AN IMAGE ACCESS PROTOCOL : DESIGN,IMPLEMENTATION AND SERVICES

by

KONG TAT CHEONG

A Thesis

Submitted In Partial Fulfilment Of

The Requirements For The Degree Of Master Of Science

Division Of Information Engineering

The Chinese University of Hong Kong

MAY 1992

UL

360286

thesis
TA
1632
K66

ABSTRACT

This thesis presents the design of an Image Access Protocol (IAP), and the implementation of an Image Access System (IAS) using this protocol. The Image Access Protocol is designed for remote image retrieval and processing applications. It comprises the session and presentation layers of the OSI 7-layer model specified by the International Standards Organization. Using a client-server model, IAP defines how a client computer access image services from an image server across communication networks. Image servers are usually image database managers with image processing capabilities while client computers are common display stations or desk-top computers. As IAP is flexibly designed for systems with different image formats and processing powers, a wide variety of image applications can be built upon it.

Based on the Image Access Protocol, an Image Access System is implemented on DEC workstations with ULTRIX platform which are interconnected by an Ethernet LAN. In this system, an image server workstation is built to provide a wide range of image services to other client workstations. These services include medical image database access, image scaling or coding, and color image (color-mapped and 24 bit RGB format) retrieval services. Although no special image processing hardware is equipped, the image system has shown satisfactory performance for accessing

images with medium size (512x512) and resolutions (256 colors). However, for complex image processing tasks with high resolution images, dedicated image processing boards should be employed in the image server to obtain a better result.

ACKNOWLEDGEMENT

I would like to take this chance to show my earnest thanks to my project supervisor, Dr. P C Wong, for his continuous support throughout the project and the valuable suggestions given to me. I am also grateful to Mr. M S Yeung and Mr. Wallace Wong for their advice and recommendations, especially for the software developments in this project.

CONTENTS

CHAPTER 1. INTRODUCTION

Nowadays, Local Area Networks (LAN) have been widely implemented in different organizations. These networks typically have throughputs range from several Mbps to tens of Mbps and they can carry millions of data bytes within seconds. Moreover, with the advance of technologies, the throughputs of these networks are still growing. Hence, we may expect future LANs to carry more traffics and support more services with satisfactory performance.

As network capacities increase, networks are being directed to carry more bandwidth-hunger services such as images and motion video. These image communication services will play an important role in network communications in future. Some designs and implementations of image services in LANs can be found in recent literatures [1] [2].

Based on the above observations, this thesis presents the development of an Image Access System on a typical LAN environment so that existing LAN bandwidth and resources can be effectively utilized to provide flexible image services. Development of the system begins with the design of an Image Access Protocol (IAP), which governs the communications between an image server and its requesting clients. This protocol occupies the session and the presentation layers of the ISO's OSI 7-layer model. Hence it can be easily implemented in present as well as future LANs by interfacing with the LANs' transport layers.

Implementation of the Image Access System is made on DEC workstations (ULTRIX platform) which are interconnected via an Ethernet LAN. The Image Access Protocol in the system uses the TCP/IP transport layer for reliable message communications [3]. An image server has been built on one of the workstations. It manages an image database and accepts image service requests from the client workstations. The services provided by the image server include image retrieval, pre-processing and coding. Although this Image Access System is only a prototype, it can support practical applications such as image library and remote medical image diagnosis with satisfactory performance.

The organization of this thesis is as follows. A review of some recent literatures and developments in image communications are presented in Chapter 2. This review gives the reasons why the design of the Image Access Protocol is required.

In Chapter 3, the design principles, operation mechanisms and the packet formats of the Image Access Protocol are described. A complete example of a typical IAP communication session is also given in this chapter.

In Chapter 4, the implementation of the Image Access System is explained. This includes detail descriptions for the client and server program structures, the software modules interfaces, as well as the system's operations and applications.

Several schemes for providing enhanced services in the Image Access System are discussed in Chapter 5. These include progressive transmission, call management, priority control, and concurrent control.

Lastly, in Chapter 6, the conclusion of this thesis is presented.

CHAPTER 2. RECENT RESEARCH REVIEW

In the past few years, several research centers and universities have been working towards the design and implementation of image transfer protocols and video communication systems [6]-[11]. Most of these developments are concerned with packet videos and fast packet switching networks which require specialized hardware and protocol hierarchies. For example, Shimizu, Mera and Tani developed a prototype image communication system in a 400 Mb/s multiaccess loop LAN with a 100 Mb/s user interface. This system employs a dedicated channel for image transmission and is capable of transferring several frames of high resolution image in one second [2].

Although these research systems have demonstrated good operational performances and high image qualities, they are essentially incompatible with present systems and thus cannot be implemented in existing Local Area Networks such as Ethernets and Token Rings. Hence extensive uses and applications of these sophisticated designs are very unlikely until more powerful communication networks (such as ATM networks) are widely implemented.

Some recent research works in the area of image communications have concentrated in specific image applications. For instances, Kositpaiboon and his colleagues have designed a packetized image transfer system over LANs for radiographic and medical images [10]. Chamzas and Duttweiler have proposed a facsimile encoding algorithm for bilevel images transfer in packet switched networks [11]. These works have taken into account the statistical behaviors and usage patterns of the concerned images and thus can achieve very efficient operations. However, the designs of these systems are usually based on specific image formats and focused on particular image applications. Therefore, extending these image communication systems to support other image applications may require substantial modifications.

There have been some image communication systems developed from off-the-shelf hardware [17] [18]. One example is a Medical Communication System [17] built by Karmouch et al using SYTEK 6000 LAN with CSMA/CD medium access control (MAC) protocols. The server in the system is a SUN-3/160 workstation while the client terminals are Compaq 386/20 microcomputers.

A test implementation of this system is at the Ottawa Civic Hospital for distribution of X-ray films to radiologists and physicians. As common EGA displays are used for the client terminals, the resolution of analog X-rays are reduced to 1000 by 1024, 8 bit gray scale, before image transmission and display. This may have caused some contrast problems on viewing the images as indicated by some radiologists participated in the trial operation of the system. The author suggested that enhancement algorithms can be used to overcome this problem.

Based on the above studies, we find that different designs of image communication systems have pros-and-cons on various aspects such as system complexity, operation efficiency, flexibility and ease of implementation. As very limited time is available for this project, we chose to design an image communication system which is compatible with existing LAN architectures so that its implementation would not be too complicated. The advantages of such a system are simple, flexible and easily expandable.

In next chapter, we first presents a protocol - the

Image Access Protocol (IAP), specifically designed for building the image communication system. This protocol can be implemented in existing Local Area Networks without the need of additional hardware. As it is mainly a session layer protocol (with a few presentation layer functions), it can interface directly with the transport layers of communication networks. This relieves the burdens of low-level protocol design and network software modification. For instance, the Image Access Protocol is implemented in a Unix network by accessing the Berkeley Socket interfaces of the Transport Control Protocol [4]. Details of this implementation is given in Chapter 4.

Furthermore, the Image Access Protocol is designed to be flexible and enhancable so that it is not limited to specific image applications. Thus it can be used as the framework for developing different image communication services.

CHAPTER 3. IMAGE ACCESS PROTOCOL

Nowadays, image communications in local area networks are usually performed by file transfer applications such as File Transfer Protocol (FTP) and Trivial File Transfer Protocol (TFTP). Although these protocols ensure reliable transmission of image pixels, additional image services are not supported. For example, a user cannot use FTP to request an image server to perform image processing operations, such as compressing an image before transmitting that image.

Hence, it is required to design an Image Access Protocol which defines a communication standard for client computers to access image services provided by image servers. With this protocol, image applications can be developed for multi-vendor computers across communication networks.

