Reflective Memory vs Traditional Networking

J-Squared Technologies Inc.
REFLECTIVE MEMORY DIVISION

Toll Free: 1.855.365.2188

Local: 1.613.592.9540

rfm@jsquared.com

Reflective **Memory**

What is REFLECTIVE MEMORY?

A Reflective Memory network is a special type of shared memory system designed to enable multiple, separate computers to share a common set of data.

Reflective memory networks place an independent copy of the entire shared memory set in each attached system. Each attached system has dull, unrestricted rights to access and change this set of local data at the full speed of writing to local memory.

When data is written to the local copy of Reflective Memory, high speed logic simultaneously sends it to the next node on the ring network. Each subsequent node simultaneously writes this new data to its local copy and sends it on to the next node on the ring. When the message arrives back at the originating node, it is removed from the network and, depending on the specific hardware and number or nodes, every computer on the network has the same data at the same address within a few microseconds.

Local processors can read this data at any time without a network access. In this scheme, each computer always has an up to date copy of the shared memory set. In the four-node example shown, it takes 2.1 µs for all computer to receive the data that was written to Reflective Memory.*

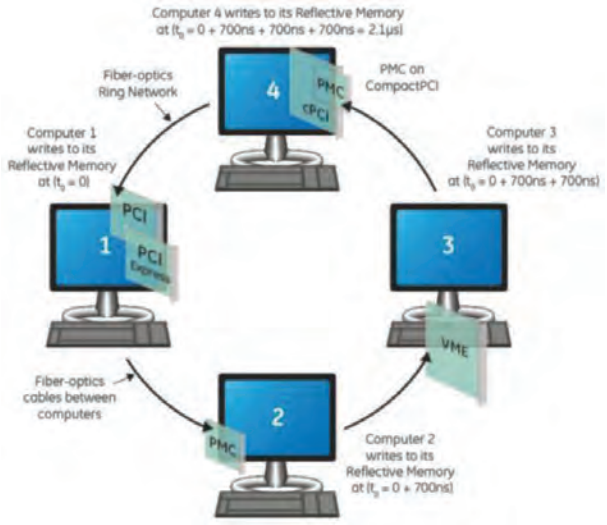

***Figure 1** Reflective Memory provides very low latency between nodes.*

** This latency is calculated assuming no network traffic, short cable lengths and the largest packet size is possible. Cable length and network traffic can cause the latency to increase, but as long as the bandwidth of the network is not exceeded, the latency should not increase significantly.*

DETERMINISTIC DATA TRANSFERS

Reflective Memory is a hardware-based network. All data transferred to a node is stored in local memory and automatically sequenced out to all the other nodes' memory. There are no software delays and minimal hardware delays associated in the data transfer. Any latency is imposed at the hardware level and can be predetermined within a very small window of best-to- worst case latency. The determinism of Reflective Memory, the guaranteed time in which communication between two or more nodes is completed, allows system designers to build effective real-time LANs that can guarantee data delivery within a tight window of time. This enables guaranteed scheduling of sequential actions and ensures that data is not lost.

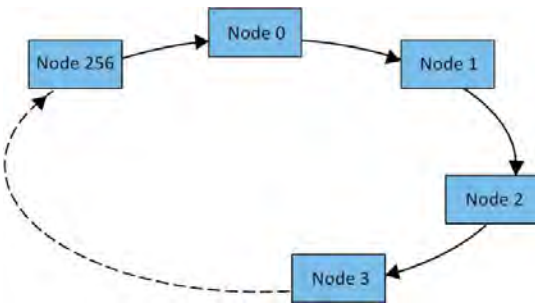

***Figure 2** Reflective Memory Ring Architecture connects up to 256 separate network nodes in real time*

Why CHOOSE REFLECTIVE MEMORY?

Reflective Memory LANs or Real-time Networks are usually constructed because the designer has needs or problems that are solved by one or more of the following Reflective Memory board characteristics:

- DETERMINISTIC DATA TRANSFERS
- HIGH-SPEED PERFORMANCE
- EASE OF USE
- OPERATING SYSTEM AND PROCESSOR INDEPENDENCE
- ECONOMICS AND AVAILABLE TIME-TO-BUILD SYSTEMS
- ADVANTAGES OVER STANDARD LAN TECHNOLOGIES

WEAKNESSES OF TRADITIONAL NETWORKING FOR DISTRIBUTED COMPUTERS

There are many ways to transfer messages or large blocks of data between systems, and each method has its own unique capabilities and limitations. The simplest data transfer technique uses bus repeaters to transfer the CPU read and write signals from one computer to the backplane of another computer. A second technique, Direct Memory Access (DMA), moves data between the global memories of two or more computers. DMA requires backplane control from the local processor. Other methods include message passing via a single shared-global RAM, and standard LANs like Ethernet and Gigabit Ethernet.

