

Controller area network for monitor and control in ALMA

Michael J. Brooks^a

National Radio Astronomy Observatory, 949 N. Cherry Ave Tucson AZ 85721 USA

ABSTRACT

The Controller Area Network (CAN), initially developed for the automotive industry, is becoming increasingly popular in industrial process control applications. The need for distributed low data rate monitor and control networking in industry is similar to the needs of the various instrumentation and support equipment in a modern radio telescope. In particular, immunity to noise and low radio frequency emission characteristics are common to both domains.

The Atacama Large Millimeter Array (ALMA) project has adopted CAN technology for use in local monitor and control applications at each of its 64 antennas. A standard interface slave node providing flexible I/O options is under development and a simple application-level protocol making use of CAN to access these nodes in a master/slave fashion has been implemented.

This paper will present the work which has been completed to date including experiences in the use of CAN in an astronomical environment. In addition, analysis and simulation of CAN networks is compared with the performance of our implementation in the lab.

Keywords: ALMA, CAN, field bus, distributed control

1. INTRODUCTION

This paper presents the author's work with the ALMA project over the course of 1999 and early 2000. The paper is structured in the following manner. An introductory section will give some background on the ALMA project and the Controller Area Network bus protocol. In Section 2, an application level protocol developed by the author is described for use with CAN. Performance aspects of the proposed protocol are analyzed in Section 3 making use of specific data available regarding the monitor and control requirements for the subsystems at an ALMA antenna. In Section 4, an experimental network is described and the results of performance testing on this network are described. The analyses and experimental results together show that the proposed protocol and CAN are sufficient and appropriate as a solution to ALMA's monitor and control requirements.

1.1 The ALMA project

The Atacama Large Millimeter Array (ALMA) will be a millimeter wavelength synthesis telescope. The project is a joint U.S. and European collaboration; the U.S. side of the project is run by the National Radio Astronomy Observatory (NRAO). The European side of the project is a collaboration between the European Southern Observatory (ESO), the Centre National de la Recherche Scientifique, the Max-Planck-Gesellschaft, the Netherlands Foundation for Research in Astronomy and Nederlandse Onderzoekschool Voor Astronomie, and the United Kingdom Particle Physics and Astronomy Research Council.

In its current conception, this project will comprise no less than 64 12-meter antennas located at an elevation of 16,400 feet (5000 meters) in Llano de Chajnantor, Chile. These antennas may be moved to array configurations where the maximum

^a Correspondence: Email: mbrooks@nrao.edu; WWW: <http://www.tuc.nrao.edu/~mbrooks>; Telephone: +1 520 882 8250 x162; Fax: +1 520 882 7955

baselines range from approximately 150 meters to 10 km. ALMA will be largest and most sensitive instrument in the world at millimeter and sub-millimeter wavelengths.

At each antenna and at the central control building there will be many subsystems requiring monitor and control (M&C). The total number of subsystems at an antenna is approximately 30 and a per antenna total of nearly 1000 M&C points⁵ is divided among these subsystems. These 1000 points are roughly split up as 70% monitor data and 30% control data.

The typical data rates for much of the M&C at an ALMA antenna are quite modest, around 4.1 kBps (kilobytes per second)⁶, but several items require accurate timing of delivery, most notably the axis position commands to be sent to the mount control computer. Most of the subsystems at an antenna are custom built hardware requiring little computational support, and given the data traffic characteristics, an approach based on distributed input/output (I/O) rather than distributed computing was investigated. Such a distribution of interface devices with limited intelligence requiring low data rates and guaranteed delivery latencies led us to investigate industrial field buses as a solution. The environment in which the antenna network is required to operate will be harsh because of the altitude and remoteness, and likely to be electrically noisy in the areas near compressors, mount drives and digital samplers. A sturdy protocol with robust error handling is clearly required; the outcome of this investigation was a decision to use the Controller Area Network (CAN) for M&C communications with most antenna subsystems requiring monitor and control.

It should be noted that in addition to the use of CAN for monitor and control data, there will also be a general purpose network available at each antenna. Although a precise design does not exist at this point, a local area network, possibly based on ATM, will connect the various antennas to a central computer room and a link will then bring all signals to the control center in San Pedro.

