Video Applications Development Platform

Robert Thibadeau, Robert Berger, David Touretzky, Don Lindsay

March, 1993
CMU-RI-TR-93-04

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This research was sponsored by the Defense Advanced Research Projects Agency (DOD) ARPA order 6873, under contract #MDA972-92-J-1010. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

TABLE OF CONTENTS

List of Figures

ABSTRACT

The objective of the video applications development platform research is to develop the technical basis for Intelligent Frame Buffers useful in high definition displays. This project has three components: (1) **Physical implementation of the Intelligent Frame Buffer (IFB)** which includes multiple cooperating CPUs on a single display. (2) **Software operating system support** which forms the basis for display sharing and display cooperation. (3) **Applications-level software support** which forms the basis for programming the intelligent frame buffer in such applications as intelligent sensor fusion. The focus of the work is on *display sharing* by multiple processors and *intelligent autonomous display manipulation*. The IFB project operates in a domain that minimizes the need for explicit user input as opposed to domains which involve pure user manipulation, e.g. "smart controls," such as 'fly by wire' systems which have no need for display, or 'user monitoring.' This report describes the results of the first year of a multi-year project, namely: (1) architectural design and hardware prototype of regular resolution IFB; (2) Survey of work in realtime operating systems for imagery and MACH, resulting in an architectural design of real-time MACH kernel for imagery; (3) Prototype regular resolution workstation integrating realtime MACH kernel, IFB, and Advanced Video Display Systems (AVDS) applications building environments; and (4) Designs, prototypes, and systems architecture design for high resolution IFB, MACH kernel and applications building environment with networking.

Video Applications Development Platform

Robert Thibadeau, Robert Berger, David Touretzky, Don Lindsay
Imaging Systems Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh Pennsylvania

1. Summary: DARPA Report March 1993

The objective of the video applications development platform research is to develop the technical basis for Intelligent Frame Buffers useful in high definition displays. This project has three components:

(1) **Physical implementation of the Intelligent Frame Buffer (IFB)** which includes multiple cooperating CPUs on a single display.

(2) **Software operating system support** which forms the basis for display sharing and display cooperation.

(3) **Applications-level software support** which forms the basis for programming the intelligent frame buffer in such applications as intelligent sensor fusion.

The focus of the work is on *display sharing* by multiple processors and *intelligent autonomous display manipulation*. In the domain description below we show the IFB project domain as one that minimizes the need for explicit user input. At the other end, pure user manipulation, is "smart controls," such as 'fly by wire' systems which have no need for display, or 'user monitoring.'

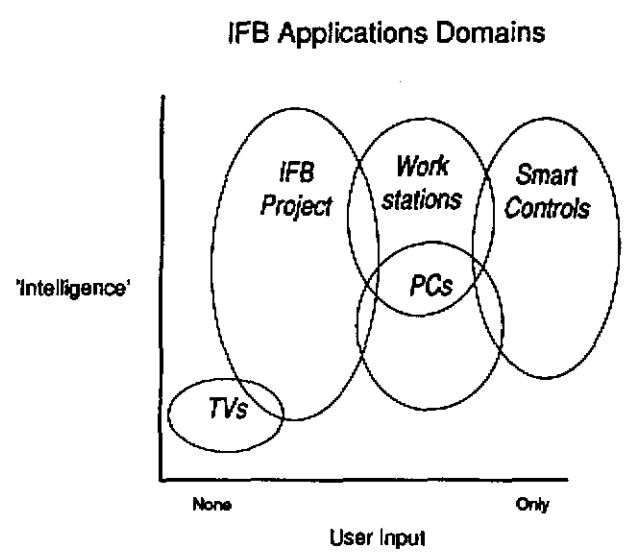


Figure 1.1. IFB Applications Domains.

The project will meet its first year deliverables of (1) architectural design and hardware prototype of regular resolution IFB; (2) Survey of work in realtime operating systems for imagery and MACH, resulting in an architectural design of real-time MACH kernel for imagery; (3) Prototype regular resolution workstation integrating realtime MACH kernel, IFB, and Advanced Video Display Ssytems (AVDS) applications

building environments; and (4) Designs, prototypes, and systems architecture design for high resolution IFB, MACH kernel and applications building environment with networking.

2. Introduction to a *Very High Aggregate Bandwidth Display*

The question of interest is whether a single display control architecture is sufficient to handle all problems in very high aggregate bandwidth displays. In the IFB, the display data is distributed or bussed across processing nodes (PNs) and each PN may modify data on the display bus. All our discussion distinguishes between *architecture* and *implementation*. The goal of our research is a statement of architecture which can form the basis for a unified theory of display control. *Implementations* are architectural experiments, committed to hardware and software, where it is possible to directly study performance implications of the various architectural decisions.

Computer architecture over the last decade has focussed on local intelligence and high performance computing with a result that display technology has suffered in certain significant ways. ***It is common nowadays to have several displays per user*** where each display constitutes a significant fraction of the cost of the workstation. A display for a 486 workstation will routinely cost 33% of the entire workstation cost.

Equally undesirable is when there is only one display location with sufficient display bandwidth to exercise the capabilities of a particular supercomputing resource. Methods for smoothly combining processing resources on a single display are largely concentrated on local area network models which require that ***display information and other, transactional, information compete for bandwidth***. The Intelligent Frame Buffer is a first attempt to study the viability of decoupling display information from the other information sources in order to, in the case of workstation profusion, lessen the number of expensive displays, and, in the other case of supercomputing centralization, make use of more, less expensive, displays.

Several other computing scenarios also contributes to the interest. With recent interest in 'multimedia' there are time critical display requirements such as ***real-time video***. Modern advancements in algorithms and computer speeds have made possible real-time video in a window[1, 2]. Various initiatives are under way to investigate operating system enhancements which permit scheduling real-time synchronization[3]. In the not to distant future, workstations will support network video in the form of one or two video streams on a workstation. However, the expense of expanding on the number of live video streams is great since considerable computational resources are required. Supposing that a video stream requires about 20% of the capacity of a modern workstation, and that workstations double in speed/performance every 2.5 years, it will require five generations, or 12 years to reach the point of having up to 32 live video sources in manipulable windows on a screen. The IFB display technology permits ***many live video sources in resizable windows*** in a single generation.

Another basis for exploring display-only busses is the emergence of very high resolution displays. Some of these may be made up of ***tiled display units***. Circuits which can directly drive displays of tens or hundreds of millions of pixels and which nevertheless permit flexible programming, such as windowing and scaling systems, are beyond the reach of monolithic driver methods. The IFB controller architecture is suitable to such tiling demands.

Multimodal presentation such as ***simultaneous flexible stereo visualization*** is another capability of such an architecture. In this situation, resizable windows may variously contain stereo views which shift to monocular views as the viewing goggles are removed or a person stands away from the "sweet spot." One of our goals is to produce a single display controller architecture suitable for various classes of stereo display [4]. The coordination of sight and sound on an IFB should also be straightforward.