BUS REPEATERS

A bus repeater connects the CPU backplane of one computer to the CPU backplane of another computer as shown. This connection allows message passing between CPUs, and also allows each CPU to access resources in the other computer. Since bus transfers may occur at any time and in any direction between computers 1 and 2, a bus arbitration cycle is required on every read or write cycle between the two systems.

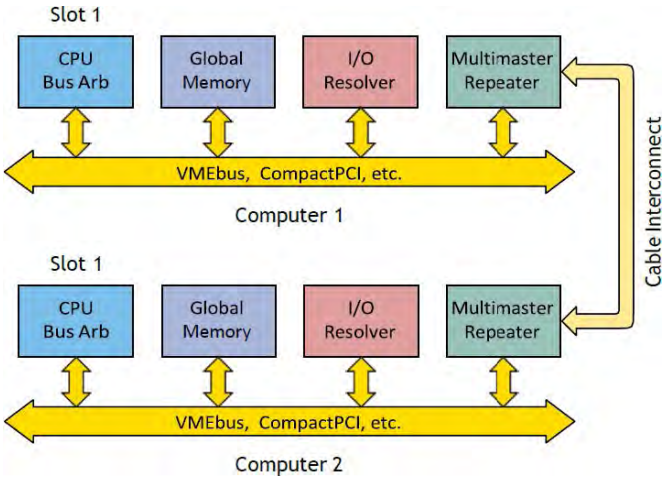

Figure 3 Bus Repeater connection

The problem with this approach is that each time a CPU wants to access a resource in a remote backplane, it must first request access to the remote backplane, and then wait until that access is granted. Depending on the priority level and type of other bus activity taking place in the remote backplane, this might take anywhere from several microseconds to several milliseconds. This overhead delay not only impedes the data transfer, but it also ties up the requesting backplane, blocking any other activity in this backplane until the remote access is completed. As the systems spend more and more time waiting for each other, the compounded latency delays become prohibitive for real-time applications.

DIRECT MEMORY ACCESS (DMA)

Bus repeaters can be very efficient for moving small amounts of data (such as bytes or words) from backplane to backplane. However, in many distributed multiprocessing systems larger amounts of data are exchanged between the various CPUs in the form of parameter blocks. These blocks of data can be moved more efficiently by using DMA controller boards, like the one shown.

In these connections, the CPU in each system initializes the address register and the size register on its own DMA controller board. In this process, the address register on the originating DMA controller board indicates where the DMA controller should begin reading the parameter block from global memory. The address register on the destination DMA controller board indicates where the DMA controller should begin storing the parameter block. Once the two CPUs have initialized their respective DMA registers, the transfer is automatic, and the CPUs can direct their attention to other activities.