1.2 The Controller Area Network

CAN was originally developed by Robert Bosch GmbH in the late 1980s for use in automotive applications. It is now an ISO standard communication protocol³, which covers Layers 1 and 2 of the OSI 7 layer protocol stack. For the physical layer, a twisted pair multi-drop cable is specified with lengths ranging from 1000m at a bit rate of 40 kbps (kilobits per second) to 40m at a bit rate of 1 Mbps. The media access control layer is based on a Carrier Sense Multiple Access with Collision Detection and Resolution (CSMA/CD,CR) approach.

One of the driving factors behind the development of CAN and other multi-drop automotive buses was the complexity inherent in a conventional multi-wire systems wiring harness required within the vehicle. CAN was developed to address this wiring complexity by replacing the harness with a twisted-pair multi-drop bus system. In addition, the bus should be useful for both “class C” applications such as engine and brake control which have critical real time requirements and for “class A” applications such as mirror, power windows and door lock control where the real time requirements are less important. Since 1994, CAN has been the most accepted protocol for automotive applications¹. More recently, the industrial control industries have recognized CAN as a powerful communication protocol for use in factory automation, and supplementary specifications have been added to help make CAN an open system suitable for use in this environment. These standards now include cable types, connectors and transceivers. CAN has also been used in experimental physics and astronomy applications¹⁰. In addition, several OSI layer 7 application protocols have been developed but are as yet not standardized.

CAN framing exists in two variants, one with an 11 bit address field and the other with a 29 bit address field. Both may coexist on a single network. The data payload of a CAN frame is zero to eight bytes; the maximum frame length is 135 bits. Figure 1 illustrates the format of an extended CAN frame, with the 29 bit address field, which is the chosen standard for use in ALMA. Each field is shown with its length in bits.

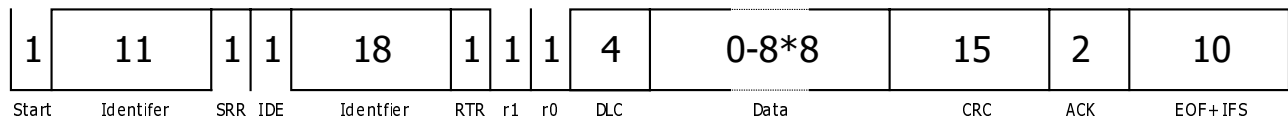


Figure 1: CAN 2.0 B Extended Identifier frame format. The name of each field is shown, together with the number of bits in the field. The data field is the only one of variable length.

- The start bit is a dominant low value, and all nodes synchronize to the falling edge of this bit.
- The first eleven bits of the identifier or address field follow.
- The Substitute Remote Request (SRR) bit has no meaning; it is included for compatibility with the 11 bit identifier CAN frame format.
- The Identifier Extension (IDE) bit is always a high level and indicates that this is an extended format frame.
- The remaining 18 bits of the identifier or address field follow.
- The Remote Transmission Request (RTR) bit is used by receivers to request transmission of a CAN frame. If this bit is set the frame contains no data field; the requested data is transmitted in a separate frame following the request frame.
- The r0 and r1 fields are reserved and always transmitted as low or dominant bits.
- The Data Length Code (DLC) field contains the length, in bytes, of the data field which follows.
- The 15 bit Cyclic Redundancy Check (CRC) field is capable of detecting up to 6 non-consecutive single bit errors or burst errors up to a length of 15 bits.
- Every active network node that detects a correct message on the bus overwrites the second bit in the Acknowledgement (ACK) field to indicate to the transmitter that the message has been received. This acknowledgement does not indicate that an application has correctly handled the data frame, only that the message was correctly transmitted on the CAN bus.
- The End of Frame (EOF) field marks the end of a CAN frame and the Inter-Frame Space (IFS) field is the minimum length of time before more CAN frames may be placed on the bus.

The CAN bit stream is encoded with a differential Non Return to Zero (NRZ) method such that a logical zero is dominant on the bus. All nodes on the bus must be able to write or sample within the same bit time and this leads to the length limitations becoming increasingly more severe with increasing bus length. The collision resolution is handled by this zero-dominant bit synchronous method as well. To guard against loss of synchronization, a transmitting node introduces stuff bits when more than 5 consecutive dominant bits occur in the bit stream.

If two slaves begin transmitting simultaneously, an error will be noted when one slave tries to write a “one” to the bus and the other slave writes a dominant zero. The slave transmitting the “one” will concede access to the other transmitting node. Because this process will occur during the transmission of the identifier field, CAN identifiers also denote the priority of a given message. Given the necessary discipline to ensure that only one node on a bus is the transmitter of a given identifier, it can be shown⁷ that a bounded delay may be calculated for every CAN message if the frequency of each CAN message transmission is known.

