TRANSMISSION OF JPEG COMPRESSED IMAGES OVER THE WIRELESS CDPD NETWORK

by

MICHAEL KWOKWING WONG

B.Sc., The University of Kansas, 1989

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Department of Electrical and Computer Engineering)

We accept this thesis as confirming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

April 1998

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical and Computer Engineering

The University of British Columbia
Vancouver, Canada

Date April 20, 1998

DE-6 (2/88)

Abstract

Image transmission over bandwidth limited wireless networks is becoming increasingly popular [1]. To transmit an image over a low bit rate wireless network efficiently is not an easy task, because wireless networks normally have a high bit error rate. In this thesis, we propose a Progressive Image Transmission Protocol (PITP) which transmits a JPEG progressively compressed image effectively over wireless networks that support the Internet Protocol (IP), and more specifically the Cellular Digital Packet Data (CDPD) network.

In the first part of the thesis, we will study the CDPD network, the IP network protocol, and the TCP and the UDP transport layer protocols. In the second part of the thesis, we will study the JPEG compression algorithm. In the last part of the thesis, we will discuss the Progressive Image Transmission Protocol (PITP) that is used to transmit an image over the CDPD wireless network. The PITP is a selective re-transmission application layer protocol which employs a JPEG progressive compression algorithm, the UDP transport protocol and a segment re-transmission algorithm. The segment re-transmission algorithm is based on the priority levels of the JPEG scans and the current wireless network conditions. The segment re-transmission algorithm is so flexible that the performance of the PITP system can be adjusted by simply optimizing a few parameters in the algorithm. Simulations were performed to evaluate the efficiency and the effectiveness of the PITP. Our simulation results indicate that the PITP yields a substantially better image transmission rate than that achieved by conventional TCP/IP.

Table of Contents

List of Tables

List of Figures

Acronyms and Abbreviations

AMPS	Advanced Mobile Phone System
ATM	Asynchronous Transfer Mode
BER	Block Error Rate
BPP	Bits Per Pixel
BPS	Bits Per Second
CDPD	Cellular Digital Package Data
CLNP	Connectionless Network Protocol
DCT	Discrete Cosine Transform
FEC	Forward Error Correction
F-ES	Fixed End System
GMSK	Gaussian Minimum Shift Keying
GPS	Global Positioning System
IP	Internet Protocol
IS	Intermediate System
ISDN	Integrated Services Digital Network
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
M-ES	Mobile End System
MDBS	Mobile Data Base Station
MD-IS	Mobile Data Intermediate System
MAC	Medium Access Control
MDLP	Mobile Data Link Protocol
MTU	Maximum Transmission Unit
NEI	Network Entity Identifier
OSI	Open Systems Interconnect
PCI	Protocol Control Information
PITP	Progressive Image Transmission Protocol
PSNR	Peak Signal to Noise Ratio
SDU	Service Data Unit
SLIP	Serial Line Internet Protocol
SNDCP	Sub-Network Dependent Convergence Protocol
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
UDP/IP	User Datagram Protocol/Internet Protocol
WAN	Wide Area Network

INTRODUCTION

With the evolution of computer memory, communications and signal processing, personal computers and powerful modems are becoming more compact and more affordable. Such technological changes affect the way we work and play, increasing our mobility and flexibility. People can work at home, and they can work in very remote locations and at random times, thanks mainly to advances in computer and communication technologies. However, unless a person's computer is connected via an RF link, his mobility would still be limited as he would have to work at a location connected to a public telephone network. Therefore, as wireless communication technologies improve, our mobility increases substantially. For example, one can now use an RF modem to connect his computer to his corporate network through a CDPD wireless network that can provide baud rates of up to 19,200 baud.

While most existing wireless networks were designed for the transmission of data, such as English text and transaction records, many of today's applications also require the transmission of image files over wireless networks. Due to the relatively large number of bits required to represent raw images, e.g. 256 kbytes, the transmission of such images is not practical using today's wireless data networks, which often have very small throughputs, e.g. below 2 kbits/sec. Therefore, optimized compression algorithms and transport methods must be employed if images are to be transmitted efficiently over wireless networks.

In the last few years, much research has been done in the area of progressive image transmission. Such research contributions include progressive image transmission using the lapped orthogonal transform [2], progressive image transmission by transform

coefficient residual error quantization [3], and progressive image transmission by content-driven Laplacian pyramid encoding [4]. However, these algorithms are optimized only to the application level, and they do not integrate the transport layer and network layer protocols. For example, if any part of an image file is lost during the transmission, these algorithms do not provide effective methods to recover the errors before reaching the application layer.

