |
| 1.0 beta | 2009-07-16 | EBS CAH JSY DFB | Adjustments to preface, data rate calculations, system diagram, appendix. Correction of minor errors. |
| 1.0 | 2009-07-16 | CAH | First release. Correction of minor errors. |

## Objectives

- To summarise the data flow requirements and characteristics expected for the Magdalena Ridge Observatory Interferometer.
- To suggest possible implementations for the data flow architecture that might successfully be implemented independent of JPL's Realtime Control (RTC) framework.

## Scope

This document contains information on the data flow requirements of the various planned and anticipated subsystems at the MROI, and explains how these requirements might be met using a customised, lightweight framework for the MROI inter-system communication that would not rely upon the use of the JPL Realtime Control (RTC) framework.

The suggested implementation would be flexible, capable and relatively straightforward to implement. It is based on experience both with interferometric (COAST/NPOI/MROI) systems and adaptive optics implementations such as ELECTRA and NAOMI which share similar needs in terms of high volume data traffic and a mixture of quasi-real-time and hard real-time control and communication.

Some initial thoughts concerning a favored online and offline data model are also presented in the appendix.

This document does not contain information on data flow management within any of the proposed MROI subsystems, and so only provides suggestions as to how the inter-subsystem data flow requirements can be achieved. Intra-subsystem data flow issues will need to be dealt with in a

separate document, though the Cambridge team would recommend that standardisation in its implementation — in as much as this is possible and feasible — is highly desirable. Lessons learnt through the MROI delay line development programme might provide a useful starting point for that topic.

# Contents

# 1 Introduction

Many kinds of data will flow between subsystems at the MROI. This data includes subsystem telemetry, science data, logs, and commands. From a data flow perspective, the physical layout of the MROI implies that the control system will necessarily have the following properties:

- **It will be complex.** The MROI will contain many subsystems that serve different purposes. These subsystems must each be able to pass data to other subsystems, and they must be orchestrated as a whole by the supervisory (control) system for the interferometer to function correctly.
- **It will be diverse.** The different subsystems will vary greatly in terms of their data rate and latency requirements. Furthermore, as no single commonly available technology available today meets all of these requirements there must be a variety of physical methods of transferring data.
- **It will be distributed.** The MROI subsystems will be physically spaced over an area several hundred metres in diameter.

From an operator's and operational point of view, the following properties of the control system will also be very desirable (if not required) for efficient commissioning and scientific operation:

- **Responsiveness:** When an operator commands the interferometer, it should react "right now" (that is, within of order 500 ms). Similarly, status information visible to the operator should display what the interferometer is doing "right now". Note that here the term "operator" is used in the most general sense, that is, it includes the possibility of an automated sequencer.
- **Flexibility:** The control system should be ready for future modifications and enhancements to the MROI, even unanticipated ones. If it is easy to use the control system in unusual ways or to make changes to it, the life of the instrument and the science payback will be enhanced at relatively low cost.

# 2 System Overview

A conceptual design overview of the interferometer supervisory subsystem can be found in [1].

It is our current understanding that certain decisions regarding data flow in the interferometer have already been made (either explicitly or implicitly):

- The physical layer (apart from a few specialised subsystem-to-subsystem connections) will be an ethernet network and data will be carried in TCP/IP packets. TCP/IP can ship data over interferometer-scale distances at high volume and is also very flexible in terms of making arbitrary data connections between subsystems without physical rewiring.

  TCP/IP makes no guarantees in terms of data delivery time (latency) and the actual time will depend on how much network traffic there is and the number of junctions between source and destination. However, on a lightly loaded MROI-sized network the typical latency is likely to be well under a millisecond.