Finally, the goal of the display architecture is to permit the definition of ***cooperative processes among disparate computations***. We could envision a view on a football field on a screen. Automatic image processing takes place to recognize, in real time, individual players, and a graphics processor is employed to overlay a graphical rendering. In at least one fictional movie, *Looker*, we have seen real time video, real time recognition, and real time rendering combined with effect. However, the more practical applications may be in confirmational vision systems which generate views of what is seen in order for the person or

machine *to readily confirm the degree of confidence in the automatic recognition* of what is seen.

The reasons for a separate display-data bus architecture are then:

- Display (or 'computer head') reduction in multiple workstation evironments.
- Display multiplication in singular high performance environments.
- Display reduction and flexibility for high bandwidth real-time data.
- Display tiling.
- Multimodal Processing for Visualization (e.g., simultaneous stereo).
- Cooperative Processing for Visualization.

Very high aggregate bandwidths are required of all these functions. For example, 32 live video sources *arbitrarily distributed* on a 64 Mhz reference display involves an aggregate bandwidth in excess of 64 * 32 * 15 or 30,720 Mhz aggregate. If there are 32 tiled NTSC television displays for a 64 Mhz display signal we get the same aggregate bandwidth requirement. Either specific case can be handled individually by special case hardware solutions, but it is interesting to explore a single architecture which can handle either and all requirements.

3 IFB Ring

3.1 Implementation for Year 1

The IFB display bus takes the form of a ring-pipe which is synchronized to the display replacement rate. By definition, each cycle of data on the IFB ring-pipe is one replacement cycle (e..g., refresh) of the reference display.

The **IFB Ring Bus** is the first implementation of the architecture being built at Carnegie Mellon that allows any number of computers, video sources, or other devices to share a single display. In this implementation, each display source has full bandwidth access to the display, as if the display were dedicated to that device.

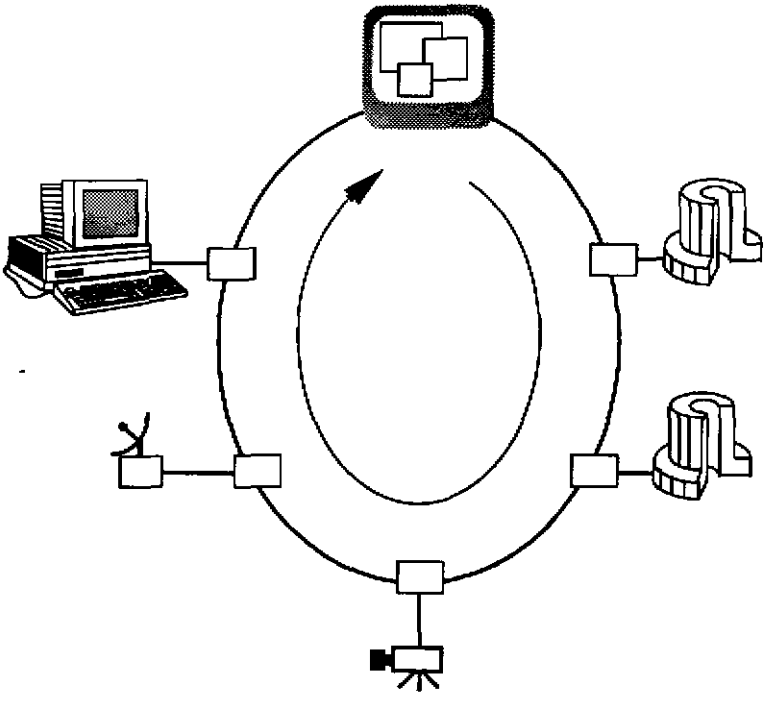


Figure 3.1. Architecture of the **IFB Ring Bus.**

Figure 3.1 shows the configuration of the IFB Ring Bus. The topology resembles that of a token ring network. However, the data that circulates on the IFB Ring Bus is synchronized to the horizontal and vertical refresh rates of the display. During each frame time the display controller at the top of Figure 1 transmits a blank frame consisting of a 32 bit word for each pixel. Each device on the ring changes some or all of the pixels in the received frame and transmits the resulting frame to the next device in the ring. When the frame has circulated around to the display controller it is displayed on the monitor. Each device has full video bandwidth to the display. Any ***arbitrary*** subset of the pixels can be changed by a device during every frame time.

Each pixel in the frame is represented by four bytes: red, green, blue, and depth. The depth byte allows each pixel from a device to be assigned to any one of 256 planes. Hardware distributed among all the ring nodes uses the depth bytes to determine in real time which node provides the displayed value for each pixel. This depth plane resolution hardware can be used to efficiently provide:

° Overlapping of windows from multiple nodes.
° Cursors of arbitrary size and shape.
° Graphics overlays for live video sources.
° Distributed animation systems.
° Stereographic 3D displays.

For example, a stereographic display can be produced by using two ring bus nodes, one each for the left and right images. After each displayed frame, switching between the two images can be accomplished by changing a single byte in one of the two frame buffer controllers, thereby changing the depth plane of that controller's image.

Figure 3.2 shows the format of the display data on the ring bus. For each active horizontal line of a display,

a packet of data circulates around the ring. This packet contains a 32 bit header and the 32 bit pixel values for the line. The header includes a type field which distinguishes the displayed data from other types of data which may be passed on the ring. The display packets are synchronized to the horizontal rate of the display.

The display data does not use all of the ring bus bandwidth. If the ring clock is at the pixel rate, a typical display format will have about 25% of the bandwidth unused due to the horizontal and vertical blanking times of the video signal. It is also possible to run the ring at a faster clock rate than the pixel rate, producing even more extra bandwidth. This extra bandwidth can be used for sending many types of non-displayed data, such as multichannel sound, inter-node synchronization information, etc. One type of non-display information implemented in the prototype ring bus architecture is a frame ID that is sent immediately after the last line in each frame. The frame ID facilitates synchronization in applications where multiple nodes cooperate to produce a dynamic display, such as animation systems or stereographic displays.