CAN framing includes quite a bit of overhead. At the maximum rate of 1 Mbps, the rate chosen for use within ALMA, a single bit transmission time is 1 μ s. The maximum transmission time of any CAN message m is then given by:

$$C_m = \left(\left\lfloor \frac{34 + 8s_m}{4} \right\rfloor + 47 + 8s_m \right) \mu\text{s} \quad (1)$$

where s_m is the size of message m in bytes. The first part of the sum is to account for the possibility of stuff bits and 47 is the maximum bit overhead of a message. So for 8 bytes of data, a CAN message will take 135 μ s to transmit which is 71 bits of overhead. A frame with no data would take 55 μ s for transmission.

Note that CAN is inherently a multi-master protocol. For each message on a CAN bus, there should be one transmitter but there can be any number of receivers.

2. A LAYER 7 PROTOCOL FOR USE IN ALMA

As the CAN standard outlines only the lower layers of the OSI model, it is necessary in any sufficiently complex system to also specify an application layer. At a bare minimum this protocol needs to specify the allocation of CAN message identifiers to nodes, either as the sole transmitters or as a receiver, and the meanings of the data bytes in the various messages.

Several commercial protocols are available for the application level. Some of these are:

- DeviceNet
- Honeywell SDS
- CAN Application Layer (CAL)
- CANopen
- CAN Kingdom

The first two are very proprietary systems requiring you to buy software and hardware from the vendor. The remaining three protocols are all fairly similar. While they offer peer-to-peer communication during operation, a master is designated for the configuration phase. This configuration phase is quite elaborate and requires the master to transmit information to each node detailing the CAN IDs that that node will use for transmission and reception of data. An implication of this is that when a reset or power outage occurs, each node except the configuration master “forgets” its identity. Processing requirements for these protocols are far greater than for CAN alone.

Within the ALMA project, the current proposal is for a simple non-standard master-slave protocol comprised of three parts as defined in the sections below. An advantage of this protocol is that it avoids a complex configuration phase, allowing slave nodes to boot up ready to run. The master-slave nature of the protocol also allows us to further control access to the bus increasing the determinism of the system.

This protocol presupposes that each slave node has two pieces of information, a unique 64 bit serial number and a 6 bit node address. Typically the serial number is obtained from a Dallas Semiconductor Silicon Serial Number device which guarantees a unique 64 bit number. The node address may be settable by a DIP switch on the slave hardware or read from a hardwired backplane frame. Because the CAN bus is electrically limited to 64 nodes, a six bit address is sufficient coverage and leaves a wide range of addresses available for each node to use with the remaining 23 bits.

In the following discussion, a number with a prefix of “0x” should be interpreted as a hexadecimal number.

2.1 Slave node identification protocol

When the bus master transmits the CAN message with identifier 0x0000000, it is interpreted by all slave nodes on the bus as a broadcast request for identification. Each slave node responds by transmitting a CAN frame with an identifier calculated as follows:

$$CAN_ID = (node_address + 1) \times 0x40000 \quad (2)$$

The serial number of the slave is transmitted in the eight data bytes of this identification frame. The identifier calculated in Equation (2) marks the beginning of a range of addresses allocated to the slave node and extending for 0x40000 identifiers. The slave node will respond to or accept CAN message identifiers only within this range.

If the master has received no response frame for 200ms, it is assumed that all slave nodes on the bus have responded and the identification sequence is terminated.

It is possible that two slave nodes will be inadvertently initialized to the same node address. Both nodes would then respond to the identification request by transmitting their different serial numbers as data in a message with the same CAN identifier. This situation may be detected in two ways. If the two slaves respond at different times, the bus master will notice that two slaves responded with the same address. It may then take appropriate action. If the slaves respond at exactly at the same time, a CAN bit transmission error will occur in one of the slaves while transmitting the serial number. This slave would then cease responding to further messages while the other slave would behave as normal.

2.2 Control transactions

Control transactions consist of the transmission of 1 to 8 data bytes from the master to a slave node. The CAN identifier must be within the range defined for the target slave by its node address. The number of data bytes transmitted is encoded in the DLC field of the CAN frame (see Figure 1). Acknowledgement of receipt of the frame is performed by the CAN hardware in

the ACK bits shown in Figure 1. Any errors in the actual data or further processing performed by the slave should be reported in a monitor transaction.