The Image Access Protocol is built directly above the transport layer of the International Standards Organization's OSI 7-layer model. Hence it can be readily incorporated in any communication networks by accessing the respective networks' transport layer interfaces. By means of the session layer of the IAP, a client computer can establish an end-to-end communication session with an image server. Then images can be retrieved from the server for processing and display at the client computers.

Besides the session layer communications, IAP also includes a presentation layer. This layer defines what pre-processing and coding operations are requested by the client computer before retrieval of the image. Typical image processing operations supported by image servers are image compression, encryption, and transformation. [12]

The Image Access Protocol does not involve the application layer specifications. For instances, neither the image database structure at the image server nor the image display mechanism at the client computers are included as part of IAP. Thus IAP is not bound to specific applications and can be used as a backbone for developing wide varieties of image services under different hardware and software platforms.

Three steps have been used to design the Image Access Protocol :

1) Formulation of the design principles,

2) Specification of protocol operation mechanism, and

3) Design of the detail IAP packet formats. The IAP packets are the message packets which actually flow between a client computer and an image server.

These design steps are thoroughly explained in the following sections.

3.1 Design Principles

Based on the characteristics of image applications, four design principles for the Image Access Protocol are identified in the followings.

1. Connection Oriented

When a user wants to access an image server, it is likely that more than one image request will be issued. Therefore, in order to reduce the average amount of control overheads in each image request, it would be more efficient to use a connection oriented protocol for communications between a client workstation and an image server. With such method, the control data for a session, such as password or server address, needs to be specified only once during session setup.

In the Image Access Protocol, this connection oriented approach is adopted and a communication session is set up between an image server and a client computer before any image service requests. The session setup includes connection request, password or client identity check, and server confirmation. IAP is also responsible for the session clearance, which involves release of system resources and mutual clearance confirmation.

2. Reliable

As the Image Access Protocol is a session layer protocol, it is assumed to receive reliable transport services from the lower layer. Thus there is no need for IAP to duplicate transport layer operations such as transmission error checks, packet flow controls and sequence managements.

However, the IAP should ensure reliable communications between the client and server application processes. This requires end-to-end control for the process operations. For example, the image server should send image information such as the image size and format prior to actual transmission of the pixels, so that image clipping at the client computer due to memory or buffer overflows can be avoided during access of large images.

Well-planned procedures for system or process error recoveries are also necessary. This can be achieved by interchange of error control packets between the client and server computers. Moreover, if non-recoverable errors (such as image server disk failure) are encountered, the user at the client computer should be informed and the communication session should be automatically cleared when all engaged system resources have been released.

3. Flexible

The Image Access Protocol specifies a communication method by which a client computer accesses the image services provided by an image server across a communication network. As we hope that different image applications can be developed basing on the protocol, its design should be as flexible as possible for supporting access to a wide range of image services. Thus in the IAP we categorize image processing operations in a broad sense as followings, so that most image operations provided by image servers can be specified.

(a) Image Pre-processing

Image Pre-processing are defined as any image processing operations which modify the contents of an image before retrieval by the client computer. Examples are scaling, dithering, edge detection etc. One practical use of this operation would be to scale down a high-resolution

2048x2048 image to a 512x512 image at the server before transmission to the client workstation so as to save network bandwidth.

(b) Image Coding

Image Coding are defined as any image processing operations at the image server which have a counter operation at the client computer. For example, an image is encrypted for security reasons at the image server before transmission and then decrypted at the client computer after retrieval.

It is hoped that the amount of image services that can be provided are limited only by the image server's processing power but not by the design of IAP. Furthermore, in order to meet the requirements of different image applications, IAP should also support multiple image formats.

4. <u>Integrated with data services</u>

Image servers usually organize its images as a database, and it is possible that there would be data or information associated with the images. Accordingly, the

Image Access Protocol should incorporate remote database information retrieval services such as browsing, sorting and searching. As it will be too complicated to incorporate a complete distributed database access protocol in the IAP, only simple database operation services will be included. Advanced database services may be added in later versions of the protocol.

3.2 Protocol Mechanism

Based on the above design principles, the operation mechanism of the Image Access Protocol can be formulated. A simplified graphical representation of the protocol mechanism is shown in Figure 3.1.

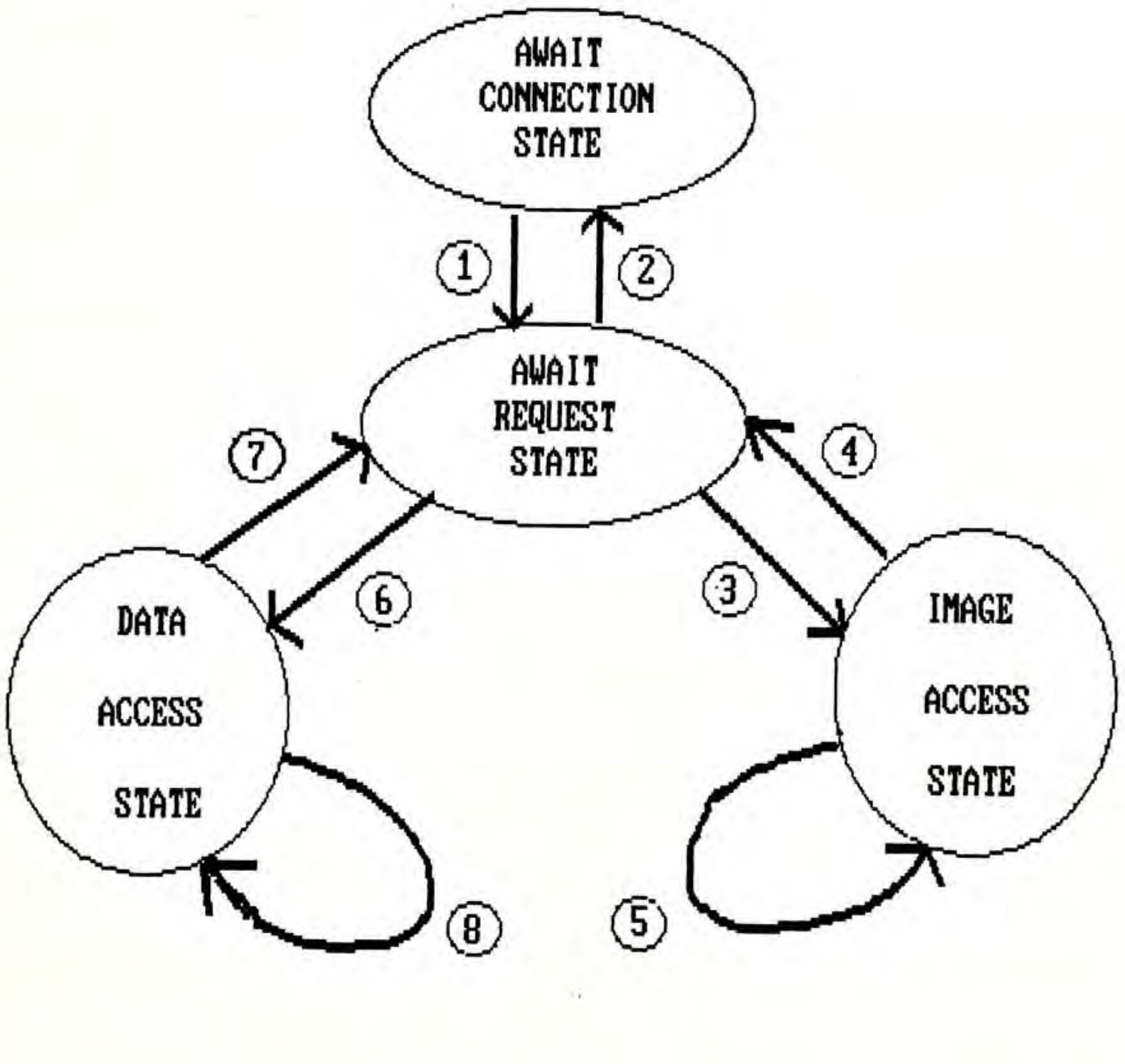


Figure 3.1 SIMPLIFIED PROTOCOL MECHANISM

There are four operation states in the protocol. Transitions of the states are accompanied by interchanges of IAP packets between the client and the image server. These state transitions have been labelled in Figure 3.1.