In this thesis, we propose a Progressive Image Transmission Protocol (PITP) which transmits a JPEG progressively coded image file over low bandwidth wireless IP networks. The PITP is a selective re-transmission protocol. The PITP will send the coded image in several scans by using UDP/IP. If any scans of the JPEG coded image are lost in the transmission, the PITP will evaluate the missing scans to determine, based on the scan priorities and the current channel condition, if the missing scans should be re-transmitted.

Chapter 1 Cellular Digital Package Data (CDPD)

There are many public wireless networks supporting IP protocol. Two examples are the CDPD network and the RAM Mobile Data network [5]. In this thesis, we will use the CDPD network as the wireless network model to evaluate the efficiency of the PITP protocol. The CDPD network is a little over four years old and is already proving to be a popular digital enhancement to the existing cellular telephone network. CDPD is a digital wireless transmission protocol that enables users to transmit packets of data over the Advanced Mobile Phone System (AMPS) [6]. When compared to other public wireless networks, such as RAM and ARDIS, the performance of the CDPD network is clearly superior. In term of throughput, CDPD maintains a 19.2 kbits/sec bit rate with an effective throughput close to 10 kbits/sec. RAM operates at an 8 kbits/sec bit rate with an actual throughput of only 2.4 to 4.8 kbits/sec, while ADRIS operates at a 19.2 kbits/sec bit rate in large cities and a 4.8 kbits/sec bit rate in other areas with an actual throughput of 2.4 to 8 kbits/sec. CDPD works by slipping packets of data into frequencies that are momentarily unoccupied by voice transmission. However, voice transmission has priority over digital packets. When a new voice call is initiated, the CDPD transmission "hops" to an unused channel. Because over 16,000 AMPS cell sites exist in North America, CDPD can provide very wide coverage to its subscribers [7].

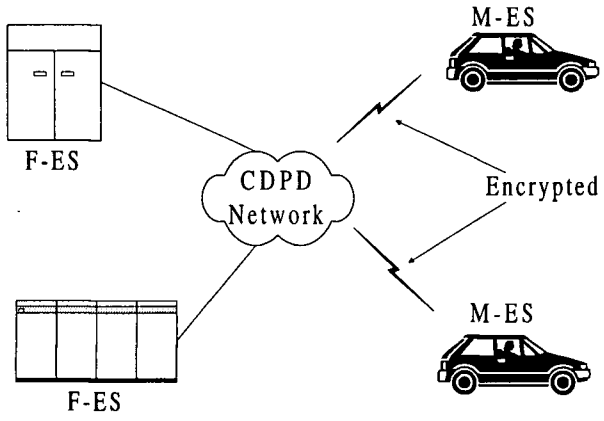
Among advantages of the CDPD network,

- CDPD supports the Internet Protocol (IP) which allows users to connect their computers to many public IP networks, such as the Internet, and it also allows easy integration to existing corporate IP networks.

- Users can setup a prototype network very quickly in a cost-effective way. A user needs only a CDPD modem and an account with the local service provider.
- The CDPD protocol uses the CCITT-V.42 bis data compression scheme to allow more efficient data transfer.
- Data will be encrypted when it is transmitted over the CDPD network.
- CDPD provides a very sophisticated error correction technique. CDPD employs Reed Solomon (63, 47) channel coding which can correct up to 8 bits of errors per FEC block.
- CDPD is an open system architecture that allows support by multiple vendors, resulting in a competitive CDPD market.

A high layer picture of the CDPD Network is shown in Figure 1.