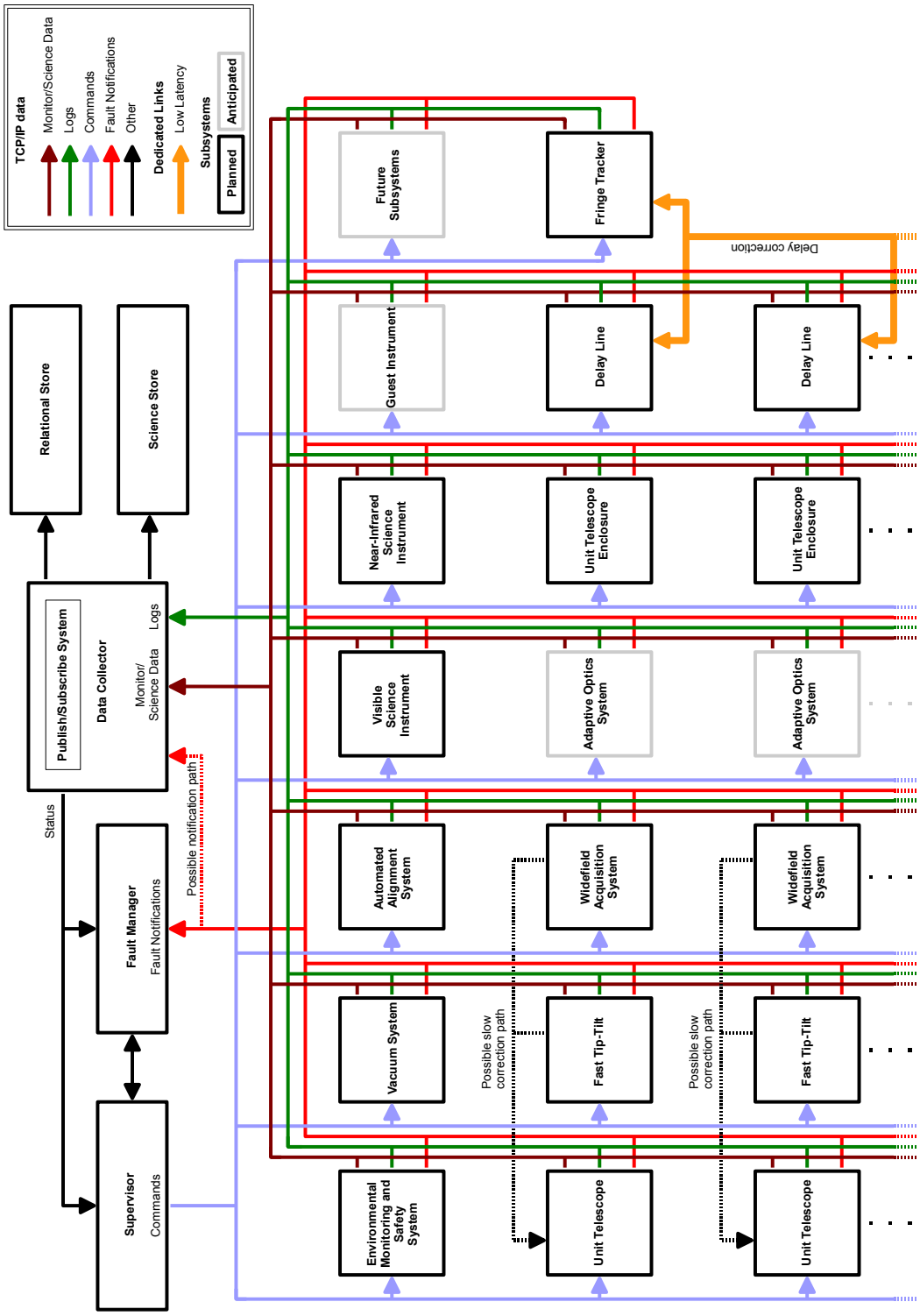



Figure 1: Overview of the MROI data flow requirements. The different colors for the lines and boxes indicate different functionalities and implicit requirements. The bottom row in the figure (which appears as the rightmost column on the page) would be replicated as any additional unit telescopes are added to the array.

- Each MROI subsystem will send monitor, log and (optionally) science data to a data collector. We understand that there will also be a fault channel. We are unsure whether the data collector will reside within the subsystem or whether all data collectors will be in a separate physical location. It is our understanding that any decision as to whether any faults are to be reported directly to a fault manager or whether they will go via a data collector has not yet been made.

  These data will eventually, however, end up in a science store or a relational store or both for near-real-time or long-term analysis.

- Status data will be extracted from the subsystem data and sent to a supervisor and fault manager. The supervisor will also be able to accept commands from an operator and from its own internal sequencer. It accordingly will send commands over the network to the various subsystems in order to make the interferometer execute whatever tasks are necessary to realise the user's desired outcome.

A chart that gives an overview of the MROI data flow requirements for a 2-element implementation can be found in Figure 1. Components in the top two rows of the figure represent components of the array that are essentially independent of the number of unit telescopes (UTs), whereas the bottom row would be replicated for each additional UT installed.

## 3 Subsystem Data Flow Requirements

The MROI will contain many subsystems, each with its own data flow requirements. In order to inform any discussion on possible implementations for the MROI data flow infrastructure, the following Subsections focus on the individual subsystems in the array and in each case provide a first-order quantitative assessment of the requirements for each subsystem. The impatient reader who just wishes to see the "bottom line" is refered to Table 1 (Subsection 3.15) which lists the data flow requirements for each subsystem in turn and then calculates a rough total.

Note that when a requirement is listed as "$< 100$ kbits/s" it means that the expected data transport method (likely to be 1 Gbit ethernet) is so rapid and capacious as to render the requirement negligible.

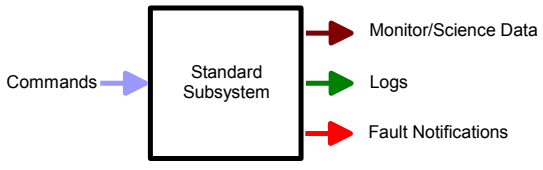



Figure 2: An MROI "standard subsystem". All data flows in and out are assumed to use the TCP/IP protocol over ethernet.

### 3.1 The "Standard Subsystem"

While subsystem requirements are diverse, there are some features common to all subsystems. Hence we define a "standard subsystem". Every subsystem in the MROI has as a minimum the properties of a standard subsystem, but might also have additional data flow requirements. These additional requirements are listed in the following sub-sections that deal with the subsystems concerned.

A "standard subsystem" is illustrated in Figure 2. It receives commands and outputs monitor, logging, fault and science data. Monitor data, also sometimes called "telemetry" in this document, contain the status of the subsystem and data from the sources within it. Some of these data are considered to be science data. Logs are occasional textual messages indicating status. Fault data indicate error conditions. It is anticipated that these data flows will use the TCP/IP protocol over ethernet, as there is no hard real-time requirement on the data latency — the data only has to arrive at its destination on a timescale that is short compared to human perception.