(i) Await Connection State

In this state, the image server is awaiting session connection request from client computers. When the server receives a CALL_SETUP packet with valid password, transition 1 occurs and the "Await Request State" is entered.

(ii) Await Request State

The image server is ready to accept image or database information requests in this state. When a client computer issues an IMAGE_REQUEST packet to the image server, transition 3 takes place and the protocol enters the Image Access State. On the other hand, the client may issue a DATA_REQUEST packet which causes transition 6 to take place and terminates at the Data Access State.

When the client computer wants to terminate the session, a CALL_CLEAR packet may be issued and the Await Connection State will be entered via transition 2. The transition 2 will also be initiated when an ERROR_CONTROL packet is issued by the client computer signifying that it encounters fatal errors.

(iii) Image Access State

In this state, the image server acts differently according to the previously received image requests. Two cases are possible:

a) If the requested image is non-existence or the requested image processing operation is not supported, an ERROR_CONTROL packet will be sent back to inform the client computer. This accompanies with transition 4 which terminates at the Await Request State, where the client may re-issue a correct image request.

b) If the client's image request is successful, the image server sends the requested image to the client using IMAGE_BLOCK packets. Since image sizes are very large, transfer of many IMAGE_BLOCK packets are usually required. This transmission process is signified by transition 5, which starts and terminates at the Image Access State. When the last IMAGE_BLOCK packet is sent, transition 4 occurs and the Await Request State is entered. The client may then issue a new request at this state.

(iv) Data Access State

In this state, the image database information at the image server is transferred to client computers, via DATA_BLOCK packets in transition 8, where terminates at the same state. When all requested data has been sent, transition 7 takes place and the protocol returns back to the Await Request State. If the client computer wants to abort the data retrieval operation prematurely, it may issue the DATA_ACKNOWLEDGE packet to the image server and transition 7 is also resulted.

When a fatal error occurs at the image server (such as disk access failure), the server will send ERROR_CONTROL packets to all connected client computers and the sessions will be cleared regardless of the present state. If a fatal error occurs in a client computer, only the session associated with that client is cleared by transferring an ERROR_CONTROL packet to the image server. These operations are of highest priority and will override the state transitions.

3.3 IAP Packet Formats

There are eleven types of packets defined in the Image Access Protocol. Every packet begins with a 2-byte length field, then a 2-byte opcode field, and followed by some optional variable-length data fields.

The length field stores the amount of the trailing data in a packet for facilitating receiver end buffer control. The opcode field specifies the type and usage of a packet. The optional data fields contain control information or image pixel data to be interchanged between image server and clients. In future, additional IAP packet types may be incorporated in the protocol to enhance its functions and operations.

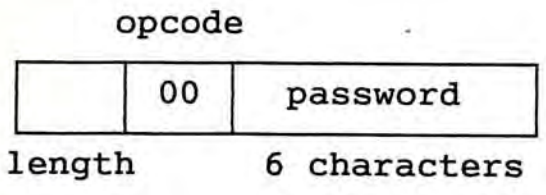
In the IMAGE_REQUEST packet, two bytes are used in the image format and processing control fields. Thus the total number of image formats, as well as that of image processing and coding operations which can be supported by the Image Access Protocol are all 65536 (2 to the power of 16). This should be more than enough for most image applications. [12]

Image compressing and processing standards are being developed by the Joint Photographic Experts Group (JPEG), as organized by ISO in close collaboration with CCITT [13] [14]. As these standards become finalized, they can be incorporated into the Image Access Protocol to achieve an image communication standard.

The eleven IAP packets are depicted below :

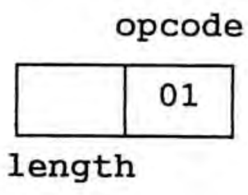
1.CALL_SETUP

(sent by client)



2.CALL_CONFIRM

(sent by server)

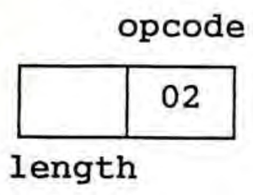


Function: Set up a session between the image server and a requesting client computer.

Note: All alpha-numeric characters can be used in the password field, thus six characters is long enough to provide sufficient security protections.

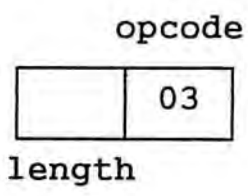
3.CALL_CLEAR

(sent by client)



4.CLEAR_CONFIRM

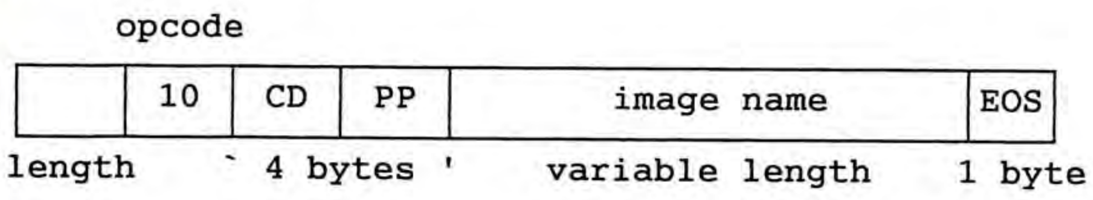
(sent by server)



Function: Clear the session between the image server and the associated client computer

5.IMAGE_REQUEST

(sent by client)



CD = image coding option
PP = image pre-processing option

Function: Request image processing services and an image with the specified name from the image server

Note: For flexibility, the length of an image name is not limited and its end is marked by an EOS byte. Two bytes are used in both CD and PP option fields so that 65536 image operations can be specified for either of them.

6.IMAGE_CONFIRM

(sent by server)

opcode

	11	HT	WT	FM	CR	image name	EOS

length ` 2 bytes each '` variable length'

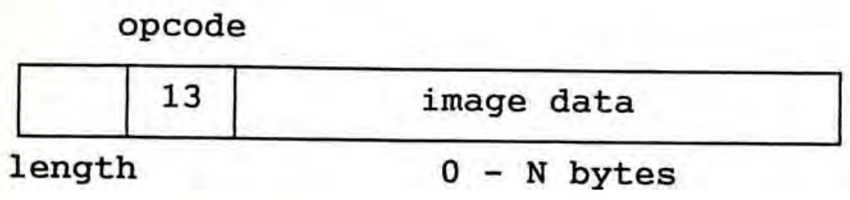
HT = height of image
WT = width of image
FM = image format
CR = number of colors/gray levels

Function: Confirm for image request and transmit image information to the requesting client.

Note: Maximum size of image supported is 65536x65536. Also 65536 image formats can be specified. The exact meaning of the contents of the CR field is related to the image format so that maximum flexibility is achieved.

7.IMAGE_BLOCK

(sent by server)



Function: Transmit image pixel data from server to client.

Note: Maximum image block size (N) can be adjusted to achieve maximum transmission efficiency. For example, N is set to not larger than 1456 in Ethernet so that packet fragmentation is avoided. Please refer to Section 4.3 for the analysis.