2.3 Monitor transactions

These transactions consist of the bus master requesting information from a slave node. The master transmits a frame with zero data bytes and a CAN identifier in the range defined for the target node by its slave address. Within 150 μ s, the slave should begin transmitting the requested data using the same CAN identifier. The number of data bytes is encoded in the DLC field of the response frame (see Figure 1). Because both the request and response frame make use of the same identifier, the normal CAN restriction of one designated transmitter node for each identifier is violated. This is mitigated by the polled nature of the protocol.

3. ANALYSIS

3.1 Calculated transaction rates

For a maximum data size of 8 bytes and at a bus speed of 1 Mbps, a CAN frame would take as long as 135 μ s according to equation (1). Ignoring the interrupt handling latency of the bus master and any hardware delays in transferring the CAN frame internally, a worst case figure for monitor and control transaction times may be calculated. Note that a control transaction consists of the master’s transmission of the frame only; the monitor transaction consists of the monitor request by the master, a worst case delay of 150 μ s to allow for slave processing and the slave transmission of the data frame.

Table 1: Calculated times to process a transaction for the bus master’s point of view.

	Expected Worst Case Transaction Time
Monitor Transaction	$55 + 150 + 135 = 340 \mu\text{s}$
Control Transaction	$135 \mu\text{s}$
Identify Sequence	$200 \text{ ms} + 55\mu\text{s} + 63 \times 135 \mu\text{s} = 209 \text{ ms}$

Given these transaction times, a worst case rate for 8 byte monitor transactions would be close to 3000 transactions per second. Note again that this ignores the bus master processing latencies and these are significant, as we shall see in Section 4.

The transaction time for the identification sequence consists of the timeout value, 200 ms, the transmission time for the broadcast identification request and the transmission time for 63 slaves to respond with their serial numbers. A CAN bus is electrically limited to a maximum of 64 nodes. The choice of 200 ms for the timeout is entirely arbitrary and any value would suffice provided it is longer than the transmission time of the CAN message containing the serial number. A 1 ms timeout, for example would allow the identify sequence for 63 slave nodes to be completed in 10 ms.

3.2 Rate monotonic analysis of CAN schedulability

Because the protocol outlined in Section 2 is a master slave system, the priority and handling of each transaction is wholly under the control of the bus master. This means that bus access may be treated as a resource allocation problem, and the method of Rate Monotonic Analysis (RMA) may be applied⁹. The treatment here follows that suggested in Reference 9.

Reference 5 is the ALMA software group’s current estimate of monitor and control points in the ALMA system. It is broken down by subsystem and includes data sizes, periodicity and delivery timing requirements. For each of the M&C points at an antenna subsystem, a CAN message was assigned. For each assigned message, the transmission time is computed from the estimated data size. For control messages this represents the bus resource allocation required by the message (processor usage in traditional RMA). For monitor points, the 55 μ s overhead for the request is added as well. No allowance was made for additional delay in the form of analog to digital conversion times or other slave processing, or for the queuing delays in the bus master.

Blocking of the bus resource is given by the maximum message transmission time over all messages. That is, the longest a message will block waiting for access to the bus is the transmission time of the longest message. Assuming some priority

management by the bus master, the maximum blocking delay is caused when a message transmission is already in progress. A higher priority message will have to wait until transmission ceases before it can access the medium even if the message in progress is of lower system priority.

The maximum message transmission time for a 1 Mbps CAN bus with the maximum message length of 8 bytes and a maximum bit stuffing overhead is 135 μ s. Individual blocking terms are then given by:

$$B_m = \frac{135}{T_i} \quad (3)$$

where T_i is the message transmission period as given in Reference 5 and expressed in microseconds.

The total utilization of the bus for all n messages is then calculated using the expression for transmission time of each individual message, C_m , given in equation (1):

$$U_{total} = \sum_{i=1}^n \frac{C_i}{T_i} \quad (4)$$

Non-periodic messages are treated as periodic with very long periods.

The total blocking term is also calculated:

$$B_{total} = \max_{i=1}^n \left(\frac{B_i}{T_i} \right) \quad (5)$$

A single utilization bound is found which depends only on the number of messages attempting to access the bus resource:

$$UB(n) = n(2^{1/n} - 1) \quad (6)$$

If the following inequality holds, then the message set can be scheduled.

$$U_{total} + B_{total} \leq UB(n) \quad (7)$$

This analysis assumes that the deadline for each message is equal to the period of that message which is true for many cases of soft real time data but not true for many of the hard cases. This analysis provides a quick and sufficient test for the schedulability of a set of messages. An Excel spreadsheet was used to calculate the above quantities.