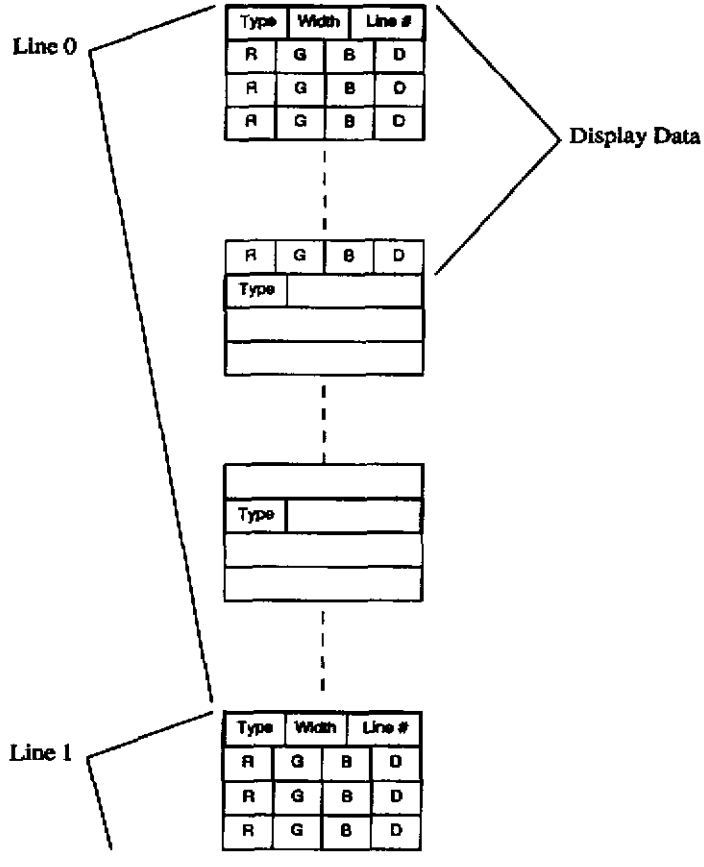


Figure 3.2. Format of data on the ring.

The hardware architecture of the display controller is shown in **Figure 3.3**. The display controller is split into two independent halves. The left half of the diagram is the initiator section, which puts blank frames onto the ring at the vertical refresh rate of the display. These frames consist of a constant background color with a depth plane of 0.

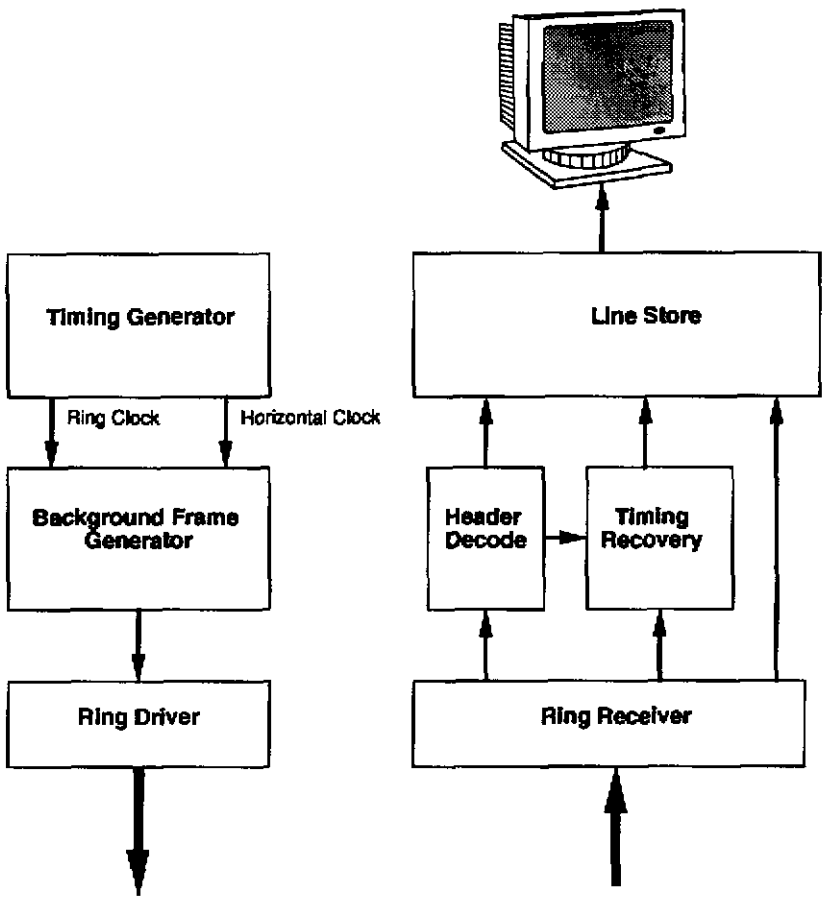


Figure 3.3. Architecture of the display controller.

The right half of **Figure 3.3** accepts completed frames from the ring and formats them into an analog video signal for output to a monitor. The ring bus clock and decoded horizontal header information are used to generate the timing signals for the monitor. The display circuitry is totally independent of the initiator, deriving its timing solely from the information coming from the ring. This allows the nodes on the ring to implement any constant delay they need for pipelining or memory latency. The display controller is not affected by such delays. The only constraint on the amount of delay a node may introduce is imposed by the interactive nature of certain applications. For example, if a node is implementing a mouse based user interface in a system where the total delay of all the nodes exceeds a frame time, it would be best to place the user interface node late in the ring to avoid perceivable delays in mouse tracking.

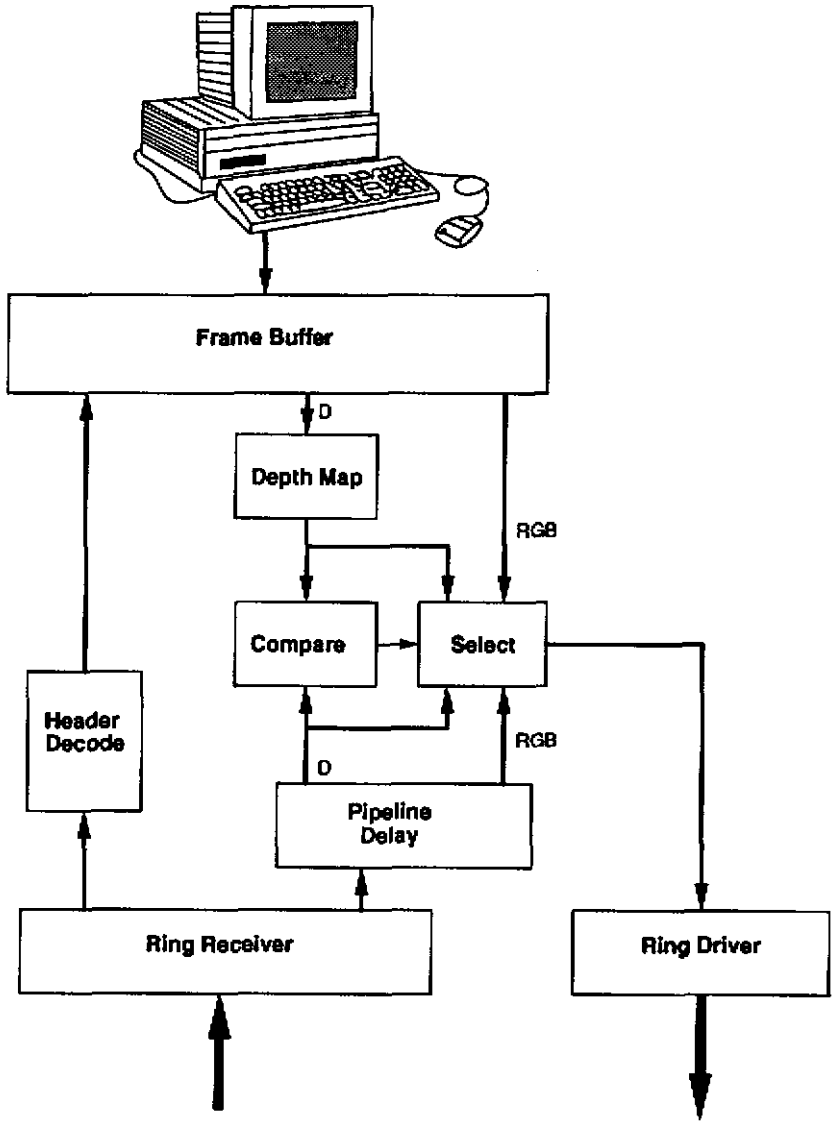


Figure 3.4. A frame buffer node.