The requested images is segmented into blocks of N pixels in rastering sequence and each pixel block is transmitted by an IMAGE_BLOCK packet. The end of image transmission is signified by an IMAGE_BLOCK packet with image data size less than N. If the image size is exactly divisible by N, an extra IMAGE_BLOCK packet with null image data will be sent for signalling the termination of transmission.

8.DATA_REQUEST

(sent by client)

opcode

	20	OP	image name	EOS

length ` variable '
length

OP = request option

Function: Request database information from server

Note: The OP field specifies the requested database operation or information. The image file name is optional and may not be required for some OP options.

9.DATA_ACKNOWLEDGE

(sent by client)

opcode

	22	CT

length

CT = data flow control

Function: Acknowledge and control the flow of database information

Note: The CT option is for data flow control purposes such as termination of a database request or temporary pause of the data transfer.

10.DATA_BLOCK

(sent by server)

	opcode	
	21	image database information
length		variable length

Function: Transmit database information from the image to the client computer

Note : Image database information field is a variable length message field which usually contains the names, heights, widths and formats of requested images. Exact contents is not specified in the protocol so that different image database structures can be supported.

11.ERROR_CONTROL

(sent by both server and client)

	opcode	
	30	EC
length	2 bytes	

Note: EC = Error codes for Image Access Protocol

EC	Descriptions
0	Undefined error
1	Invalid password
2	Image non-existent
3	Coding option not supported by server
4	Pre-processing option not supported
10	Fatal error (cause immediate disconnection)

Function: Report errors and control information for the client and server processes.

Note: Two kinds of errors (EC=0 and EC=10) are currently defined which cause immediate termination of the communication session. The other four control information (EC=1 to EC=4) causes termination of the associated connection or image request. Please refer to Section 3.2 for details. Additional error code definitions may be incorporated for future image applications.

3.4 Protocol Operation Example

The normal operations of IAP are best shown by a packet interchange diagram (Figure 3.2).

```
 CLIENT                              SERVER

 CALL_SETUP          ----->
                       <-----        CALL_CONFIRM
 DATA_REQUEST        ----->

                                (search image database)

(display data)         <-----        DATA_BLK (blk 1)
 DATA_ACK (blk 1)----->
(display data)         <-----        DATA_BLK (blk 2)
 DATA_ACK (blk 2)----->
                          .
                          .
                          .
(display data)         <-----        DATA_BLK (blk n)
 DATA_ACK (blk n)----->
 IMAGE_REQUEST       ----->

                             (search for image, not found)

                       <-----        ERROR (error code = 2)
 IMAGE_REQUEST       ----->
                       <-----        IMAGE_CONFIRM

                            (image pre-processing\coding)

                       <-----        IMAGE_BLOCK
                          .
                          .
                          .
(display image)        <-----        IMAGE_BLOCK
                                     (last block is not full)
 CALL_CLEAR          ----->
                       <-----        CLEAR_CONFIRM
```

Figure 3.2 IAP Packet Interchange

In this operation example, the client initiates a connection request to the image server by issuing the CALL_SETUP packet. After checking the validity of the password carried by the packet, the image server responds with a CALL_CONFIRM packet, signifying that a session has been successfully set up. Then the client retrieves image database information from the server by issuing a DATA_REQUEST packet. The image server complete the request by sending the requested data to the client via DATA_BLOCK packets. In the meantime, the client sends back the DATA_ACKNOWLEDGE packets to the server to control the data flow so that the received data would not scroll off the client's screen too fast.

When the database request operation is finished, the client then issue an IMAGE_REQUEST packet to the image server. However, this is unsuccessful because the server finds that the requested image is non-existent and inform the client with an ERROR_CONTROL packet. Then the client retries another image request and this request is successful. The image server then performs the requested image pre-processing and coding operations as specified in the IMAGE_REQUEST packet. After that the processed image is transferred to the client by means of IMAGE_BLOCK packets.

During transfer of the image pixels, there is no individual acknowledgement for each IMAGE_BLOCK packet. This is based on the assumptions that a reliable stream channel has been provided by the transport layer protocol. End of the image transmission is signified by the last IMAGE_BLOCK packet which is a 'non-full' packet so that the client knows when the transmission ends and displays the received image in its screen. The size of a 'full' packet is set for achieving maximum efficiency in the network where the Image Access Protocol is implemented. For example, Ethernet allows a maximum packet size of 1526 bytes without fragmentation and transfer of data using this packet size results in the least proportion of overheads.

After the client has received the whole image, it terminates the session by issuing a CLEAR_REQUEST packet to the server which responds with a CLEAR_CONFIRM packet, after releasing its engaged system resources.

CHAPTER 4. SYSTEM IMPLEMENTATION

Based on the Image Access Protocol, a prototype Image Access System (IAS) is built. The hardware configuration of the system is shown in Figure 4.1.

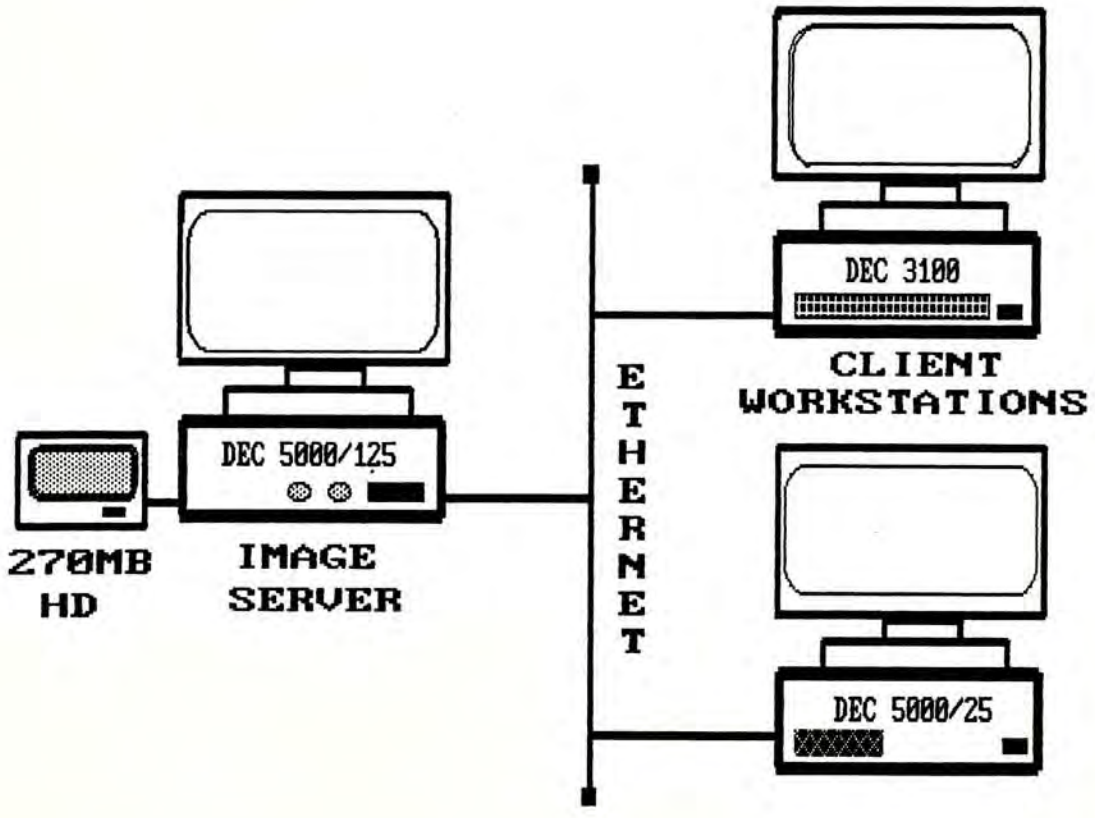


Figure 4.1 IMAGE ACCESS SYSTEM HARDWARE CONFIGURATION

The system comprises an IAP Image Server and two IAP Client workstations, all of which are interconnected via an 10Base2 Ethernet LAN. The operating system running in these workstations is ULTRIX, which is compatible with Berkeley 4.3BSD version of UNIX [4]. Since ULTRIX is a multi-tasking operating system, we developed the IAP image server as a concurrent server. Hence it can serve more than one client workstation at the same time.