A total of 937 distinct CAN messages were found, leading to a total utilization bound ($UB(n)$) of 0.6934 from equation (6).

The maximum blocking term, B_{total} , for all messages was 0.0335 (equation (5)). The total utilization, U_{total} , from equation 4 is 0.5424 so that the left hand side of the inequality in equation (7) is equal to 0.5759, which is significantly less than the bound of 0.6940. From this we conclude that the offered message load is schedulable.

The effect of prioritization of CAN messages within the bus master is still to be investigated. This model includes the queuing delays governed by master and determined experimentally as presented in the next section.

4. EXPERIMENTAL RESULTS

An experimental CAN network was constructed in the NRAO labs in Tucson, consisting of one master and two slave nodes. The master node was implemented with a Motorola MVME1603 embedded computer. This system consists of a 66 MHz PowerPC 603 CPU mounted on a VME backplane. The motherboard includes a PCI bus and has one PCI Mezzanine Connector (PMC) available. On this connector an SBS Greenspring Extended CAN Bus module was installed. This is actually a TEWS Datentechnik TP816 CAN board. The board is essentially an Intel 82527 CAN controller with its registers mapped to the PCI bus. The master protocol software implementation was developed in C for the VxWorks operating system using the WindRiver Tornado II toolset. SBS Greenspring supplied the device driver for the PMC CAN board.

The two slave nodes used in testing are both essentially the same hardware based on the Siemens 80C167CR 16 bit micro-controller with on-chip CAN controller. One of the slave boards had slightly faster RAM and FLASH memory chips than the other, however. Both boards were modified to include DIP switches for setting the slave node address and a Dallas Semiconductor DS1820 device which provides the serial number as well as the ability to monitor the ambient temperature. The slave protocol software implementation was developed in C with the Keil Electronic compiler and toolset for the C167. No operating system was used.

Tests of the slave node identification phase described in Section 2.1 were found to be dominated by the 200 ms timeout period. Each slave identification takes less than 500 μ s, so a bus of 30 slave devices would take about 215 ms to identify all nodes. The 200 ms timeout is quite arbitrary, and any value will work as long as it is greater than a CAN frame transmission time. Thus this identification sequence could take as little as 16 ms for a 30 slave bus.

In the remaining tests, a number of monitor and control transactions were made to each slave. The total time to complete a transaction was measured at the master node and from this data throughput and transaction rates were calculated. The initial tests concentrated on either monitor or control transactions only with fixed data sizes. In the later tests, a mix of monitor and control transactions was used to more closely simulate the conditions on an antenna bus.