There are many possible ring node designs depending on the desired application. **Figure 3.4** shows a possible architecture of a node designed to interface a general purpose computer to a ring bus display. As each line of displayed data is received from the ring bus, the corresponding line of pixels is read from a local frame buffer. The depth value for each pixel in the local frame buffer is used as an index into a lookup table called a **depth map**. The value from the depth map is compared to the depth value of the pixel received from the ring. If the local depth value is greater than the received depth value, the local pixel data is output to the ring; otherwise the input pixel data is output. A pipeline delay in the received pixel data equal to the access time of the frame buffer and depth map insures that the correct pixels are compared.

After the displayed data has been through all the nodes on the ring, each pixel will contain the color of the corresponding pixel from the node that had the highest depth plane for that pixel location. In this way the hardware distributed throughout the ring performs a priority selection process to merge the data from the multiple display sources in real time.

The depth map allows an arbitrarily shaped set of pixels to share a common depth value. For example, all the pixels in a graphics window, cursor, or other displayable object can be assigned a constant depth map index. The depth of the entire object can then be changed by changing a single byte in the depth map. The ability to change the depth of a large object by changing a single value facilitates dynamic applications such as stereographic displays and animations.

The operation of the display is generally independent of the order in which the devices appear in the ring. One exception is the interactive latency issue mentioned above. Another dependancy arises if two different nodes use the same depth value for a given pixel. The result of such a tie depends on the order in which the two nodes appear in the ring; the displayed result will be from the node closest to the initiator.

The prototype ring bus display controller is designed to drive a VGA standard monitor, which displays 640x480 pixel frames at a 60 Hz non-interlaced rate. In the prototype, the ring clock is the same as the pixel rate of 25.175 Mhz, eliminating the need for line buffering in the display controller. Running the ring at the exact pixel rate also eliminates the need for a phased lock loop in the display controller's timing recovery section, avoiding a potential source of display jitter.

The physical layer of the prototype ring consists of 34 differential ECL pairs. 32 pairs are used for the pixel data. One pair is used for the clock. The remaining ECL pair is a flag that identifies the header word of each packet. The total bandwidth of the ring is 800 megabits per second, of which 600 megabits are used for active display data, leaving 200 megabits for non-displayed data.

The configuration has Phase I and Phase II levels of hardware introduction. In Phase I any of several IFB Ring node controllers on separate (or the same) 486 workstation may be run. Enough Master and Node controllers are being fabricated to provide working hardware to any and all interested parties. In Phase II the node hardware will be augmented to receive and inject live NTSC resolution video with dynamic scaling and, of course, window placement. With the Phase II system it will be practical to achieve many live video views in dynamically resizable windows on a single display.

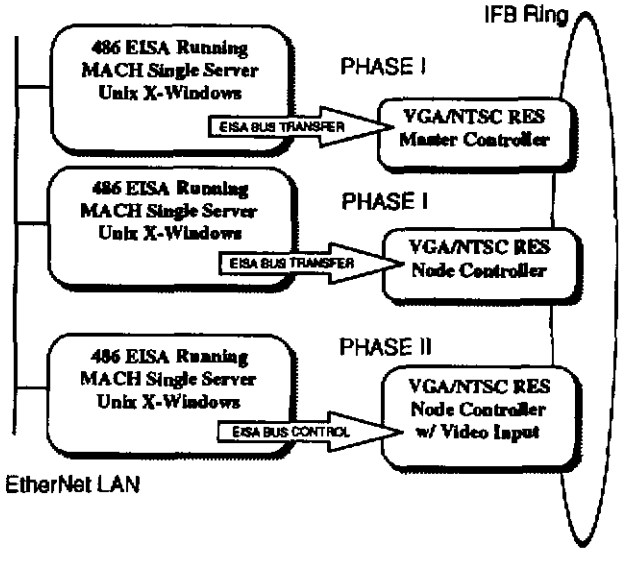


Figure 3.5. Phases for Introduction of IFB Hardware.

3.2 IFB Ring Architectural Study

3.2.1 Comparisons

In large measure the present work is unique. The lack of attention to distributed display control architectures originates because displays bandwidths are high enough, and cost considerations are such, that tight coupling contraints are preferred. In the present view, the bandwidth and cost considerations are mitigated by the gain in display efficiency for delivering *high aggregate bandwidth.*

3.2.1.1 Bus Architectures.

Most existing bus architectures are inapplicable to the IFB display bus since they were not designed to keep up and synchronize with the demands of display control and display sharing. Bi-directional busses, in general, represent an, at best, inefficient, solution even if employed in an entirely dedicated fashion. For example, the *FutureBus+* specification does offer priority and high bandwidth but does not scale and is only high-bandwidth for multiword transfers. It is inefficient for overlaying pixel priorities (viz., window management).

The Scaleable Coherent Interface (SCI) bus certainly comes closest to the IFB display bus [5, 6, 7]. This bus is similar in many respects, including the use of singular ring-pipe data flow and high bandwidth unidirectional communication. However, the SCI bus has been designed to solve cache coherency problems in multiprocessors and, as such, retains a notable lack of focus on display control. In particular it does not fix volume of traffic, nature of traffic, and pipeline delay. We are currently monitoring SCI work closely since it seems probable that it will represent a source of IC parts for the IFB controller operating at the higher speeds necessary to directly support 1M pixel displays.

3.2.1.2 Network Architectures.

The IFB can be considered an implementation of a Constant Bit Rate (CBR) network and, as such, needs to be understood in terms of the recent explosion of interest in mixed CBR and Variable Bit Rate (VBR) networks such as Asynchronous Transfer Mode (ATM) networks [8]. The physical isolation of a CBR network from the VBR network is generally considered uneconomical for wide area networks and we agree. The present argument is that a pure CBR network (the IFB) simply has certain distinct advantages in achieving high aggregate bandwidths.