Internally, a standard subsystem may also contain components that use the interferometer ethernet infrastructure for internal data flows. However, in most cases the network can be arranged so that the control system never sees these data, for example by placing the components behind switched hubs. The main exception is the automatic alignment system, which is distributed throughout the interferometer and which could, in some design scenarios, pass significant video data through the system-wide ethernet.

### 3.2 The Environmental Monitoring and Safety System

The Environmental Monitoring and Safety System monitors the local environmental conditions. These data can be used to search for correlations between environmental changes and other system data, or warn the interferometer control system of impending bad weather so that it can park the telescopes and close the telescope enclosures. Data rates and latency demands are negligible by TCP/IP standards.

| | |
|---|---|
| Maximum sample rate: | <1 Hz |
| Maximum system data rate: | < 100 kbits/s |
| | |
| Unusual interfaces: | None. |

### 3.3 The Vacuum System

The Vacuum System comprises a set of gauges that monitor the pressure within those MROI components that can be evacuated and associated valves and pumps. The data volume and latency requirements are amply served by TCP/IP.

| | |
|---|---|
| Maximum sample rate: | 0.1 Hz |
| Maximum system data rate: | < 100 kbits/s |
| | |
| Unusual interfaces: | None |

### 3.4 The Automated Alignment System

The Automated Alignment System (AAS) is a distributed system with components found throughout the optical trains of the interferometer, including those on the fringe tracker beam combining table. The AAS is responsible for automatically aligning mirrors at the start of each night's observing using a set of actuated mirrors, semiconductor lasers and video cameras. The precise infrastructure to be deployed has yet to be finalised and the sequencing of the alignment procedure is complex, but the data rates and latency requirements are easily met by the network.

The required data volume will increase if video from the alignment cameras needs to be transmitted across the network, either for analysis during the alignment procedure or for operator viewing. A 640×480×8 bit video stream at 30 Hz generates 73 Mbits/s of raw data and there might be ten cameras operational at a time. Lossy video compression would reduce this to a tiny fraction of the rate but should not be used where analysis is required because it introduces artefacts into the video. Lossless video compression might be able to reduce the data rate by a factor of several times but would have to be tested on alignment video data. A compromise would be to reduce the framerate to, say, 2 Hz, and design the alignment system with this rate in mind. This results in a total data rate of $\sim 50$ Mbits/s for ten cameras.

It should be kept in mind that the alignment system exists to align the interferometer optics prior to observing and hence it does not usually run or consume network resources during observing. So during observing we state the alignment system rate as "< 100 kbits/s".

| | |
|---|---|
| Maximum sample rate: | 30 Hz (or 2 Hz for compromise system) |
| Maximum system data rate: | < 100 kbits/s for observing, 50 Mbits/s for setup, |
| Unusual interfaces: | None. |

### 3.5 Visible Science Instrument

This instrument will be responsible for collecting science data at wavelengths from roughly 600 nm to 1000 nm.

The maximum rate of data production will be limited by the maximum pixel readout rate of the Visible Science Instrument photon-counting CCD. The fastest suitable cameras approach a rate of 10 Mpixels/s so assuming 16 bits per pixel after digitisation we arrive at a rate of 160 Mbits/s per camera. In a fully commissioned instrument with ten telescopes a total of two cameras will be required. These data will need to be transmitted in a lossless fashion to a science archive for analysis.

This subsystem completely dominates data generation within the interferometer. However, there are no plans for a visible science instrument in Phase 1 of the interferometer operations plan. Hence the data flow could be managed simply by allowing space for the appropriate network infrastructure and deferring purchase of the required components until necessary, when they will cost considerably less.

| | |
|---|---|
| Maximum sample rate: | <1 kHz |
| Maximum system data rate: | 320 Mbits/s |
| Unusual interfaces: | None |

### 3.6 Near-Infrared Science Instrument

This instrument will be responsible for collecting science data at wavelengths of roughly 1150 nm to 2400 nm. Data flow planning for this instrument is still somewhat speculative, but the camera readout rate will be lower than that for the Visible Science Instrument. Conservatively assuming a camera with a readout rate of 1 MHz per quadrant (this is faster than anything on the market today) and 16 bits per pixel, we arrive at a maximum data rate of 64 Mbits/s per camera. In a fully commissioned instrument with ten telescopes, two cameras would be required.

| | |
|---|---|
| Maximum sample rate: | <1 kHz |
| Maximum system data rate: | 128 Mbits/s |
| Unusual interfaces: | None. |

### 3.7 The Unit Telescopes

The unit telescopes collect light from an observational target and direct it into the interferometer for combination and analysis. There can be up to ten unit telescopes and each one presents a subsystem interface to the control system.

| | |
|---|---|
| Maximum sample rate: | 20 Hz |
| Maximum system data rate: | < 100 kbits/s per telescope |
| Unusual interfaces: | The tip-tilt mirror in each telescope receives a tilt signal from its associated fast tip-tilt subsystem at ∼1 kHz. However, this is an analogue signal and so from a data flow point of view the tip-tilt mirror is really part of the fast tip-tilt subsystem.<br><br>Each telescope also receives a slow correction signal from a fast tip-tilt system to keep the observed source centred in the field of view. Our understanding is that this signal is to be sent via the supervisory system in the standard way, however if this introduces too much latency it could go via TCP/IP over the local network. |

### 3.8 The Fast Tip-Tilt System

There is one fast tip-tilt system per telescope and each one presents its own subsystem interface to the control system. Each tip-tilt system monitors light from the observational target for wavefront tilt introduced by the atmosphere. It then sends its associated telescope M2 mirror a signal to correct for this. Due to the rapidity with which atmospheric turbulence affects the tilt, a high sample rate is required. However, the transmission to the fast tip-tilt mirror is analogue so from a data flow point of view the mirror is part of the tip-tilt system.