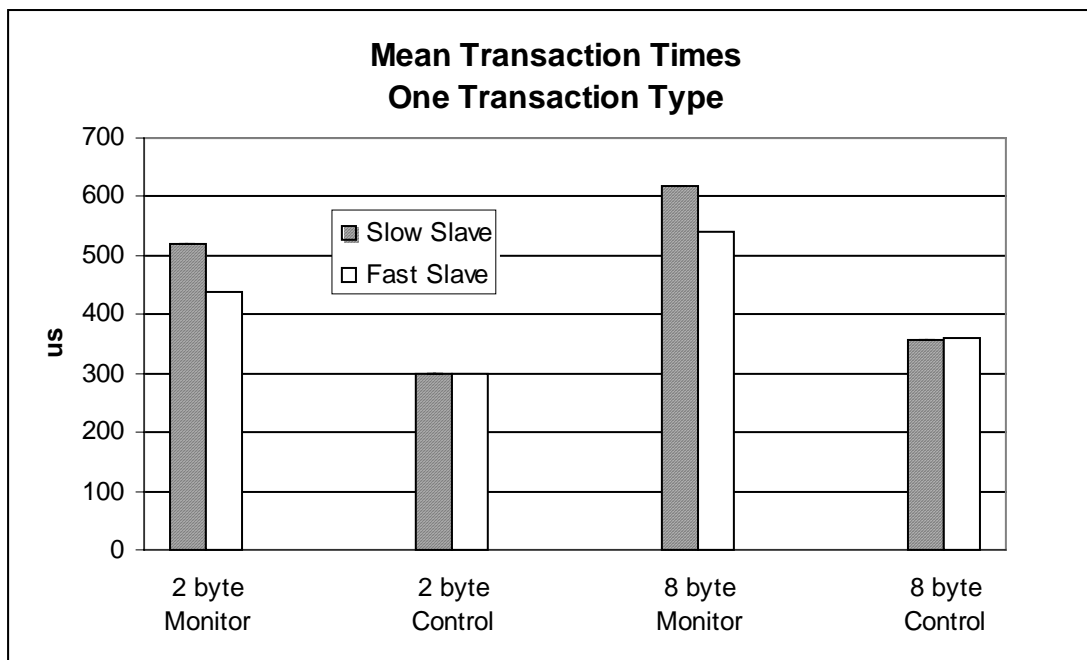


Figure 2: Average times for bus master to process transactions. In a test, 10000 transactions of a fixed data size were timed. Each test consisted of monitor or control transactions only.

Figure 2 illustrates the average times to perform monitor and control transactions of differing data payload sizes. Ten thousand monitor or control transactions were performed to obtain the results. Variances were extremely small, of the order

of picoseconds. The graph demonstrates that control transactions do not depend on the speed of the slave node, but that monitor transactions do.

Using an oscilloscope, the response time of both slave nodes was measured. The “slow slave” took about 140 μ s to begin transmitting a monitor response for an 8 byte message. The “fast slave” took about 80 μ s due to faster memory devices. As an 8 byte CAN message should take 135 μ s to transmit, the figures for control transactions indicate several hundred microseconds of interrupt latency and context switching on the master system. This is partly due to the antiquated nature of the CPU and partly due to inefficiencies in the coding of the device driver. Work is currently proceeding to improve the performance of the master.

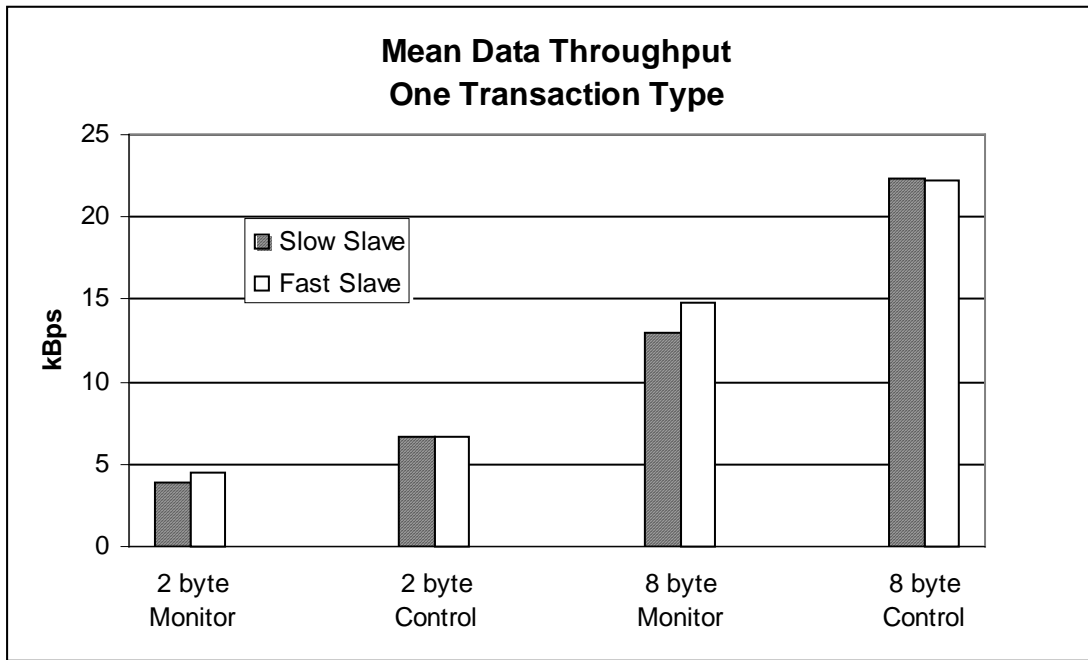


Figure 3: Average payload throughput for differing data field sizes. Each test consisted of 10000 transactions of monitor or control data only.

Figure 3 shows the net data throughput of the protocol considering the overheads of CAN framing bits, master system message handling latencies and slave response times. As expected with higher payload sizes, throughput increases. Note that even with the poor master system implementation and an inefficient 2 bytes per transaction, the data throughput is sufficient to meet the data rates anticipated^{5,6}.