Nevertheless, the work on ATM networks must be considered deeply. The preference in achieving point-point or point-multipoint contacts is to reduce data jitter in such bursty networks by specifying an intercell or interpacket spacing. It is not our job (yet) to make specific implementation *proposals*, but the focus on a CBR-only network should not prohibit the use of mixed CBR/VBR network implementations with CBR-only data. Indeed, a nearly ideal situation would be an IFB node which can act, at low cost, as a *gateway* to and from an ATM, or similar virtual circuit, CBR capable, network.

The ATM specification with its small, fixed, 53 byte cell size and relatively high, 46 byte, user density, is similar, in many regards to the present focus on small, fixed size, low overhead, cells in the ring pipe. However, the ATM is considerably slower, at a current peak 150mb/sec. Even SONET, at 1.2gb/sec does not achieve the practical levels achievable with a small area display bus such as the IFB. Our focus in achieving high aggregate display bandwidths is not fundamentally at odds with modern "multimedia network" proposals. But it should be clearly understood that those proposals do not address the problems of high aggregate display bandwidth very effectively.

3.2.2 Applicability Considerations.

The IFB Ring, in its current implementation, provides an obvious solution for each of the following four application domains:

° Display (or "computer head") reduction in multiple workstation environments.

° Display multiplication in singular high performance environments.

° Display reduction and flexibility for high bandwidth real-time data.

° Multimodal Processing for Visualization (e.g., simultaneous stereo).

The current implementation does not provide ready solutions for the remaining two application domains:

° Display tiling.

° Cooperative Processing for Visualization.

We now consider enhancements necessary to extend the domain of application.

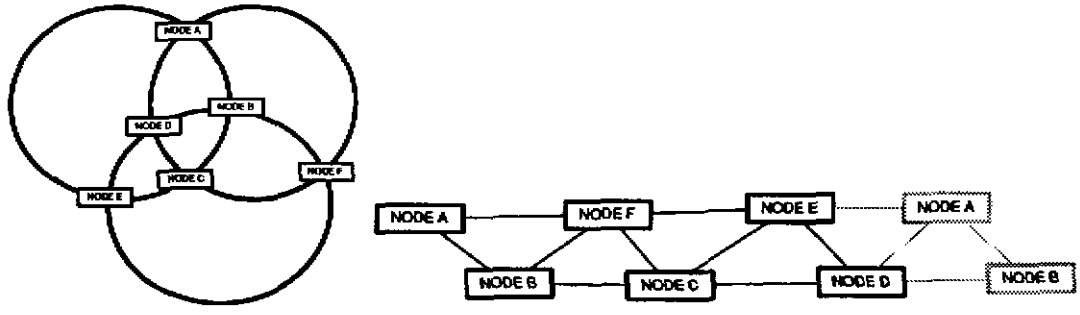
3.2.2.1 Display Tiling

The display tiling problem is relatively simple as long as actual resolution remains within bounds of the bandwidth of the IFB ring. Architectural extensions are required to supply higher real bandwidth than a single ring can deliver in current technology.

In an NXM tiling of displays, each of several PNs controls a display tile and each PN has direct access to the NXM source data. The IFB, in this case, is employed to establish synchronization of the source data to the NXM display. In the simplest case, it is an absolute clock. If the NXM display is simply a blow up of a display, the IFB would be defined as pipelining the single-display data to Modified PNs which individually control tiles. Pipeline delay on raster supplied data is proportional to the number of lines which one pixel in input makes in NXM pixels in output. The display latencies will be distributed and extreme: this will require buffering several display replacement cycles (or, several pixels up to exactly one display replacement cycle). The Modified PNs will have to buffer display pixels.

One possible means of minimizing buffering while retaining synchronization is suggested by the scheme below which supports six display tiles (Modified PNs) with three clockwise moving rings. Each node operates as described above for the single ring but, in addition, can switch data (2 X 2 crossbar + 2->1 output for display) to an alternative ring *depending on fixed programmable display addressing*. This scheme generalizes well and is comparable to CBR ATM architectural design based on both the switching capacity of the node and number of tiles (nodes). The fixed addressing provides for fixed, solvable, prioritization and latency prediction. This method distributes ring bandwidth load by multiplexing. In the instance illustrated below we can prove that each node receives data as a balanced binary tree and each PN therefore receives its display data from 1/2 the capacity and configuration of two input rings and the remaining half capacity is used to relay data to the appropriate ring. For example, if Node A injects data for display on Node D, Node B must relay that data and Node C must switch it.

Figure 3.6 Three Rings and six nodes for a six node tiling. Diagram on right is logically equivalent but may help in understanding the discussion in the text.



With this scheme, a given node processor can generate more display data than a single other node (including itself) may accept. For example, it may generate a window of image data which involves more than one other node for display. The physical IFB ring has two channels and therefore each channel can run at one half the real bandwidth of the entire 3X3 display. However, there must be explicit pixel addressing since pixel location is the basis for the 2X2+2->1 switch in each node. The addition of three other rings to two nodes each (A-C, F-D, B-E) reduces channel bandwidth loading to 1/3 total but requires a 3 X 3+3->1 switch at each node. The advantage of this six ring case over the three ring case is that routing, and therefore latency, is now reduced by half. These are all well understood concepts in switching theory. The point of this analysis is not to review switching theory but to note that such a structure permits complete, high resolution, display control by a single node or any other configuration programmed. To the extent that CBR ATMs can be dedicated to display control, the basic ATM architecture may be applicable to the IFB although the SCI bus, as mentioned earlier, may provide all the control required using its provision for joined rings.

3.2.2.2 Cooperative Processing

Cooperative processing involves the case where a node may modify, not simply replace, the contents of a display value. This puts constraint on either (a) the speed of the electronics in the node, or (b) the depth of the (effective) FIFO or both. Generally speaking display control pushes speed to the limit, although tiling schemes for tens or even hundreds of low bandwidth individual displays as outlined in the last section may be used to stay well below any limits. In any event the depth of the FIFO is likely to be larger and therefore the need to explicitly make provision for latencies is important. The simplification which the IFB provides is that the latency for each node to display is known and fixed. Therefore software and hardware can be prepared for such latencies.

4. Operating Systems Software

The bandwidth requirements on the IFB ring-pipe depend on how data on the ring-pipe is mapped to the display. A PN, such as a workstation or video injector, has access to the IFB by defining a FIFO buffer of a specified depth in display ticks. This introduces a pipeline delay for each PN of at least one tick but potentially considerably more. A special necessary PN is the single master controller. This may, or may not, be directly feeding data to a display. A given IFB has a *fixed pipeline delay* which implies that data to the display is always delivered at the full replacement rate of the display. The *pipeline delay* is fixed so that different PNs by design or by programming can anticipate proper display actions. In other words, in a synchronous display bus there is an opportunity for implementation software to synchronize on display. In the *architectural definition*, we anticipate the use of display control data ("invisible pixels") which communicate synchronization parameters to software. The pipeline delay structure can be measured in display ticks (the equivalent of display clock cycles) or whole replacement cycles. The longer the pipeline delay the less "interactive" the display although it seems clear that little is sacrificed to a display which is

only a few display cycles (e.g., 100 msec.) delayed from external input control since the architecture delivers "smooth" control as opposed to "jittery" control. In pure visualization tasks which lack all but "button press" control, the pipeline delays could reasonably be counted in many display cycles (e.g., as much as a few seconds).