Also to be considered is the camera video generated by the monitoring. The worst case is a readout of a 32×32 pixel area with a depth of 16 bits per pixel and a frame rate of roughly 800 Hz. These

data are used to calculate a correction for the tip-tilt mirror and yield 13 Mbits/s per telescope. However, not all of this need be sent over the network. If it is subsampled to yield a network frame rate of, say, 10 Hz then the network data rate becomes 160 kbits/s per telescope or 1.6 Mbits/s for all ten unit telescopes.

Finally, data generated at 800 Hz might be sent as telemetry over the network. The raw data rate for this is estimated at 360 kbits/s per telescope.

| | |
|---|---|
| Maximum sample rate: | $\sim$1 kHz |
| Maximum system data rate: | 520 kbits/s per telescope |
| Unusual interfaces: | Each fast tip-tilt subsystem sends an analogue tilt signal to a tip-tilt mirror in its associated telescope at $\sim$1 kHz.<br><br>Each fast tip-tilt system also sends its telescope a slow correction signal, in the worst case also estimated at 360 kbits/s, to keep the observed source centred in the field of view. Our understanding is that this signal is to be sent via the supervisory system, however if this introduces too much latency it could go via TCP/IP over the local network. If it consequently is routed via a local switched hub it will not be seen by the control system as a whole. |

### 3.9 The Widefield Acquisition System

There is one widefield acquisition system per telescope and each one presents its own subsystem interface to the control system. These cameras have a much wider field of view than the telescopes that they serve, and are used to locate sources of interest when the telescope pointing model is ill-defined. During source acquisition each sends a pointing correction signal to its associated telescope.

The necessary bandwidth increases if the camera video is to be sent over the network. In this case the discussion is similar to that for Subsection 3.4 except that the acquisition cameras will need to work while observing (though not when interferometric data are being collected). If they are each subsampled for the network at 10 Hz (or run at 10 Hz natively) then the transmitted data rate will be 2.5 Mbits/s per telescope.

During observing, however, the widefield acquisition system is not normally expected to be in use and the data rate in this context is "< 100 kbits/s".

| | |
|---|---|
| Maximum sample rate: | 10 Hz |
| Maximum system data rate: | < 100 kbits/s per telescope |
| Unusual interfaces: | When required, each system sends its telescope a pointing error signal during source acquisition using the same protocol that the supervisor uses to point the telescope. This signal might be sent via the local network rather than via the supervisor. A similar scheme works well at COAST. |

### 3.10 The Adaptive Optics System

It is possible that each telescope will eventually have an adaptive optics system installed, which would be used to correct higher-order wavefront errors at the telesacope. Any definition of the interface at this stage is purely speculative, however as the data requirements would be similar to those of the fast tip-tilt system it could well be implemented in a similar way.

### 3.11 The Unit Telescope Enclosures

The Unit Telescope Enclosures open and close according to the need to observe or to protect each telescope. They also rotate during observing. There is one enclosure subsystem per telescope. The data volume and latency requirements are small and easily met by TCP/IP.

| | |
|---|---|
| Maximum sample rate: | 10 Hz |
| Maximum system data rate: | < 100 kHz |
| | |
| Unusual interfaces: | None. |

### 3.12 The Delay Lines

The delay lines change the optical distance that light from the telescopes travels before it is combined. There can be up to ten delay lines at the MROI and they are expected to each present a standard subsystem interface to the interferometer control system although they are all physically managed by a single controller.

| | |
|---|---|
| Maximum sample rate: | 5 kHz |
| Maximum system data rate: | 1 Mbit/s per delay line. If the operator wants to view or log video from one of the built-in shear cameras, then the discussion in Subsection 3.4 again becomes relevant. If each video stream is subsampled at 10 Hz then 2.5 Mbits/s must be added to the above figure. However, shear camera video is a diagnostic tool and should not normally be needed during observing. |
| Unusual interfaces: | Every 2–5 ms, $\sim$5 bytes per delay line are to be received from the fringe tracker with a latency of no more than several hundred microseconds. There is also a low-volume, low latency flow in the opposite direction that indicates the current optical delays. Although this is drawn in Figure 1 as being connected to all the delay lines, in practice it is likely to be a single direct line to the metrology system that controls all of the delay line trolleys.<br><br>Our preferred option for this connection is a dedicated real-time point-to-point ethernet link. The metrology section of the production delay line subsystem will be written in Xenomai (a real-time Linux) which has a real-time ethernet layer called RT-net. Time-of-flight for packets with RT-net has been shown to be typically 100 $\mu$s with only a few microseconds of jitter[2].<br><br>Alternatively a dedicated reflective memory link could be used. The fringe tracker would simply write to the reflective memory and the delay lines would read that data from it as if it were local memory in both cases. Writing and reading are possible at any time, effectively decoupling the data rates of the two systems.<br><br>However the link is implemented, due to the latency constraints it will not be part of the interferometer-wide TCP/IP network. |

### 3.13 The Fringe Tracker

The fringe tracker measures the current errors in the optical delays introduced by the delay lines. It sends these measurements to the delay line with the aim of minimising these errors.