For the following experiments, a mixture of control and monitor transactions were performed and timed. Again, ten thousand transactions were run but each transaction was randomly performed as one of the two transaction types. This allows us to examine the bus performance with a load approximating the final ALMA antenna bus. Loads of 70% monitor data and 30% monitor data were offered for varying data payload sizes as before. The results in terms of transactions per second and data throughput are illustrated in Figure 4 and Figure 5, respectively.

Current estimates of antenna bus loading find the 70% monitor data case the most accurate forecast of the actual load. In Figure 4 we find that a transaction rate close to 2000 transactions per second or greater is achieved under all mixes of traffic type and data payloads. This rate of 2000 was a target rate arbitrarily set to ensure a 1 ms per transaction goal. With improvements in the master system hardware and the master device driver these numbers should increase under similar loads. As would be expected, the transaction rates are higher for the cases where control transactions dominate the offered load. This is due to the fact that there are two CAN transmissions for each monitor transaction, but only one CAN transmission for each control transaction.

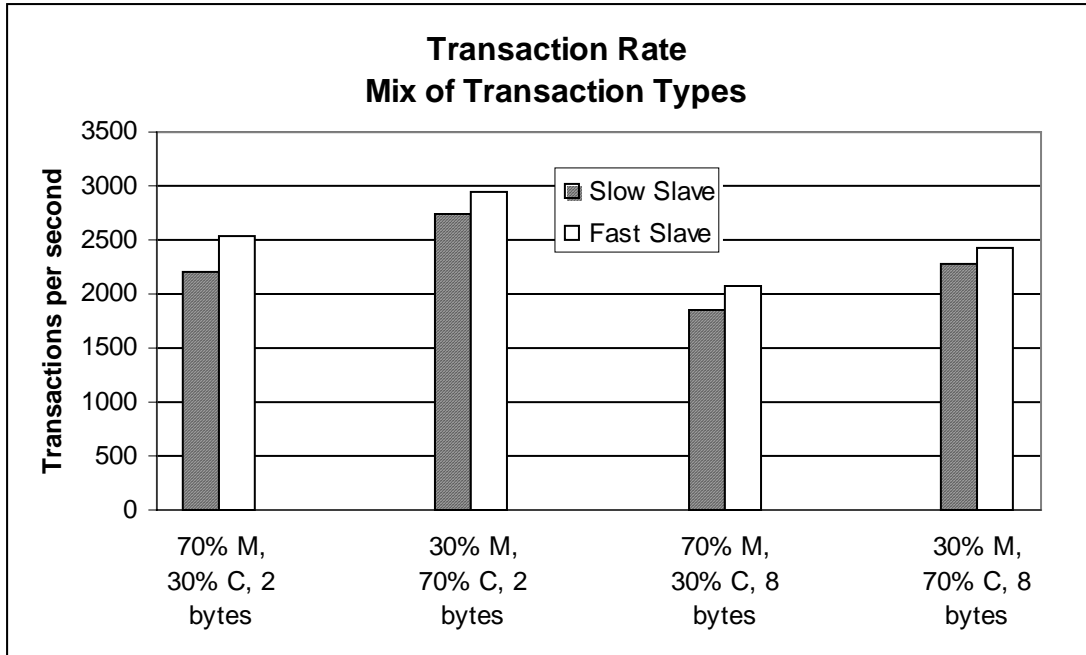


Figure 4: Average transaction rate for tests with a mix of monitor and control transactions. Again, 10000 transactions were performed, each with a fixed data payload size.

In Figure 5, we again see the increase in data throughput as the data payload size increases and the cost of the framing overhead is lessened.