The fixed pipeline delay associated with the IFB architecture is associated with the latency introduced by Dannenberg[9] in his client-server architecture for real-time operating system control. A *Tactus Server is* defined as a real-time server with a fixed latency registered to all clients or the operating system scheduler. This server constitutes an essential ingredient of the IFB architecture.

The basic purpose of the effort in MACH operating system development is:

Getting very fast and smooth interaction between humans, and a a display which can be extremely active but not jittery.

A way to scale the image display and image transformation power of a system, smoothly and incrementally, from low to high end.

A way to get diverse equipment and data under the same human interface.

Development of an appropriate *Tactus Server for each IFB node which is able to add data to the display (or modify existing display data)* is now assumed to be the major requirement.

4.1 Evaluation of Real Time Mach

The Mach 3.0 microkernel has been developed to include various real time features by ourselves and others [10] including OSF [11]. The 'real time' features include

- higher resolution timing services including an absolute clock
- privileged programs able to wire pages down
- privileged operation to set scheduling policy
- resource reclamation after wired-down processes die
- lock reclamation after threads die
- threads may tell the system their deadlines, and say who is to be informed if a deadline is missed
- a mutex form offering priority inheritance
- Dannenberg's Tactus server which tells clients to expect fixed latencies and that establishes threads with tightly known deadlines

The Tactus Server is a Mach 3.0 multithreaded message server. During this period, we completed a complete Mach 3.0 device driver for a video board which can act as such a message server.

In one experiment, we established that 'tightening down MACH' for real time performance will still not remove jitter problems on the display. Even with these modification, experiments revealed the limitations of existing display technology. MACH 3.0 with X-windows is fast enough that it can play uncompressed color video from disk, thru X-windows, to a 160 x 120 window at 10 frames/second (on a 33 MHz 486, IDE disk). It can do 15 f/s, but the jitter, even at this low screen resolution, is now substantial. The removal of jitter problems in a large range of desirable workstation and high resolution view-only display behaviors requires a coupling of real time MACH enchancements with the cabability of writing high bandwidth data directly to the display.

An interesting property of the Tactus Server running on a PN which may also act as a gateway to an ATM network is that it can declare its latency based on both its configuration in the IFB ring pipe and the demand it places for CBR data on the ATM network. The result is a consistent and predictable CBR with a given latency to the display without the arbitration of a CPU which may be running out of cycles to maintain its real-time behavior.

5. Applications Layer Software

Very high aggregate bandwidth displays will be capable of delivering substantial information to the person watching the display. In order to provide a working environment for the video applications development platform, we have investigated intelligent filtering as client processes in the MACH environment. The objective was to demonstrate that this type of display has compelling practical applications. In the IFB, these applications may all be thought of as clients of Tactus Servers. Our goal was to understand how these client applications may make demands on these display servers. For this purpose we are prototyping intelligent agents which may simply be called on to monitor information traffic otherwise available to the system and provide appropriate visual display information to the user when, where, and how he needs it.

5.1 Television Computer Experiment

The first experiment involved consumer uses of a "television computer" that monitored a variety of information streams and extracted information relevant to the user. These information streams included constantly-updated public safety advisories (tornado warnings, chemical spills, etc.), a variety of classified advertising streams, resource catalogs (e.g., restaurant guides), and event scheduling (programming schedules for both television shows and the information stream itself.)

We developed a simple, menu-based user interface designed to be immediately usable by people with no computer training. We also explored various styles of interacting with the device as an information filter. For example, when browsing the restaurant guide, there were various ways to restrict the search (e.g., by cuisine, location, hours of operation, etc.), and the current restrictions were always displayed in a "search header" at the top of the listing. Classified ads could also be browsed, but a more common access strategy would be to set up a "clipping service" agent to continually clip ads that meet certain specifications. The user could then check periodically, at his convenience, what the intelligent agent had clipped for him. The control of clipping requests was via the same menu-based interface used for browsing data and communicating with agents.

At the opposite end of the spectrum from the clipping service, the application that handles emergency advisories would grab control of the display on its own initiative, perhaps even turning the display on if it had been powered down, and beep to attract the user's attention. Users would be able to obtain progressive warnings about a threat (e.g., estimated time of arrival of a tornado approaching their location) on request; they could also disable further interruptions from the application concerning that threat.

5.2 Thousand Eyes Experiment

The second experiment was a futuristic "thousand eyes" battlefield scenario in which video cameras were dropped over a large area, each broadcasting (in 'spread spectrum' fashion) from its own position, with client applications selecting camera views dependent on the position of the individual viewer in a jeep or tank. The idea was to show the viewer anything ahead or nearby that might pose a threat or affect his mission, given knowledge of his present position and intended route. In addition to the ground cameras, we also assumed video transmissions from orbiting drone aircraft. Thus, there were a huge number of possibly relevant real-time video streams: far more information than one person could sift through manually. The client applications were responsible for making the first cut of what to display, subject to further modification by the user. They might display several video streams simultaneously, but not hundreds.

In creating this experiment, we used a scenario in which the city of Pittsburgh was under attack from the east. The "front" ran in a north-south line just west of the CMU campus. A military vehicle (an actual

HMWWV jeep borrowed from the Autonomous Land Vehicle group) was attempting to travel east on Forbes Avenue from the downtown area to CMU. This route had been seeded with "thousand eyes" video cameras every few blocks, all broadcasting simultaneously, which could see the vehicle as it passed by. In addition there were two drones in the air; one orbited the downtown area, while the other followed a north-south flight path to monitor the front. We used actual aerial footage of Pittsburgh shot from a small plane by one of the project members.

The display was designed to be useful to the operator of a vehicle traveling in the combat zone. There was a moving map display showing the vehicle's present position, the position of nearby ground video cameras and their fields of view, the ground tracks of the airborne drones, and the front line. The experiment ran on a NeXT machine with three attached VCRs. There were also three simultaneously active video windows. Two of these displayed aerial images from the drones. The third displayed an image from a ground camera. As the vehicle moved across the terrain, its position on the map changed, and the client software automatically selected a new ground camera video stream so that the operator could always monitor what was ahead on his intended route. The vehicle itself was visible briefly as it passed by the ground camera, before the view switched to a camera further ahead.

Due to the limitations of the NeXT Dimension board, it was not possible to display three real-time video sources simultaneously. Therefore we displayed real-time images from one source, and still frames from the other two, updated every five seconds. The user could choose which window ran in real-time by mousing on it. Of course, with a functioning IFB it would be possible to display all three streams in real-time, scale the images to fit in smaller windows, and so on.