The fringe tracker in a fully commissioned instrument consists of four cameras, each transmitting 25 pixels of data digitised at 16 bits per pixel from an infrared detector array at 500 Hz. The resultant raw data rate is 200 kbits/s. Additional calculations will also be sent, such as group delays and visibilities, and these in the worst possible case might increase the total data rate to 1 Mbit/s.

| | |
|---|---|
| Maximum sample rate: | 500 Hz |
| Maximum system data rate: | 1 Mbit/s |
| Unusual interfaces: | The fringe tracker is at the other end of the "unusual link" that the metrology systems use. For more information, please refer to Subsection 3.12 above. |

### 3.14 Guest Instrument, Future Subsystems

Provision should be made to handle the data flow requirements of a guest instrument. Allowance should also be made for the possible needs of future subsystems that have not even been thought of at this stage.

### 3.15 Summary Table

The above data can be summarised in Table 1. Note that the data rates for any adaptive optics, guest or future subsystems would be purely speculative and so are not included here. However, we also note that in general, the use of a guest instrument will replace one of the existing ones described above. i.e. the total number of instruments (excluding the fringe tracker) will only ever be two.

| | | Phase 1 (6-tel) | | Phase 2 (10-tel) | |
|---|---|---|---|---|---|
| Subsystem | Rate/ [Mbits/s] | Qty | Total/ [Mbits/s] | Qty | Total/ [Mbits/s] |
| Environmental Monitoring & Safety | < 0.1 | 1 | 0.1 | 1 | 0.1 |
| Vacuum | < 0.1 | 1 | 0.1 | 1 | 0.1 |
| Automated Alignment | < 0.1 | 1 | 0.1 | 1 | 0.1 |
| Visible Science Instrument | 160 | 0 | 0 | 2 | 320 |
| Near Infrared Science Instrument | 64 | 1 | 64 | 2 | 128 |
| Unit Telescope | < 0.1 | 6 | 0.6 | 10 | 1 |
| Fast Tip-Tilt | 0.52 | 6 | 3.12 | 10 | 5.2 |
| Widefield Acquisition | < 0.1 | 6 | 0.6 | 10 | 1 |
| Unit Telescope Enclosures | < 0.1 | 6 | 0.6 | 10 | 1 |
| Delay Lines | 1 | 6 | 6 | 10 | 10 |
| Fringe Tracker | 1 | 0.5 | 0.5 | 1 | 1 |
| Total | | | 76 | | 468 |

Table 1: Estimated data rates during science observations for Phase 1 and Phase 2 arrays.

From our summary table we estimate that in Phase 1 (a six telescope array) where there is no visible science instrument and only one near-infrared science camera the total data rate while observing will be ∼76 Mbits/s. In the highly unusual scenario that widefield acquisition and delay line shear camera video are both required during observing the total is closer to ∼110 Mbits/s.

In Phase 2 (ten telescopes) we estimate the total data rate while observing with the fringe tracker and two science instruments will be ∼ 470Mbits/s. Adding widefield acquisition and delay line

shear camera video in this case brings the total to ∼517 Mbits/s. Interestingly, the global data rate and volume is wholly dominated by the contributions from the two back-end instruments.

Finally, the reader should keep in mind that the network will have to be able to handle these data flows whether RTC is chosen as the software framework or not.

# 4 Discussion

## 4.1 Data Flow Characteristics

When discussing the data flow between the various MROI subsystems it is helpful to make a distinction between the various kinds of data that need to be transfered on the basis of their "latency categories":

- **Data which is only stored for future reference.** In this case, the latency requirement is only that the data reaches its destination within hours of being sent. This data can be sent over the TCP/IP network but the transmission might be deferred, say, until daylight hours, to reduce the network load during observing. High volume monitor data and logs might fit in this category.

  Example: science data.

- **Data which is needed by one subsystem after occasionally being generated by another.** These data can be transmitted from the first subsystem via its normal telemetry stream to the data collector, which the supervisor interprets and then commands the second subsystem. Alternatively, a publish/subscribe system (Subsubsection 4.2.2) might be suitable. Fault notifications and some monitor data might fit in this category.

  Example: Weather data needed to inform a decision to open the telescope enclosures.

- **Data which is a continuous stream and has a latency requirement below that of human perception.** This data can be transmitted over the TCP/IP network directly between the subsystems concerned, as doing it via the supervisor will be too slow.

  Example: The slow correction signal from a wide field acquisition system to its associated telescope.

- **Data with short, hard real-time deadlines where correct functionality will not occur unless those deadlines are met.** In this case TCP/IP is not suitable and a dedicated link between the subsystems concerned is required.

  Example: The offset commands sent from the fringe tracker to the delay line subsystem.