The IAP Image Server is a DEC 5000/125 workstation with a 270 Megabytes external hard-disk. A small image database is already stored in this hard-disk. Inside the image server, a server program - 'IAP SERVER' is running which waits for session setup requests from the client workstations.

One of the client workstations is an DEC 3100 workstation and the other one is an DEC 5000/25 personal workstation. Both of them are equipped with high resolution display units (1024x1024). The 'IAP CLIENT' program is run in these client workstations for requesting image services from the IAP server.

4.1 Software Architecture and Interfaces

The IAP SERVER and IAP CLIENT programs are the System Software of the Image Access System. They are developed in standard 'C' Language and run under the ULTRIX operating system of the workstations. The software modules in these programs are shown in Figure 4.2.

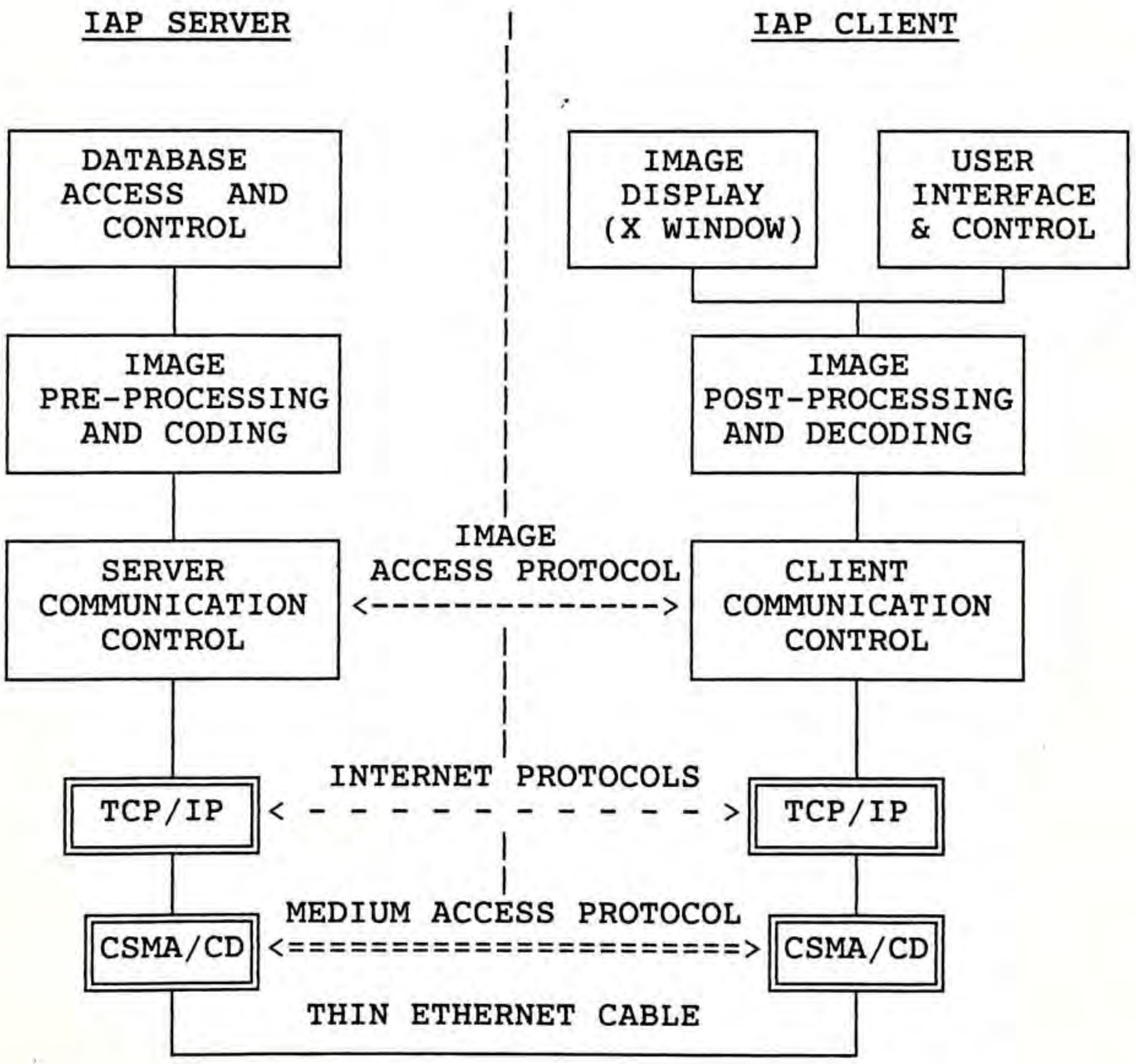


Figure 4.2 SOFTWARE MODULES FOR IMAGE ACCESS SYSTEM

Refer to Figure 4.2, the Server and the Client Communication Control Modules are responsible for implementation of the Image Access Protocol, by interface with the BSD Sockets [4]. They contain a number of functions which can be used for developing new IAP image systems in communication networks supporting TCP/IP protocols. These functions form the Application Programming Interface (API) for the Image Access Protocol.

The "Pre-processing And Coding Module" and the "Post-processing And Decoding Module" contain image processing sub-programs which are dedicated to specific image manipulations. New image processing services can be added at the image server easily by incorporating relevant sub-programs in these modules.

The top-level modules in the System Software are application specific modules. They control the logic of the programs, implement the user interfaces, and operate the image database and display images via X-window library calls. [5] With these modules, the Image Access System is virtually a complete system which provides hands-on image applications.

Interfaces between the program modules are via function calls. This maintains highest flexibility because new functions can be added to these modules according to the specific image applications required.

4.1.1 'IAP SERVER' Program

The IAP SERVER program has three program modules.

1. Server Communication Control Module

The Server Communication Control Module implements the server operations of the Image Access Protocol. It is responsible for connection setup, session clearance, and transmission and reception of IAP packets. The functions in this module are included in a header file (IAP_SERVER.h) as the application programming interface (API) for the Image Access Protocol. Please refer to the Appendix 1 for the descriptions of this API.

2. Image Pre-processing and Coding Module

The Image Pre-processing and Coding Module contains sub-programs which actually perform image pre-processing and coding operations for various image formats. This module can be viewed as a library of routines which are executed according to the pre-processing and coding options specified in a received IMAGE_REQUEST packet. Thus the services of this Image Access System can be easily extended by incorporating additional processing routines in this Module.

3. Database Access and Control Module

The image application services of the image server are implemented in the Database Access and Control Module. It is responsible for retrieving images from the image server hard-disk and accessing image information in the image database. Furthermore, it controls the logic flow of the program according to the image services being requested. When a new connection request reaches the server, this module forks a child process for handling the requests come from this new client. Thus the image server is a concurrent server because simultaneous connection sessions can be set up and multiple client workstations are served at the same time.