A more mundane application utilizing multiple real-time video streams is video conferencing. Mantei et al. (1991) report on a prototype system in which each office is equipped with a video camera. The system displays a map of the environment area with an icon (in this case, a miniaturized photo) in each room depicting the owner of that office. Users could put together a video conference by mousing on icons and dragging them to their own office location; each participant then shows up as a separate video window on the display. Some of the tehcnical problems they report with lighting adaptation, audio level adjustment, focusing, and choice of camera angles could perhaps be addressed using signal processing and computer vision techniques. The IFB would make it feasible to experiment with real-time adaptive processing of multiple video streams.

5.3 Sensor Fusion Experiment

We constructed a sensor fusion experiment based on the same NeXT Dimension technology used in the previous experiment. The primary goal was to illustrate the advantages to be had from combining diverse data sources from multiple, independent processors into a single display presentation. These sources might include (1) maps; (2) diverse real-time imagery: video, radar, IR, etc.; (3) non-image sensor information, such as transponder codes or RF signatures; (4) graphic overlays; and (5) database queries.

The scenario for the experiment was a command and control application in which the operator must track a series of unknown targets and assess their threat potential. The format was modeled after the AEGIS system. The following paragraphs describe the layout of the display and the various options available.

5.3.1 Detailed Sensor Fusion Experiment Description

In this demo, the TACSIT (tactical situation) display is the largest window on the screen. Additional information windows appear along the sides. Some are associated with targets being tracked on the TACSIT; others are pop-up windows that appear in response to operator requests. There is also a row of icons along the top of the TACSIT, used for controlling the simulation.

The background for the TACSIT is a map of some portion of the world. Water is displayed in blue; land in yellow. Map data is provided by an independent map server process, which we created based on code from the GOTS Alternate Environment Map Server version 1.5.6.8 by Tiburon Systems. This code was obtain from Intermetrics as part of their CRSS (C3I Reusable Software System).

Mousing on the map control icon (a globe) causes a pop-up window to appear with a small map of the world and various map control buttons. The region currently being displayed on the TACSIT is shown by a white rectangle on the world map. The operator can shift the displayed region to anywhere on the globe, change the display scale, and add political boundaries if desired.

Also on the TACSIT display, superimposed on top of the map information, are target symbols. Target types include aircraft, missiles, surface vessels, and submarines. They may be classified as hostile, friendly, or unknown status. We use the AEGIS conventions for target shapes and colors. Hostile targets are shown in red; friendly ones in blue; unknown targets in green. Target information, including type, location, speed, and altitude, is supplied by a locally-written OTH (Over the Horizon) data server driven by radar track files. The track file data was obtained from GE in Moorestown, NJ. In some cases these files contain data from real events involving US warships. The main track file, called all.out, depicts an incident involving the shooting down of two Libyan Migs in the Gulf of Sidra by two US Navy F-14s.

Mousing on the display control icon summons a pop-up window with TACSIT display controls. Some of the available options include: display a compass rose, choose which types of targets will be shown (air, surface, or underwater), display heading vectors for selected targets, display cumulative track information for selected targets, and center the display over a selected (possibly moving) target.

Any target on the TACSIT display can be selected by mousing on it (called "hooking" in AEGIS terminology.) Hooking a target brings up a pop-up window along the left side of the TACSIT display which can be used to view several different types of data. It intially comes up as a radar target information display (menu item "INFORMATION"), showing the target's lat/long coordinates, course, and speed. Other menu items allow the operator to call up other sorts of information and/or image data, described below.

If the target type has been identified, e.g., an F-14, the DATABASE menu item calls up information about that type of target, such as maximum speed and altitude and known armament types. The PICTURE menu item displays a color photo of that aircraft type.

The RADAR IMAGE menu item displays a real-time radar image of the target. We are extrapolating here from recent reports in Aviation Week showing that radar technology can be used to image aircraft in flight. For the purposes of this experiment, we display a synthetic time-varying image based on the silhouette of a typical aircraft.

The VIDEO menu item, available only for friendly targets, displays real-time video image data transmitted from the target back to the AEGIS cruiser. Typically this would be the view from a video camera mounted next to the pilot's head, looking forward through the HUD (Heads-Up Display), so that the AEGIS operator can have a pilot's eye view of the situation.

A third icon located above the TACSIT display is the database access icon. It allows the user to retrieve information from various textual and image databases in a subwindow. One such database is a schedule of civilian airline flights whose route is predicted to cross the region on the TACSIT display. In several past incidents involving military warships, this information has been helpful in identifying an uknown target as a probable civilian flight.

Another database resource is digitized ONC (Operational Navigation Chart) images for the region on the TACSIT display. ONC charts show cities, terrain type and elevation, air navigation aids, and some types of airspace structure -- information not available from the map server. The user can scroll around on the chart to examine areas of interest.

5.3.2 Assessment of the Sensor Fusion Experiment

The sensor fusion experiment was intended to show the advantages an IFB would offer in command-and-control situations, namely, the ability to combine and display multiple high-bandwidth information streams. However, because of the limitations of the NeXT Dimension video board, we were not able to fully emulate

the functions an IFB would provide. Thus it was not possible to display, for example, simultaneous real-time pilot's eye views from multiple aircraft. The NeXT Dimension also did not permit us to scale images in real-time, e.g., to monitor a miniaturized version of the video stream in a tiny window while most of the display is used for something else. However, we can still show some of the effects of an IFB by simultaneously displaying video data in one window and synthetic radar imagery in another.

Due to speed limitations of the NeXT workstation, it also is not practical to scale or otherwise manipulate stored digital images such as the ONC charts, since this would take too long relative to the pace of the simulation. In a real IFB implementation, dedicated processors would handle these computations, so the operator would be able to manipulate a variety of image sources simultaneously.

A good example of an industrial application for this type of sensor fusion is the recent work of Tani et al. (1992) on video-mediated interaction with real-world objects. They use multiple video cameras to monitor machinery in a mock-up electric power plant installation. The machinery may either be fixed (e.g., pipes, boilers), or mobile, provided the nature of the motion is known to the system. Their "Object Oriented Video" system superimposes grapics on top of the video stream, and switches video streams as the operator changes his focus of attention. For example, pointing to a fuel pipe in a video window causes (a) the fuel pipe to be outlined in blue, (b) a small graphic to appear next to the pipe in the video window, indicating the current fuel flow rate, and (c) a plot of fuel flow through the pipe over time to appear in an associated graphics window. Pointing to a burner causes it to be highlighted by a blue rectange, and a pop-up window appears inside the video window with an explanation of the ignition procedure. Pointing to an observation port on a boiler unit causes it to be highlighted; zooming in then causes the system to switch to a different video source: a camera monitoring the flame inside the boiler.