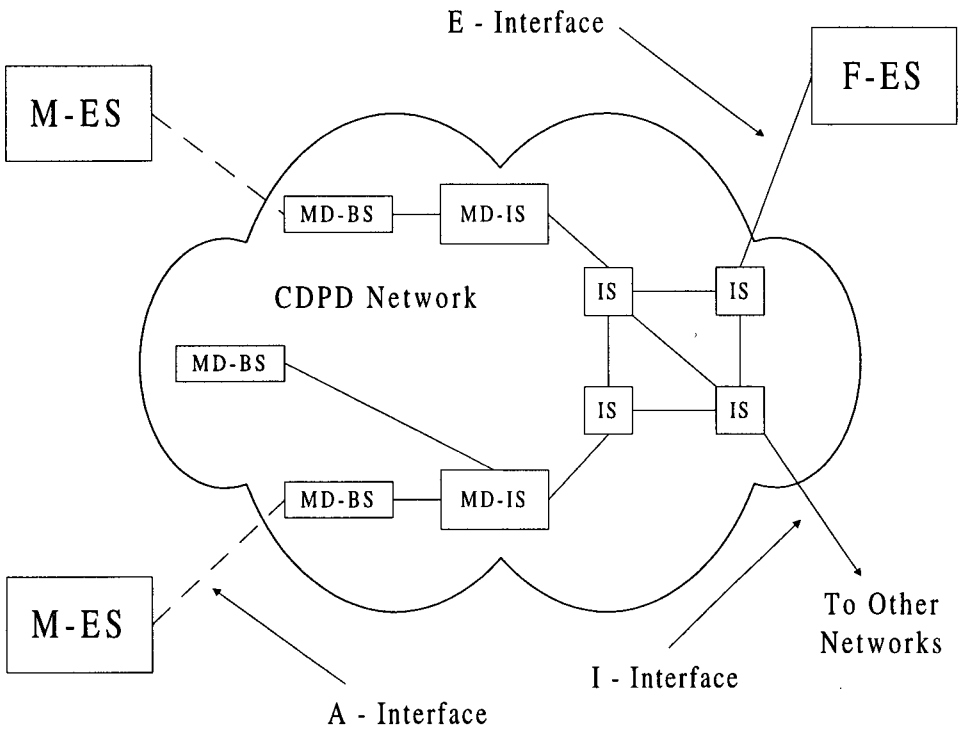
Figure 1: CDPD Network



M-ES : Mobile End System
F-ES : Fixed End System

The CDPD Network Layout is shown in the Figure 2.

Figure 2: CDPD Network Layout



The CDPD network is intended to provide end users with a means to make use of the existing corporate network resources where no physical connections to these resources are available. The user can communicate with his corporate network via the wireless air link interface (A-Interface). All data passing over the air link interface is encrypted as part of the CDPD protocol. The encryption key management is based on the Diffie and Hellman Electronic Key Exchange procedure. After a user makes the connection to his corporate network through the CDPD network, he will be able to access all the databases and services provided by his corporate network as if he were physically hardwired to the network. Additionally, different CDPD networks from different service

providers can be integrated together to become one big CDPD network so that a user can roam from one CDPD network to another, providing flexibility to the end users. Moreover, some CDPD networks provide additional services to the end users, such as:

- Directory Service: A user can look up another end user Network Entity Identifier (NEI) so that he can send messages to him.
- Message Handling Service: The CDPD network can store for, and forward messages to, end users.

1.1 CDPD Network Elements

The CDPD network consists of the following elements as shown in Figure 2.

- Mobile End System (M-ES): The M-ES can be any computing device which has a CDPD modem installed. The M-ES communicates with the Mobile Data Base Station (MDBS) via the A-Interface (Air Link Interface). During the authentication process to the CDPD network, an M-ES has to provide a valid Network Entity Identifier (NEI) which is a typical IP address. Each M-ES may support either full duplex or half duplex operation. However, the CDPD network is a full duplex network.
- Fixed End System (F-ES): The F-ES can be classified into either an External F-ES which is the user's host computer or an Internal F-ES which is the host computer from the CDPD service provider. The F-ES connects directly to the Intermediate System (IS) of the CDPD network via the E-Interface. The E-Interface can be in any of the following forms which vary from service provider to service provider,

- ISDN
- DataPac X.25
- ATM
- Hyperstream Frame Relay

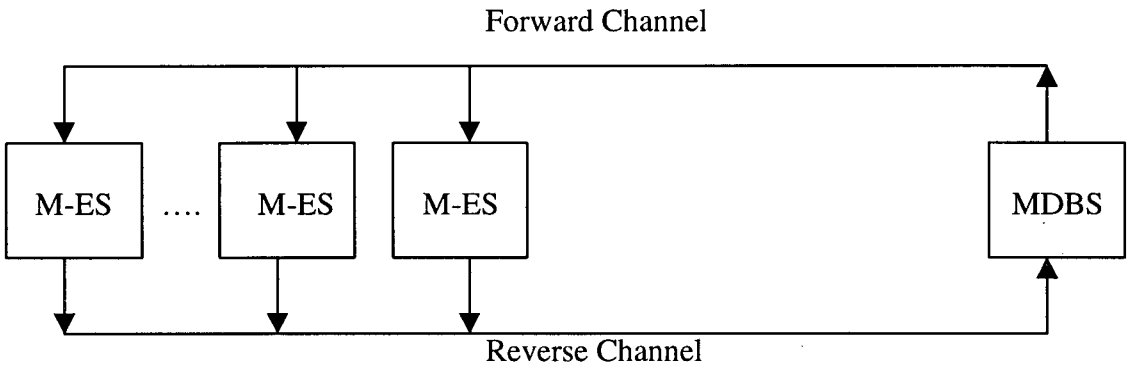
- Mobile Data Intermediate System (MD-IS): The MD-IS performs the routing function of the data package based on the current location of each M-ES. Each MD-IS is responsible for a part of the CDPD network coverage area, and each M-ES will be registered to a Home Area MD-IS in which he will spend most of his time. The Home Area MD-IS maintains a database of the current locations (cells) of its member M-ESs. When a data package is sent to an M-ES, it will first send to the M-ES's Home Area MD-IS. If the M-ES is not located in the coverage area of his Home Area MD-IS, his Home Area MD-IS will forward the data package to the Serving MD-IS based on the current location of the M-ES. When an M-ES visits a new area, the Serving MD-IS will inform the M-ES's Home Area MD-IS of the new location of his member M-ES.
- Mobile Data Base Station (MDBS): The MDBS is responsible for detailed control of the allocation of radio resources. When data packages arrive at a cell site from a F-ES or an M-ES, the MDBS will relay the data package onto a cellular CDPD forward channel for transmission toward the M-ES. The MDBS will also keep monitoring all his radio resources to make sure that they are properly used. If a MDBS finds an M-ES is not working properly, for example if the transmission power is too high, the MDBS will force the M-ES to sign off from the CDPD network. The MDBS will also inform all other M-ESs located in his coverage area to switch channels when he finds out that the current channel will be used for voice transmission. This is because