Furthermore, it is important to realize that the data flow within the MROI will exclusively be:

- **Asynchronous:** Various subsystems will be able to transmit data to other subsystems at rates that are convenient to them and which are independent of the other subsystems of the MROI.
- **Non-real-time:** In many cases the data delivery time must be short compared to human perception, but variations in delivery time on the scale expected of a TCP/IP network will not compromise the performance of the MROI.

The **only** exception to these rules will be the data flow between the fringe tracker and the delay lines. In this case dedicated links must be used to guarantee performance.

## 4.2 How the MROI data flows might be managed without the use of RTC

As part of the current discussion concerning the feasibility/desirability of managing the MROI data flow in as simple yet effective way as possible, we present below our thoughts as to how this might be realised without relying on RTC.

### 4.2.1 The Data Flow Framework

Any data flow framework for the MROI must meet or exceed the characteristics described in Subsection 4.1. Until recently a ready-made lead contender for the framework has been JPL's Realtime Control (RTC). Its suitability has, however, been questioned due to its use of Common Object Request Broker Architecture (CORBA) for communication between subsystems. This opens the question of how the framework is to be implemented, if not via RTC.

The Cambridge Delay Line Team believes that a lightweight, custom-written framework is the way forward. Such a framework need not be real-time, as the hard real-time network requirements at MROI are subsumed into custom hard links between the subsystems concerned. The MROI is otherwise asynchronous and the time-of-flight requirements on inter-subsystem data traffic are well within the abilities of ordinary TCP/IP, provided that the data volume is not excessive. Indeed, a system which does not rely on hard-real-time data deadlines for correct functionality will more robust when those deadlines are, for whatever reason, not met. RTC itself relies almost exclusively on non-real-time code for delivering data between subsystems.

A custom approach has been shown to work well with the prototype delay lines, where the effort required to develop and document a data flow framework was several person-months. Although those requirements were significantly different from those of the MROI system as whole, that code could easily be adapted to meet the overall MROI data flow requirements.

One advantage of RTC that has not been emulated during the MROI delay line development is the graphical user interface that permits data display as the data are generated. A similar capability would need to be developed if a custom alternative to RTC were chosen. The Cambridge Delay Line Team has experience writing such interfaces for the COAST project and has begun exploring the possibility of doing the same in Gtk, the graphical user interface library used for the prototype delay line project. The open-source domain is also rich in plotting libraries that might be useful. In terms of control interfaces, progress is already occurring in New Mexico[1]. We are confident that an appropriate user interface can be developed straightforwardly in-house.

### 4.2.2 Low Bandwidth Data

Much MROI telemetry will consist of small amounts of information generated at low bandwidth. Currently this information is expected to be handled like any other kind of telemetry, but its diverse, distributed and evolving nature means that a "publish/subscribe" system might also be an elegant solution. Various subsystems could publish data of potential use to other subsystems, for example the environmental monitoring subsystem could publish the local temperature (these data would also be sent to the data collectors as usual for recording). Subsystems that needed to know this information would then subscribe to it. The publish/subscribe system would then signal the subscribers when the temperature reading (for example) had been updated, or possibly only when it had changed.

In this scenario, there would be no need for subsystems or for the supervisor to know which other subsystems might need the data, or for the receivers of that data to poll for it. The system would then become very flexible in terms of adding or modifying subsystems that consume or provide these data.

An example of such a subsystem (although probably one that we would not want to use) can be found at http://alphaworks.ibm.com/tech/rsmb. It has a size measured in kilobytes and shows that such systems can be easily developed. The Cambridge Delay Line Team would prefer that a custom system be developed in-house. We believe that this is a manageable task.

#### 4.2.3 Data Storage

MROI telemetry will need to be collected (by "data collectors") and stored as science and engineering data for future analysis. As the amount of data to be collected is large, any solution must work around the possibly of there being a limited ability of the network to carry this traffic volume without affecting interferometer performance.

Our understanding is that the current MROI paradigm proposes that this data should be stored locally on each subsystem during observing, thereby removing the requirement to transport it over the network. If an observer or engineer wishes to view data, they would interrogate a relational database, which would fetch only the requested data from the subsystems concerned. These data could be buffered so that when the network load was not critical, during daylight hours or during lulls in observing for example, these data would be upoaded to a central repository for archiving.

While this solves the network loading problem, it introduces inconsistency in terms of data analysis because the requested data might span the archive and the subsystems, and additional programming complexity would likely be needed if that inconsistency were to be hidden from the user. Some vendor-sourced subsystems also have no native method of temporarily storing such data.

We prefer, and would suggest, a more direct approach, where data is sent directly to the central repository over the network when it is generated. For local subsystem use when the network is unavailable, an instance of a data collector should also be able to run locally.

This proposal has the drawback that it might heavily load the network: however we would prefer to mitigate that risk by reducing the amount of data transmitted by the subsystems rather than by storing it locally. For example, video frame rates can be reduced, or linear data can be transmitted in bursts – one second of 5 kHz data every 10 seconds, say, for the purposes of near-real-time display. The most important data can still be streamed full-time. There should be a consistent remote interface to each subsystem that controls how much of each stream should be transmitted so that the total network load can be managed and an optimal compromise between data availability and network loading can be reached.