The pseudo-codes for the IAP SERVER are shown as follows. The functions begin with an asterisk (*) belong to the Image Pre-processing and Coding Module. Those with a double asterisk (**) are in the Server Communication Control Module. The Database Access and Control Module contains all other codes and forms the main structure of the program.

```
/* IAP SERVER Program */

#include "IAP_SERVER.h" /* header file for IAP operations */

request_handler()
{
  for (;;)
  {
    **receive_packet();
    switch (IAP packet type)
    {
      case IMAGE_REQUEST:
        image_search();
        if (image exists)
        {
          image_retrieval();

          switch (pre-processing option)
            case option 1:
              *image_pre-processing_1();
                  .
                  .
            case option N:
              *image_pre-processing_N();

          switch (coding option)
            case option N:
              *image_coding_N();
                  .
                  .
            case option N:
              *image_coding_N();

          if (both image operations are successful)
            **send_image();
          else
            **report_error(); (EC = 3 or 4)
        }
        else
          **report_error(); (EC = 2)
        break;

      case DATABASE_REQUEST:
        image_database_search();
        **send_data();
        break;

      case CLEAR_REQUEST:
        **confirm_clear();
        release system resources;
        exit();

      default:
        **report_error();
        exit() on error;
    }
  }
}


main()
{
  for (;;)
  {
    socket initialization;
    **receive_packet();
    if (CONNECT_REQUEST packet received)
      {
        check password;
        if (password correct)
          {
            **confirm_connection();
            fork a child process;
            if (in child process)
              request_handler();
          }
      }
}
```

4.1.2 'IAP CLIENT' Program

The IAP CLIENT program has four program modules and also a modular architecture.

1. The Client Communication Control

The Client Communication Control Module is responsible for IAP operations such as session setup and clearance, IAP packets generation, and communication with the image server. It contains a library of functions organized in a header file (IAP_CLIENT.h), which can be accessed as the interface for implementation of new IAP client applications. Details of these functions as an application programming interface for the Image Access Protocol is given in Appendix 1.

2. The Image Post-processing and Decoding Module

The Image Post-processing and Decoding Module has two duties. Firstly, it decodes a received image which is previously encoded in the image server before transmission. These coding and decoding are counter operations. Secondly, it performs local image processing (post-processing) for the received image, such as image magnification. These post-processing operations are independent from the pre-processing operations in the image server.

3. The Image Display Module

The Image Display Module is an application layer program module. All received images, after decoding and post-processing, are passed to this module for display. Displaying images in the client workstations are by X-window libraries calls [5]. The design of this Image Display Module is dependent on the display hardware of the client computer and thus it is not specified as part of the Image Access Protocol.

4. The User Interface and Control Module

The User Interface and Control Module is responsible for showing message menus, processing user inputs and displaying database information. According to the user choices, it controls the program logic flow and performs the requested operations by calling functions in the other three program modules. Friendly user interfaces and menu driven operations has been incorporated in this module with trapping of user input errors.

The pseudo-codes of the IAP CLIENT program are shown in the following. The functions begin with a single asterisk (*) are found in the Image Post-processing and Decoding Module. Those with a double asterisk (**) are belonged to the Client Communication Control Module. The Image Display Module begins with a triple asterisk (***) and all the rest codes can be found in the User Interface and Control Module.

```
/* IAP CLIENT PROGRAM */

#include "IAP_CLIENT.h"    /* header file for IAP functions */

image_request()
{
  for (;;)
    {
      display image request menu;
      switch (user's choice)
        {
          case choose options:
            get image name, pre-processing and coding options;
            break;

          case send request:
            **request_image();
            if (request successful)
            {
              *image_decoding();
              ***image_display_module();
            }
            break;

          case local processing:
            if (an image is already retrieved)
              *image_postprocessing();
              ***image_display_module();
              break;

          case quit:
            return to main menu;
        }
    }
}

data_request()
{
  display data_request menu;
  get user's searching option;
  **request_data();
  if (request successful)
    display image database information;
}

main()
{
  socket initializations;
  get user password;
  **request_connection();
  while (session setup is successful)
    {
      display main menu;
      switch (user's choice)
        {
          case request images:
            image_request();
            break;

          case request database information:
            data_request();
            break;

          case quit:
            **send_clear();
            exit();
        }
    }
}
```

4.2 System Operations and Applications

Images of four different formats (Table 4.1) are stored inside the image server for testing the operations of the Image Access System. With these four image formats, the system is able to support many image applications.

FORMAT	DESCRIPTIONS	IMAGES IN DATABASE
GRAY256	Raw pixel maps with 256 gray levels	Common images used in image processing
HIPS	Variable resolution pixel maps	Medical images : Magnetic Resonance Images (MRI) and Computer Tomography (CT)
Img	Color-mapped raster files in Img Software Set	General color and gray level images
RGB	High resolution 24 bit RGB images	Fine portrait and photo quality images

Table 4.1 IMAGE FORMATS IN IMAGE ACCESS SYSTEM

Operational tests of the Image Access System were carried out to investigate the image access performance of the system. A wide range of image sizes (128x128 to 1024x1024) have been used in the tests. These tests verify that many practical image applications can be supported by the Image Access System.

1. Images Retrieval Test

Retrieval of images in the above four formats from the IAP server has been practised when the Ethernet is moderately loaded. The average access time for an 512x512 image for either one of the first three image formats (GRAY256, Img, HIPS) is around 12 seconds, of which about two-third is used in image display. For smaller 256x256 medical HIPS images, the average access time even reduces to about 3 seconds. With dedicated image display terminals, this access time can be further reduced by decreasing the display time. Hence, the performance of the Image Access System is satisfactory for accessing medium-sized images of general formats. Furthermore, general image applications such as image library and relational image database can be easily developed based on it.

Retrieval of RGB images is also verified. Since each RGB image is composed of three image components (RED, GREEN, BLUE), the access time of an 512x512 RGB image requires approximately 130 seconds which is quite unsatisfactory. About 85% of this time is used for image displaying at the client workstation while the rest is used in image transmission. Hence the bottleneck is also in the image display module, which can run much faster with specialised display processors. Nevertheless, this prototype Image Access System supports access for images of photo-quality as required in journalistic or advertising applications.

2. Image Pre-processing and Coding Test

The image server supports simple image pre-processing operations such as image scaling and segmentation for GRAY256 and RGB images. Since these processings are software controlled, they normally take rather long time for operating on RGB images. For instance, shrinking an 512x512 RGB image to a quarter of its size would require as long as 1.5 minutes. Hence it is advised to install image processing boards in the server if many image processing requests from client workstations are expected which justifies the cost of the additional hardware. Otherwise, the performance of the whole Image Access System will be suffered.

Image coding and decoding are counter operations in the image server and the client workstation respectively. In the Image Access System, the Unix File Compression and Decompression Utilities are employed. These operations reduce about 20% of image sizes and save the same proportion of LAN bandwidth. However, by employing powerful image compression hardware such as Transform Coding Processors, up to 90% of image sizes can be reduced. With sophisticated coding schemes, much network capacity can be saved and security in the image transmission is improved. Hence this IAP coding option could be very useful for implementation of Image Access Systems in slower and public networks such as a Packet Switching Network or an ISDN.

3. Image Post-Processing Test

An image post-processing operation for HIPs images is implemented in the IAP CLIENT program. It is an image magnification operation applied specifically on the HIPS medical images (MRI images and Computer Tomographes) whose sizes are normally 256x256. The magnification scale can be adjusted according to the image contents. This magnification operation is based on linear interpolation from adjacent image pixels [12], which maintains very good qualities for images without sharp edges. Local magnification at client workstation is preferred to processing at the image server because it is more flexible and occupies less network bandwidth by transmitting smaller images.