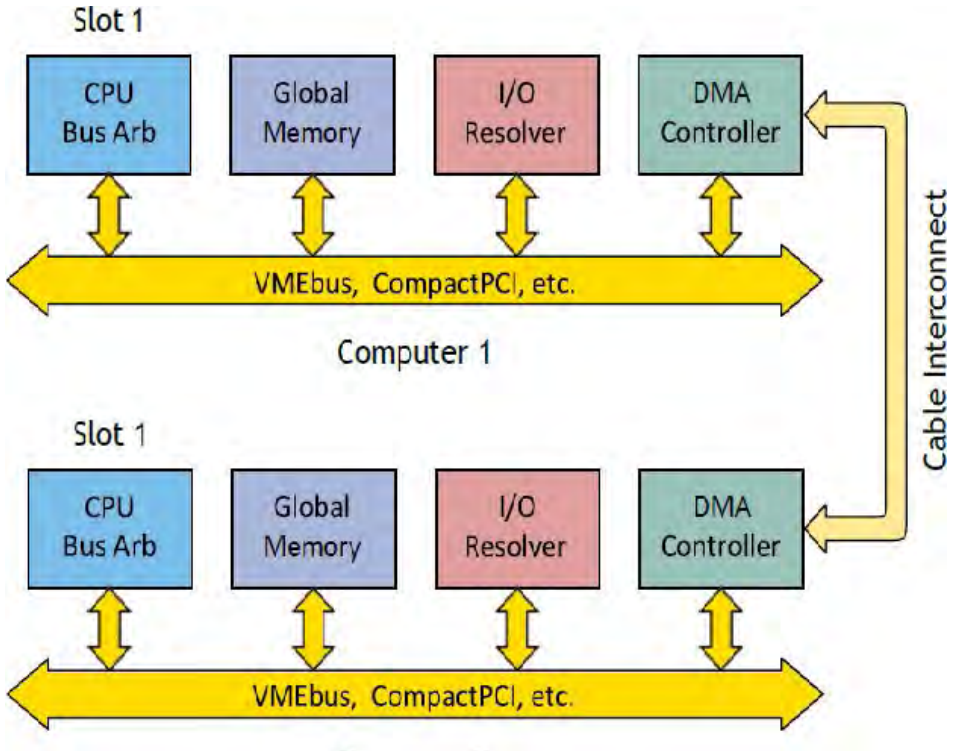

Figure 4 DMA Controller connection

DMA transfers can occur at a very high rate of speed. That is, once all the above overhead programming and setup has occurred. Both the originating and destination computers must have active involvement in the data transfer. Most importantly, every time a block transfer is completed, both processors must be interrupted so they can reconfigure the DMA controller to prepare for the next transfer. While these DMA transfers are occurring, each local processor must share the available bus bandwidth with its DMA board. This setup can be efficient in certain circumstances, but frequent updates require frequent interrupts that impose latency. Splitting of bus bandwidth between the DMA and the main application can create a data bottleneck for the host application, as well as the DMA process.

MESSAGE PASSING VIA SHARED (GLOBAL) MEMORY

A third configuration would be for two or more computers to share a single set of global memory as illustrated. A typical shared global memory scenario would be two or more computers residing in the same backplane-based chassis (usually VMEbus or CompactPCI). Each of these computers would have their own local memory where accesses occur at the full speed of the processor. The computers could then communicate and share data with each other via a global memory set resident in the same backplane by utilizing a pre-established message protocol scheme.

In this type of system, the global memory is basically a single-ported memory shared among several computers and, while it may be accessible to all computers residing within the same chassis, access to this resource must be arbitrated. Also, inter-processor communications occur at the speed of the bus memory card combination, which is typically much slower than accessing local memory. The individual computers end up competing for the one scarce resource that facilitates the sharing of information and even when a processor has free access to the shared memory, it is at a lowered speed.

The communication becomes more cumbersome when externally distributed computers are connected into the single-ported global memory via repeaters, DMAs, or LANs. The total data latency may become compounded as each processor must wait its turn to access the memory (both in writing in new data and in receiving messages from other computers via the global memory). In this scenario data latency (which can be broadly defined as the time it takes before all computers can gain access to new data) can quickly spiral out of control.

TRADITIONAL LOCAL AREA NETWORKS (LANS)

The most familiar method of sharing data between computers is to rely on conventional networking standards such as 10/100 or Gigabit Ethernet. With this approach, the computers may be physically separated and connected via a network, and a common database is maintained from data sent through that standard network. This allows for wider connectivity and a more standardized communications approach, but adds considerable overhead for data transmissions. Also, because of Ethernet's arbitration schemes, determinism (the ability to define a time window in which communication will become available at a specific place on the network) is lost.

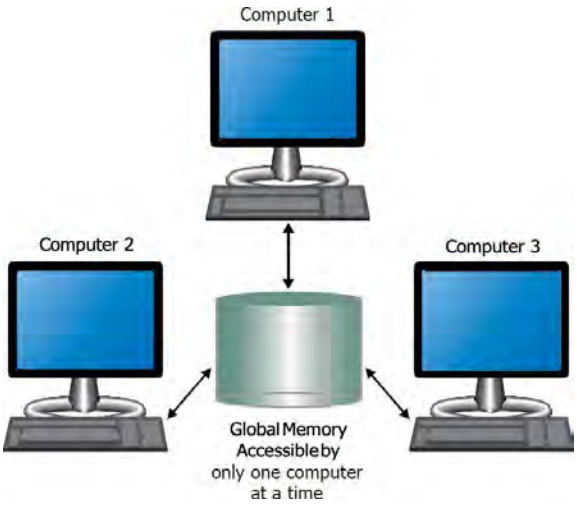

Figure 5 Global Shared Memory Architecture

The communication overhead of a LAN protocol like Ethernet adds another layer of complexity while decreasing the usable data payload. Once a system grows beyond a few nodes, that overhead can outweigh the advantage provided by the shared memory scheme. Like the other examples, this is still a single-ported memory approach and only one node may update the database at any one time. While LAN technologies enable developers to distribute their systems, they do not address the bottleneck of accessing the single-ported memory, which is still essentially an arbitrated resource.

Gigabit ETHERNET EXAMPLE

The following example shows the process required to share data between two computers. These steps would hold true for regular Ethernet, as well as Gigabit Ethernet LANs. In this example, Computer A collects raw data samples from ten different types of sensors. With 20 sensors of each type, there are a total of 200 sensors. This data is stored in computer A's own memory, then transferred to computer B for processing and display via a Graphical User Interface (GUI).