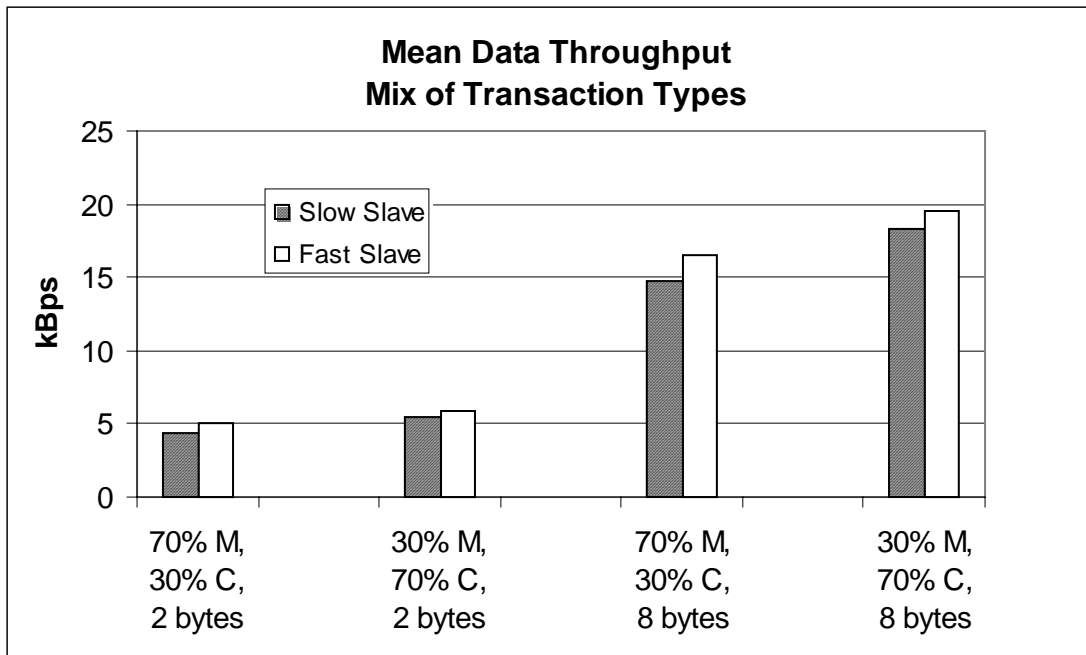


Figure 5: Average payload throughput for differing data field sizes. Each test consisted of 10000 transactions of mixed monitor and control data in the proportions given.

The slowest rate found (4.4 kbps) is still sufficient for the estimated rates but since much of the data will be larger than 2 bytes per transaction, this is very much a worst case.

5. CONCLUSIONS

The analysis work carried out and the experiments demonstrate that the CAN bus used in conjunction with the application level protocol in Section 2 is quite suitable for use in low data rate monitor and control applications. Field reliability issues, and noise and error handling capabilities are still to be investigated in a pilot project on NRAO's 12m radio telescope on Kitt Peak.

The protocol specification and bus concept is currently subject to the ALMA documentation review process. It is intended that a second CAN network be set up at ESO in Garching and that the CAN device drivers for VxWorks be integrated into the software group's proposed ALMA Common Software (ACS) by the third quarter of 2000. In addition to the VxWorks support for CAN mastering, LabView code has been developed for use as a CAN bus master. LabView is intended for use by hardware subsystem designers to gain access to monitor and control within their devices under development in the lab and before a coherent ALMA software system is available to support such testing.

ACKNOWLEDGEMENTS

Thanks to Brian Glendenning, Ron Heald, Gianni Raffi and Steve Scott for reviewing drafts of this paper. Gareth Harris, Jim Pisano and Fritz Stauffer also contributed many comments on the basic design and concepts.

REFERENCES

1. W. Lawrenz, *CAN System Engineering*, Springer-Verlag, New York, 1997
2. M. Brooks, "ALMA-US Computing Memo #7 ALMA Monitor and Control Bus Draft Specifications", 1999
3. ISO 11898:1993 Road vehicles - Interchange of digital information - Controller area network (CAN) for high-speed communication
4. F. Halsall, *Data Communications, Computer Networks and Open Systems*, Addison-Wesley, Wokingham, 1992
5. F. Stauffer, "ALMA-US Computing Memo #1 Monitor and Control Points for the MMA", 1999
6. M. Brooks, "ALMA-US Computing Memo #6 ALMA Monitor and Control System, 1999
7. K. Tindell and A. Burns, "Guaranteeing Message Latencies on Controller Area Network (CAN)", *Proceedings 1st International CAN Conference*, Mainz, Germany, September 1994
8. M. Brooks, "ALMA-US Computing Memo #5 ALMA Monitor and Control Bus", 1999
9. M. Klein, T. Ralya, B. Pollak, R. Obenza, M. Harbour, *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*, Kluwer Academic, Boston, 1993
10. H. Burckhart, D. Fernández Carreiras, M. González Bergés, H. Milcent, D. Myers, "A Prototype Application for Evaluating EPICS and CAN Bus", CERN Internal Report, <http://itcowww.cern.ch/EPICS/epicscan/REPORT.HTML>, 1998