Typical 256x256 medical images are too small for the fine details to be exposed and identified. Thus they are usually magnified at the client workstations for at least four times as large to become 512x512 images. This provides doctors and medical experts with fairly clear medical images so as to facilitate their diagnosis. Therefore, the Image Access System can be readily used as a remote medical image diagnosis system for sharing of a centralized medical image database in hospitals.

4.3 Image Transmission Efficiency

The transmission efficiency of the Image Access System can be estimated by calculating the relative overheads in an IMAGE_BLOCK packet used for image transmission. For simplicity, the following assumptions were taken for the computation.

1. Inefficiency due to transmission errors, packet collisions and fixed minimum packet spacings in Ethernet are neglected. Analysis for these factors can be found in Reference [16].

2. The Transport Control Protocol (TCP) would send one acknowledgement packet for each TCP data packet [3]. As the IAP packets are transported by TCP in the Image Access System, every IMAGE_BLOCK packet is associated with an TCP acknowledgement packet. This acknowledgement packet is a transport layer function implemented by TCP. Hence it is completely transparent to the upper layer protocols such as the IAP.

3. The maximum allowable packet size (1526 bytes) of an Ethernet packet is reached for building the IMAGE_BLOCK packet of the Image Access Protocol. This assumption is

usually justified for image transmissions because a large amount of data is involved which makes most packets to be transmitted with the maximum allowable size. This results in minimum relative overheads in the transmission.

The encapsulation of an IMAGE_BLOCK packet in the Image Access System is shown in Figure 4.3.

Ethernet Header	IP Header	TCP Header	IAP Header	Image Data	CRC
22 bytes	20 bytes	20 bytes	4 bytes	1456 bytes	4 bytes

Figure 4.3 Encapsulation of IMAGE BLOCK packet

In an IMAGE_BLOCK packet :

Total Overheads = 22 + 20 + 20 + 4 + 4

= 70 bytes

Total Image Data = 1526 - 70

= 1456 bytes

An TCP acknowledgement packet contains all the fields depicted in Figure 4.3 except the IAP Header and Image Data. Hence the data in this packet = 22 + 20 + 20 + 4 = 66 bytes. However, the minimum allowable packet size in Ethernet is 72 bytes. Therefore, the actual size of the TCP acknowledgement packet is 72 bytes.

Summarizing above, total overheads for transmitting one IMAGE_BLOCK packet = 72 + 70

= 142 bytes

Hence, the image transmission efficiency of the Image Access System can be estimated.

$$\text{Transmission Efficiency} = \frac{\text{Image Data}}{\text{Image Data + Total Overheads}}$$

$$= \frac{1456}{1456 + 142}$$

$$= 91\ \%$$

The above computation gives an estimate to the maximum possible transmission efficiency that can be achieved in the Image Access System, under the condition that traffic load in the Ethernet is very low (assumption 1). The image transmission efficiency actually depends on the amount of traffic in the Ethernet. As the traffic load increases, packet collisions become more significant. Then the image transmission efficiency would drop rapidly, as discussed in Reference [16].

CHAPTER 5. ENHANCED SYSTEM SERVICES

The Image Access System can be further improved to achieve better efficiency and enhance functionality. In the followings, several schemes are suggested which can be implemented without massive modifications in the Image Access System.

5.1 Progressive Coding

In slower communication networks, transmission of images usually takes too much time and does not meet the standard required for interactive applications. For example, transfer of an 512x512 8-bit gray scale image via an ISDN B-channel (64kb/s) requires more than 32 seconds. This does not include the image database access time at the image server and the image display time at the client workstation. Thus the total access time is very substantial.

One way to improve this performance is to implement progressive coding and transmission schemes [12]. With these schemes, a rough sketch of an image is transmitted first, and then, the details are added later in successive levels. At the final stage of transmission, the exact replica of the original image is obtained, probably with high overall coding efficiency. These gradual operations are highly desirable in image database visual searches because the user can recognize image content as quickly as possible, then halt transmission of unwanted detail and switch to retry another image.

The Image Access System can also support progressive coding schemes, provided that appropriate image processing modules are added. A trial progressive coding scheme using simple pixel sub-sampling has been implemented in the system which involves three coding stages. In Figure 5.1, a 4x4 pixel block is shown with each pixel's sub-sampling stages labelled.

1	3	2	3
3	2	3	2
2	3	2	3
3	2	3	2

Figure 5.1 Progressive Coding Stages

Stage 1 :

Segment an image into blocks of 4x4 pixels, then sub-sample the top leftmost pixel of each block and transmit. The size of the resultant image is one sixteenth of the original one.

Stage 2 :

In each of the 4x4 image blocks, sub-sample one in every two adjacent pixels except those have been sent in Level 1. These additional pixels are sent to the client workstation which inserts them to the previously received pixels to form an image of half the original image's size.

Stage 3 :

All the pixels labelled '3' in the image block are transmitted. The client workstation then gathers these pixels to form an exact replica of the original image.

This progressive coding scheme can be supported by the Image Access System without any changes in the basic operation mechanism of the Image Access Protocol. Similar to general image requests, a client workstation can initiate progressive transmission for any one of the above stages, by issuing an IMAGE_REQUEST packet with a coding option specifying the corresponding stage. Thus for a whole image to be retrieved, totally three IMAGE_REQUEST packets would be required in the progressive transmission scheme.

Upon receiving an IMAGE_REQUEST packet of this coding scheme, the image server performs image sub-sampling of the specified stage and then transmits the sampled image pixels to the client workstation. When the client received all these image pixels, it assembles them with the pixels received in the previous transmission stages and form the ultimate image for display.

For demonstration of principles only, we have implemented these sub-sampling operations in software. Since image pixel sampling involve manipulations of large amount of data, the total image access time is lengthened substantially. Thus the bottleneck of the progressive transmission scheme is in image sub-sampling at the image server.

This shortcoming can be overcome by sub-sampling all the images and store the sampled pixels as a hiearchical data structure in the server. With this technique, there will be no image pixel sub-sampling operations at the image server during image requests. Thus the Image Access System becomes more efficient and the user at a client workstation can have a fast preview of the requested images. Clearly, this is very valuable in visual searching of images in the server.

In additional to hierarchical data structures, the performance of the Image Access System can be further improved by adopting sophisticated progressive coding schemes [15] implemented in dedicated image processors. Such enhancements are essential for development of image communication systems in slower networks where the performance bottleneck is in image transmission.

5.2 Call Management

As image communications impose very high demands in network bandwidth, excessive image requests would result in reduced throughput of the server and severe network congestions. Hence it is worthwhile to implement a call management scheme at the Image Access System to limit the number of simultaneous connected sessions as well as the amount of image requests.

Such an implementation can be achieved by setting threshold levels on total number of connected sessions and alarm levels on the image request rates. When the preset limit is exceeded, control actions such as pending of image requests and blocking of new session setup can be triggered. In critical situations where severe packet losses occur, clearance of low priority sessions can be exercised.

It is also possible to have excessive image processing and coding requests reaching the image server. In this case, some of the image requests may be blocked so that the image server is not overloaded and satisfactory service quality can be maintained.

5.3 Priority Control

Since the image server in the Image Access System is a concurrent server, there can be many simultaneous sessions connected to it. Hence it is valuable to set up priority controls for various client workstations so that some of them may have a higher access priority for the image server.

As described in Section 4.1.1, the image server uses a separate child process for handling individual connection session. Thus setting priority controls for the sessions requires coordinations of the operation sequence of the child processes. This can be performed by inter-process communication such as Semaphores or Pipes between the parent and the child processes [4]. When an image request of a higher priority session arrives, the parent process in the server can then make the arbitrations and pause the operations of low priority sessions temporarily to release system resources for the prioritized one.