If even this is not sufficient, it would be possible to implement a pair of independent and co-located networks, one dedicated to fast traffic which was to remain lightly loaded, and another for everything else. Alternatively there could be a dedicated line for the most demanding subsystems, the visible science instrument for example.

# 5 Conclusions

We can summarise our conclusions as follows:

- We estimate the data rate during normal science observations with a Phase 1 (six telescope) array to be of order 76 Mbits per second. With a fully commissioned Phase 2 (ten telescope) array that figure becomes 468 Mbits per second.
- From a data flow point of view, the MROI will not be a real-time instrument, although some subsystems have to run locally in real time. This permits the use of ordinary TCP/IP infrastructure throughout the interferometer to coordinate observations.
- A custom, lightweight data flow framework is a viable alternative to RTC.
- A publish/subscribe system might be an efficient way to manage low volume, low rate intersystem data communications.
- The potential volume of telemetry traffic is large enough that it might adversely affect data delivery times. One solution is to retain the data on each subsystem until a convenient time, but the Cambridge Delay Line Team would prefer that less telemetry is generated, and that the amount generated is remotely configurable.

# A Proposed MROI online and offline data model

## A.1 Context

The end purpose of the MROI hardware is to produce a set of data which can be queried by both online and offline software to make decisions (e.g. the conditions are right for good science data), create derived data (e.g. RMS visibility) and to plot it for the "user" (e.g. plot the visibility versus time). Here we attempt to derive an outline format for the data, without getting into the details of any specific implementation. These details can be discussed later after taking into account the available software and other constraints on such an implementation.

Ideally, there would be no difference in the data available to the online and offline software (except that the offline software can "see" data over a longer timespan), and so it would be helpful if the overall structure of the data available were as similar as possible in the two domains. This is particularly the case in the "near-real-time" software, which is essentially identical in function to parts of the offline software, but runs online. We therefore describe as far as possible a data model which is useful both when deciding on data formats "on the wire" – i.e. communicating between different online subsystems — and when deciding on data formats for recording data for offline use. Having a common model will aid automatic translation between these two formats.

## A.2 Generic data model

We define first a simple model which can capture all of the data we need, without reference to efficiency. Later we look briefly at more efficient ways of representing that data.

The data which is the domain of this model is all the data in the online system which might end up being recorded for offline use. To make things simpler, we exclude the commands sent between between subsystems from this model.

We define our dataset as a set of named variables whose values can vary over time. The names are arbitrary strings constructed in a hierarchical fashion, for example

"/UnitTelescope/W15/Config/PointingModel/AzimuthOffset" or

"/FringeTracker/Camera/PixelData".

The values can be either numbers (integer or real), rectangular arrays of numbers (e.g. images or spectra), or character strings (which can be used, for example, to represent a set of discrete states such as "tracking" or "slewing", or just to represent human-readable information). The values are assumed to be sampled at discrete times. These times are arbitrary, and could be at regular or semi-regular intervals or only when the variable changes (for example when a state machine transitions from one state to another).

Our dataset is thus a set of "samples", where we define a sample as the tuple `(name, sampletime, value)`. A dataset comprising all such samples taken over a whole night (plus perhaps some of or all of the preceding day) would be the primary output of the data recording system. The dataset for a given science or engineering program might be a suitable subset of this nightly dataset. An even smaller subset of these data would be available at any one instant to a live display program which is part of the online software.

This data model is very general and can include science data, engineering data, monitor data, configuration data, logs, and alarms — there are examples of this below. If the data is represented using this model, then different kinds of data can in principle be mixed in the same data stream, because the data is to a large extent "self describing", in that we can tell from the name what to do with a variable — software can simply discard data which it is not designed to make use of.

### A.3 Chunked model

Any software infrastructure which allows samples to be transmitted between subsystems and to be recorded on disk will satisfy our basic requirements. In addition however, it would be beneficial if the data were structured in a way that allowed [a] compact representation to save network bandwidth and storage space, and [b] rapid access to the samples of any particular variable over some time interval.

Both of these aims can be facilitated by "chunking" the data, i.e. grouping the data in such a way as to minimise the overhead per sample in terms of transmitting, storing, and retrieving the data. The possible methods of chunking include:

**Time sequences** If there is a not-too-short time sequence of the same variable, then we can group together all the samples of that variable over some interval (e.g. one second or on minute depending on the sample rate of the data) and send/store them as a single packet, thereby saving repeating of the variable name.

**Regular sampling** Where data are sampled at uniform intervals, then the timestamp overhead can be removed by including only a start time, sampling interval, and number of samples for a contiguous sequence of values.

**Simultaneity** Where several variables are always sampled at the same time (typically because they are sampled in the same place), then we need only one timestamp for all the variables sampled at a given time. These data can be represented as a table, with rows representing all the variables sampled at a given moment, and columns representing the time sequence of a given variable. However, there is not much saving of space if the data is regularly sampled, since chunking based on regularity will already have removed the timestamp overhead.

**Kinship** Data not sampled simultaneously but related in some other way can be grouped together. For example, all the configuration data for a given subsystem might be stored together, or all the log data from a given subsystem might be stored together. If all data names have the same prefix in the heirarchical naming scheme then the common prefix need be stored/transmitted only once for all the variables. Again the space saving from this form of chunking is likely to be small.