Tani et al. do not attempt to display multiple video streams simultaneously. Nor do they devote much computation to analyzing the contents of the streams; their system must be pre-loaded with the locations of fixed objects in each video stream, and programmed to associate current positions of moving objects with particular locations in the camera's field of view. A system that could dynamically track objects would of course be more expensive to construct, but also much more flexible and useful. Dividing this work across multiple processors is feasible with an IFB.

5.4 Future Applications Experiments

We plan to further develop our sensor fusion demo into a more realistic demonstration of handling multiple real-time data streams. One idea we're investigating is using two adjacent displays, driven by separate NeXT machines, to provide a greater number of simultaneous image streams. Another idea is to use some NeXT Dimension programming tricks to simulate brief segments of real-time video in multiple small windows at the same time.

We plan to develop a customized radar "script" to replace the GE track files. Various targets will appear on the screen and the operator must react. Some will be harmless civilian targets, but others will be hostile and must be identified and dealt with before they come within striking range of the ship. This scripted approach will make for a more directed simulation and will make it easier to show off the sensor fusion issues which are the real point of the demonstration.

5.5 Applications Layer Architecture

The variety of applications scenarios provide for high aggregate display bandwidth which would normally be expected to dramatically degrade the performance of even the most powerful workstations. It seems clear that a single high resolution display invites display participation by quasi independent, high bandwidth, nodes. Figure 5.1 summarizes some of the high bandwidth sources which have been investigated for architectural considerations.

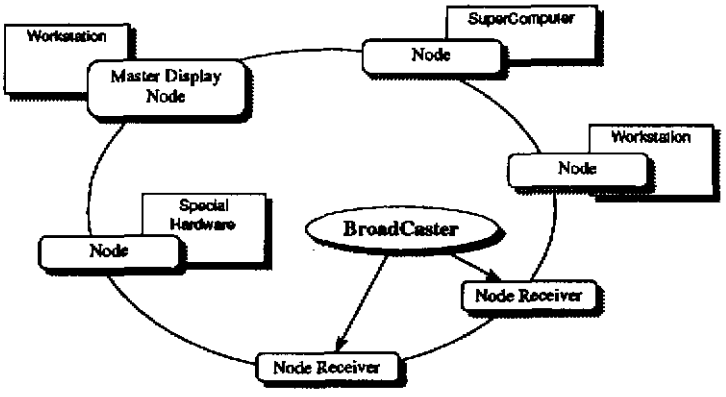


Figure 5.1 Example IFB Configuration

The central concept of an application client to a Tactus Server on an IFB node appears to be that the server can feed back display requirements to the client as server to the information source. This is illustrated in Figure 5.2.

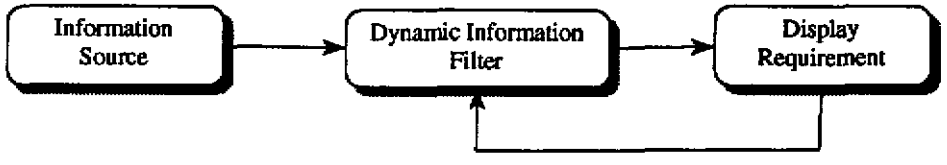


Figure 5.2 The Client Display Filter

If this successfully generalizes, then the Tactus Server will have an enhanced form. The Enhanced Tactus Server is envisioned to hide latency considerations from the information source and to handle display filtering (e.g., display scaling, information search) dependent on display requirements which may be dictated by the user(s) of the very high aggregate bandwidth display(s). A filter behaves like a registery. A simple filter is one that provides 'private' regions of any given display which are visible only to the local display. Furthermore, the attribute of 'window rights' coupled with the attributes of 'window priority' and 'local-only override' become the basic ingredients for a well behaved distributed display control system. [12, 13]

REFERENCES

1 **The Society of Motion Picture and Television Engineers** (1991) SMPTE Standard for Television - Component Videosignal 4:2:2: Bit-Parallel Digital Interface., ANSI/SMPTE 125M-1992 July 16, 1992

2 **Ang, Peng H., Ruetz, Peter A., and Auld, David** (1991) Video Compression Makes Big Gains., IEEE Spectrum, Vol. 28: No. 10, October 1991, pp. 16-19.

3 **Tokuda, H.; Mercer, C.W.** (1989) ARTS: a distributed real-time kernel Operating Systems Review, Oper. Syst. Rev. (USA), vol. 23: no. 3, July 1989, pp. 29-53.

4 **Jurgen, Ronald K.** (1992) Digital Video., IEEE Spectrum, March 1992, Vol. 29: No. 3, pp. 24-26.

5 **Johnson, Ross E. and Goodman, James R.** (1991) Interconnect Topologies With Point-to-point Rings, Computer Sciences Department University of Wisconsin - Madison, Computer Sciences Technical Report #1058, December 1991.

6 **Scott, Steven L., Goodman, James R. and Vernon,Mary K.** (1991) Analysis For The SCI Ring, Computer Sciences Department University of Wisconsin - Madison, Computer Sciences Technical Report #1055, November 1991

7 **Gustavson, David B.** (1992) The Scalable Coherent Interface and Related Standards Projects, IEEE Micro, February 1992, pp. 10-22.

8 **Turner, Jonathan S.** (1992) Managing Bandwidth in ATM Networks With Bursty Traffic, IEEE Network, Vol. 6: No. 5, September 1992, pp. 50-58.

9 **Dannenberg, Roger B., Neuendorffer, Tom, Newcomer, Joseph M., Rubine, Dean** (1991) TACTUS: Toolkit Level Support For Synchronized Interactive Multimedia, ITC Internal Report Carnegie Mellon University.

10 **Rashid, R.F.; Tokuda, H.** (1990) Mach: A System Software Kernel Computational Technology for Flight Vehicles Symposium, Washington, DC, USA, 5-7 Nov. 1990, Computing Systems in Engineering; vol.1: no.2-4, 1990, pp. 163-9.

11 **The OSF/1 operating system EurOpen.** (1991) UNIX Distributed Open Systems in Perspective. Proceedings of the Spring 1991 EurOpen Conference, Tromso, Norway, 20-24 May 1991, Buntingford, UK, EurOpen; viii+331, 1991, pp. 33-41

12 **Mantei, M. M., Baecker, R. M., Sellen, A. J., Buxton, W. A. S., Milligan, T., and Wellman, B.** (1991) Experiences in the use of a media space. In S. P. Robertson, G. M. Olson, and J. S. Olson (Eds.), Proceedings of CHI '91, pp. 203-208. New York: Association for Computing Machinery.

13 **Tani, M., Yamaashi, K., Tanikoshi, K., Futakawa, M., and Tanifuji, S.** (1992) Object-oriented video: interaction with real-world objects through live video. In P. Bauersfeld, J. Bennett, and G. Lynch (Eds.), Proceedings of CHI '92, pp. 593-598, and 711-712 (color plates). New York: Association for Computing Machinery.