1. Computer A collects the data for each sensor type at different intervals; therefore, it does not send a fixed format data stream of all the data since this is too inefficient. Instead, Computer A sends the data to Computer B by sensor type. To accomplish this, Computer A must include the sensor type and number (1-20) with the sensor's data so that Computer B knows how to process the incoming data.

2. Between these two computers, there must be an application that encodes and decodes these sensor type/number/data messages. It is clear that Computer A would have to know how to encode ten different types of messages, one for each type of sensor's data, and Computer B would have to know how to decode ten different messages. Computer B would then act on the contents of those messages.

3. Computer A, after constructing a sensor type/number/data message, must transmit that message to Computer B. It does this by relying on the network hardware and the hardware's driver software. Computer A passes these constructed messages to the network adapter. The network adapter then reformats this message into data packets for transmitting through the network. The adapter hardware has to add information such as routing addresses, error checking information, and other networking protocols so that the receiving hardware interface on Computer B gets the information and can check its validity.

4. Upon receiving the information, the network adapter hardware in Computer B reads and interprets the data packets to verify that the packets arrived intact and error free. The hardware adapter then notifies the computer that the transmission data is ready to be placed in memory. Computer B then decodes this type/number/data message constructed by Computer A. Computer B must decode this message to separate the sensor type and sensor number from the actual sensor's data.

5. Computer B determines the sensor type and number through various case or case-like statements to determine which particular sensor type this message contains, and it must also determine which of the 20 sensors the data came from. After this information is extracted, then and only then, can the actual data originating from Computer A be written into the memory of Computer B so that the actual processing of this data may begin.

The same example implemented with Reflective Memory:

1. Computer A places the raw data from each of the sensors into the memory on its Reflective Memory board. Each sensor has its own unique address within the memory.

2. Reflective Memory automatically replicates this data to Computer B's Reflective Memory board.

3. Computer B now has the data available in its local memory, and may begin processing this data.

In summary, standard LANs have several shortcomings when real-time communication is required:

» Transfer rates are low.

» Data latency is hard to predict and is typically too large for real-time distributed multiprocessing systems.

» Layered protocol software consumes too much valuable processor time.

Why CHOOSE REFLECTIVE MEMORY?

SUMMARY

Reflective Memory is an optimal way to share data in time-critical applications ranging from data acquisition and process control to advanced simulation. Reflective Memory networks provide a real-time networking capability that surpasses most communications technologies for low latency and deterministic performance. Reflective Memory networks connect systems with minimal update delays and no access restrictions, to enable multiple, remotely located nodes to share a single data set in real time. The following table summarizes key technology characteristics by media type.

Reflective Memory Network Characteristics	5565/ 5565RC 5565PIORC	10/100 Ethernet	Gigabit Ethernet
Transmission Speed	2.1 GBaud/s	10/100 Mbit/s	1000 Mbit/s
Data Transfer Speed	170 MB/s	1/10 MB/s	100 MB/s
Endian Data Conversion	Yes	No	No
Software Transparent	Yes	No	No
Media	Fiber Optic	Coax, UTP	Fiber Optic
Topology	Ring	Ring, Hub	Ring
Network Data Transmission/ Reception Is Deterministic?	Yes	No	No
Network Transfer Scheme	Data Insertion	Carrier Sense Multiple Access/ Collision Detect	Token Passing
Memory Mapped Access to Shared Data?	Yes	No - Messaging Application Application Must Be Built	No - Messaging Application Must Be Built
Application Must Be Constructed to Share Data?	No	Yes - Messaging Application	Yes - Messaging Application
Application Must Encode/Decode Messages?	No	Yes	Yes
Application Must Perform Error Check/Handling Retransmits, etc.?	No	Yes	Yes
CPU Overhead to Support Shared Data Functionality?	No	Yes	Yes
CPU Overhead Required at Transmission Hardware Interface?	No	Yes	Yes

ESL Technology INC
TESTING YOUR LIMITS

TEST STANDARDS:

MIL-STD-167	MIL-STD-461G
MIL-STD-810	60068-2
MIL-STD-108E	60529
MIL-E-5400T	60945
MIL-STD-2164	60598-2-3
MIL-S-901D	

OCTAGON SYSTEMS

- Small Form Factor Rugged Computers/Mission Computers
- Ruggedized Servers
- Ruggedized Switches
- Ruggedized Displays

J SQUARED Reflective Memory

Toll Free: 1.855.365.2188
Local: 1.613.592.9540

Email: rfm@jsquared.com
reflectivememory.com