It can be seen that all the above methods of "chunking" not only help to reduce data volume, but also help to search for data. The most common request for data inside any given piece of data reduction or display software will be to retrieve the sequence of values of some subset of variables over some subset of the total time interval; searching for this subset over a smaller set of chunked variables is likely to be far quicker than searching over a large set of single samples.

The exact chunking of any given set of variables needs to be decided on a case-by-case basis, depending on the volume of data, frequency of sampling and types of implementation overheads (data headers etc) associated with representing a particular variable and so we do not say any further about this here.

## A.4 Data naming and hierarchy

The implied data hierarchy in the variable naming scheme is to some extent arbitrary, and it is likely that different hierarchies will make sense for different applications. For example we might find that `/UnitTelescope/W00/Alarms/Status` and `/Alarms/Status/UnitTelescope/W00` were more or less advantageous under different scenarios. We propose to arrange the hierarchy so that the prefix represents the work package which is responsible for defining the ICD for the given variable. In this way, the work package software lead is in control of the relevant namespace. In the above example the person responsible for defining the format of the alarms is likely to be the WP leader for the supervisory system, so a possible name would name be something like: `"/SupervisorySystem/Alarms/Status/UnitTelescope/W00"`.

Clearly, the names defined in the "subsystem worksheet" and "monitor worksheets" in the ISS conceptual design document[1] could form a basis for the naming scheme for monitor variables, and other worksheets could be developed as needed for other types of variables.

## A.5 Examples

As mentioned above, the data model is quite flexible and can accomodate all the data streams envisaged for MROI in a more or less straightforward way, and we give some examples here.

We define "configuration data" as the data required at startup by any subsystem, plus any data hardwired into the subsystem but which might be subject to change over a long period (for example software version numbers, barcodes on hardware components). It can include variables which are set on initialisation but changed after initialisation in response to a command. Everything else we define as "telemetry data", which includes "science data", "engineering data", "monitor data" (the boundaries between these three are not clearly defined), "log data" (data meant primarily for human consumption for debugging) and "alarm data" (more structured data intended for a semi-automated response).

With this in mind, we can imagine the following as example samples. Timestamps are not shown but are implicitly associated with each sample. Some comments are included in italics.

### A.5.1 Configuration data

- `/UnitTelescope/W15/Config/PointingModel/ModelVersion` String "1.3.4"
- `/UnitTelescope/W15/Config/PointingModel/InnerAxisOffset` Real 0.003456
- `/UnitTelescope/W15/Config/TelescopeId` String "AMOS01"
- `/UnitTelescope/W15/FTT/Config/PixelScale` Real 0.000234 *Pixel scale in degrees*
- `/DelayLine/W0/ShearCamera/Config/FrameRate` Real 30.0 *Value could be changed in later software revisions*
- `/DelayLine/Global/Config/BaselineModel/UnitTelescope/W15/PivotPoint` Real array [15.8456, 40.5678, 1.6] *Meters in x,y,z from array centre*

### A.5.2 Telemetry data

- `/UnitTelescope/W15/InnerAxisRotation` Real 43.54345678
- `/UnitTelescope/W15/FTT/StateMachineState` String "Acquisition"
- `/UnitTelescope/W15/FTT/NasImage` Real array [512x512] *Values not given here!*
- `/SupervisorySystem/Logs/DelayLine/W00/Info` String "Searching for datum"
- `/SupervisorySystem/Logs/SupervisorySystem/Operator` String "This data looks great. Nature paper in the bag"
- `/SupervisorySystem/Alarms/DelayLine/Global/State` String "OK"

# B Outline implementation

One possible way of transmitting these samples "on the wire" would be to use the "serialise" library[3], which is in use at COAST and as part of the `dlmsg` library in the delay line subsystem, to encode samples. The exact format of the messages would need to be decided on, but, for example, the following C code would send samples of two variables, one chunked and one not, from the fringe tracker to the data collector using the "serialise" library:

```
SendMessage(msgSocket, status, "(si(sssdsd)(sssdsdsIsD))",
            "telemetry",protocolRevisionNumber,

            "name","/FringeTracker/DewarTemperature",
            "sampleTime", sampleTime,
            "value",dewarTemperature

            "name","/FringeTracker/GroupDelayEstimates",
            "startTime", startTime,
            "sampleInterval", groupDelaySamplingInterval,
            "dims",groupDelayShape
            "value",groupDelay,numBaseline*numSample,
           );
```

When such a message arrived at the data collector, the relevant samples would be forwarded as messages (in the same format) to any subsystems requesting these particular variables, and the samples would also recorded to disk

The on-the-wire data and the recorded data file could have essentially the same format, but perhaps differing in the amount of "chunking" that was applied to the data. In addition, the stored data format might include an index which would help in locating any given sample within the file. The data recording system would then be a fairly simple and generic piece of software, and would not need to be rewritten when a new variable is added to the system.

## References


[1] A. Farris, MROI Supervisory System: A Conceptual Design Overview, internal document (WP 4.09.03 Version 1.0), 20 February 2009.

[2] A. Barbalace *et. al.*, Performance Comparison of VxWorks, Linux, RTAI and Xenomai in a Hard Real-Time Application, *IEEE Transactions on Nuclear Science*, Vol.55, No.1, February 2008, pp.435–439.

[3] D. F. Buscher, Proposed network message protocol for COAST, COAST internal document, January 2000.