5.4 Concurrent Control

In some applications, it is often required to view and compare several images concurrently. For example, a doctor may want to see a patient's X-ray images from different perspectives simultaneously. This requires a more versatile image request method to be implemented. However, only few modifications in the operation mechanism of the Image Access Protocol is necessary.

Concurrent control for multiple image requests can be incorporated into the Image Access System by modifying the application layer modules of the server and client programs. Using the previous example on X-rays access system, a client program may be designed so that when a doctor selected a patient name, all IMAGE_REQUEST packets relevant to the X-rays of this patient will be issued to the image server. The server then handles these requests one at a time and transmits the requested X-ray images to the client as normal. Finally, when all these requested X-ray images are received, the client workstation may display them side-by-side for the doctor's diagnosis. To the user of the client workstation, this concurrent request of multiple images is completely transparent.

CHAPTER 6. CONCLUSION

The Image Access Protocol is a protocol specifically designed for image access and processing applications in communication networks. It is composed of the session and presentation layers of the OSI 7-layer Model. Using this protocol, client computers can retrieve images and obtain image processing services provided by image servers. Moreover, the protocol is very flexible and different image formats can be supported. Hence most image applications can be developed based on it.

An Image Access System is built on an Ethernet with an image server workstation and two client workstations. The server and the clients communicate via the Image Access Protocol. An image database with images of four different formats is created at the image server. The server also provides some preliminary image processing services to the client workstations such as image scaling and segmentation. The operation test of this system gives satisfactory performance except on high resolution RGB images, which require excessive time to access due to large amounts of data. Moreover, the image transmission efficiency of the system has been found to be rather high (maximum 91 %), as estimated by computation of relative overheads engaged in the transport of IMAGE_BLOCK packets.

Several schemes for provision of enhanced services in the Image Access System are suggested which include progressive transmission, priority control, call management and concurrent control. A trial implementation of a simple progressive transmission scheme has been performed in the system. The result shows that it is a feasible way to overcome the bottleneck in image transmission if dedicated progressive coding processors are implemented.

APPENDIX 1 APPLICATION PROGRAMMING INTERFACE

The Application Programming Interface for the Image Access Protocol are a set of functional interfaces for implementation of IAP operations (based on TCP). Examples for use of these interfaces can be found in the pseudo-codes in Section 4.1. All the functions return '0' for successful operations and '-1' for unsuccessful operations or errors.

A. Interfaces for IAP SERVER

```
1) #include "IAP_SERVER.h"
   Note - a header file storing the functions, pre-defined
          types and variables for the programming interface
          of IAP SERVER operations

2) receive_packet(pkt_field)
   field_struct * pkt_field;

   Note - field_struct is a pre-defined structure:
          struct {char * field1; ... ; char * fieldN;}
        - field1 to fieldN points to the fields of the
          received packet in sequential order
        - these function is called for receiving IAP
          packets

3) report_error(err_code)
   int err_code;

   Note - send ERROR_CONTROL packet with EC = err_code
```

```
4) send_data(data_ptr)
   char * data_ptr;

   Note - send the database information pointed by
          data_ptr
        - receive and process DATA_ACK packets
5) confirm_clear()

   Note - send CLEAR_CONFIRM packet

6) send_image(image_ptr)
   char * image_ptr;

   Note - send the image pointed by image_ptr
```

B. <u>Interfaces for IAP CLIENT</u>

```
1. #include "IAP_CLIENT.h"

   Note - a header file storing the functions, pre-defined
          types and variables for the programming interface
          of IAP CLIENT operations

2. request_image(req_field, img_field)
   req_struct  * req_field;
   img_struct  * img_field;

   Note - req_struct is a pre-defined structure :
          struct {char * cc_opt, char * pp_opt, char *
                  img_name};
          cc_opt = coding option,
          pp_opt = pre-processing option,
          img_name = image name
        - send IMAGE_REQUEST packet with the options
        - img_struct is a pre-defined structure :
```

```
        struct {char * ht, char * wt, char * fm, char *
               cr};
        ht = height, wt = width, fm = format, cr =color
      - receive IMAGE_CONFIRM packet and IMAGE_BLOCK
        packets


3. request_data(data_field)
   data_struct  * data_field;

   Note - data_struct is a pre-defined structure:
          struct {char * req_opt, char * img_name};
          req_opt = request option
          img_name = image name
        - send DATA_REQUEST packet with the options
        - receive DATA_BLOCK packet
        - send DATA_ACK packet

4. request_connection(pw)
   char * pw;

   Note - pw = password
        - send CONNECT_REQUEST packet with the password

5. send_clear()

   Note - send CLEAR_REQUEST packet

6) report_error(err_code)
   int err_code;

   Note - send ERROR_CONTROL packet with EC = err_code
```

REFERENCES

[1] Karlsson and Vetterli, "Packet Video and Its Integration into the Network Architecture", IEEE J. Sel. Areas Comm., Vol-7, No.5, P.739, June 1989.

[2] Shimizu, Mera and Tani, "Packet Communication Protocol for Image Services on a High-Speed Multimedia LAN", IEEE J. Sel. Areas Comm., Vol-7, No.5, P.782, June 1989.

[3] Douglas Comer, "Internetworking with TCP/IP", Chapter 12 pp.129, Prentice-Hall International Editions.

[4] Richard Stevens, "Unix Network Programming", Chapter 5 TCP/IP pp.197, Chapter 6 Berkeley Sockets pp.258, Chapter 12 TFTP pp.465, Prentice-Hall Software Series.

[5] Johnson and Reichard, "X Window Application Programming", Section III, pp.473, Tech Publications.

[6] Chow and Adachi, "Achieving Multimedia Communications on a Heterogeneous Network", IEEE J. Sel. Areas Comm., Vol-8, No.3, P.348, April 1990.

[7] Poggio et al, "A computer-based multimedia information system", IEEE Computer, vol. 18, pp.92-103, Oct. 1985.

[8] Ichikawa, Aoki and Uchiyama, "High-speed packet switching systems for multimedia communications", IEEE J. Select. Areas Commun., vol. SAC-5, pp. 1336-1345, Oct. 1987.

[9] Nicolaou, "An Architecture for Real-Time Multimedia Communications Systems", IEEE J. Sel. Areas Comm., Vol-8, No.3, P.391, April 1990.

[10] Kositpaiboon et al, "Packetized Radiographic Image Transfers over Local Area Networks for Diagnosis and Conferencing", IEEE J. Sel. Areas Comm., Vol-7, No.5, P.842, June 1989.

[11] Chamzas and Duttweiler, "Encoding facsimilie Images for Packet-Switched Networks", IEEE J. Sel. Areas Comm., Vol-7, No.5, P.857, June 1989.

[12] Netravali, "Picture Coding : A Review", Proceedings of the IEEE, vol. 68, NO. 3, pp.366, March 1980.

[13] Ang, Ruetz and Auld, "Video compression makes big gains", IEEE Spectrum, pp. 16, Oct. 1991.

[14] Gregory Wallace, "The JPEG Still Picture Compression Standard", Communications of the ACM, Vol. 34, No. 4, pp. 31, April 1991.

[15] Takahashi, Sato and Ishii, "A Level-Plane Coding Scheme for Progressive Transmission of Gray-Scale Images", Conf. Rec. IEEE ICC'89, pp.261, March 1989.

[16] Gerd E. Keiser, "Local Area Networks", Chapter 6, pp. 227, McGRAW-Hill International Editions.

[17] Karmouch et al, "A Multimedia Medical Communications System", IEEE J. Sel. Areas Comm., Vol-8, No.3, P.325, April 1990.

[18] Scherr, "Pepperoni and Paperwork", BYTE Magazine, pp.309, Dec 1989.