voice transmission will always have higher priority than transmission of digital data packages.

1.2 The Model of CDPD Channel Stream

Each CDPD channel consists of a forward channel and a reverse channel as shown in Figure 3. The forward channel is for the MDBS to communicate with all the M-ESs which are located in its coverage area, and the reverse channel is for the M-ESs to communicate with the MDBS. The reverse channel will be shared by all the M-ESs using the same cellular channel.

Figure 3: Model of the CDPD Channel Stream

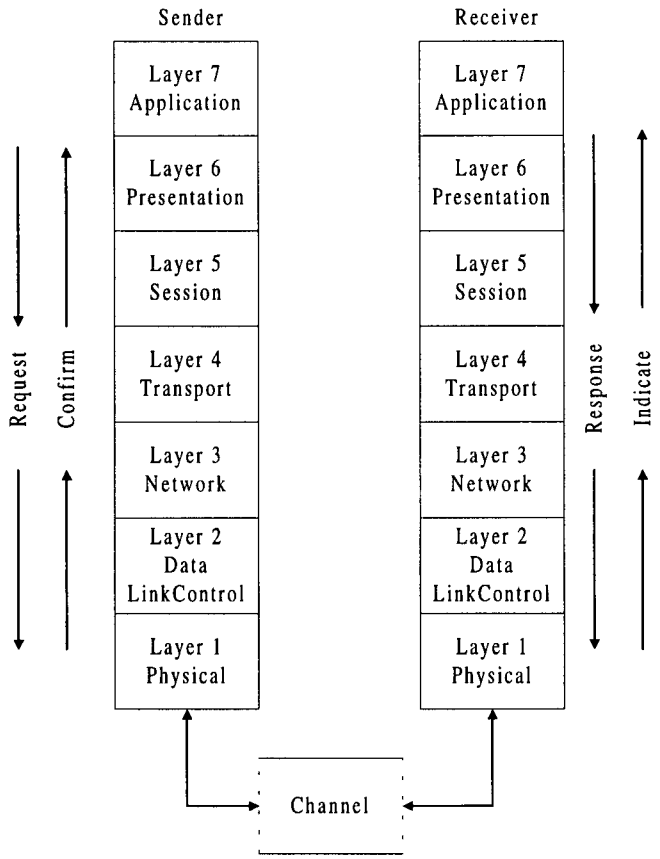


1.3 The CDPD Air Interface Profile

The CDPD Air Interface Profile is similar to the Open Systems Interconnect (OSI) model. The OSI model consists of seven layers of processes. They are the Application Layer, the Presentation Layer, the Session Layer, the Transport Layer, the Network Layer, the Data Link Layer and the Physical Layer as shown in Figure 4. Each layer provides a different service to the communication process. When data passes from

layer 7 down to layer 1, a control header from each layer is prepended to the data stream, and only the same layer on the other end of the communication channel will be able to make use of the control header information.

Figure 4: OSI Model



The actual information that is processed by an OSI layer is referred to the Service Data Unit (SDU). When a layer has completed its processing task, the original SDU will be passed to the next lower layer with a control header appended to it. The added control header is referred to as the Protocol Control Information (PCI). For the CDPD network, the Air Interface Profile consists of only 5 layers. They are the Network Layer, the Sub-Network Layer, the Link Layer, the Medium Access Control Layer and the Physical

Layer as shown in Figure 5. The Air Link Protocol Profile for the CDPD Network is shown in Figure 6.

Figure 5: CDPD Air Link Layering and Protocol Diagram

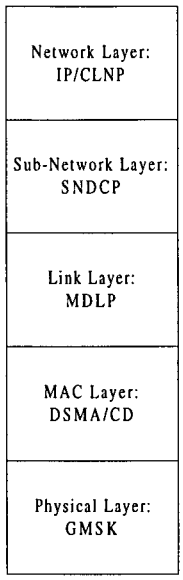
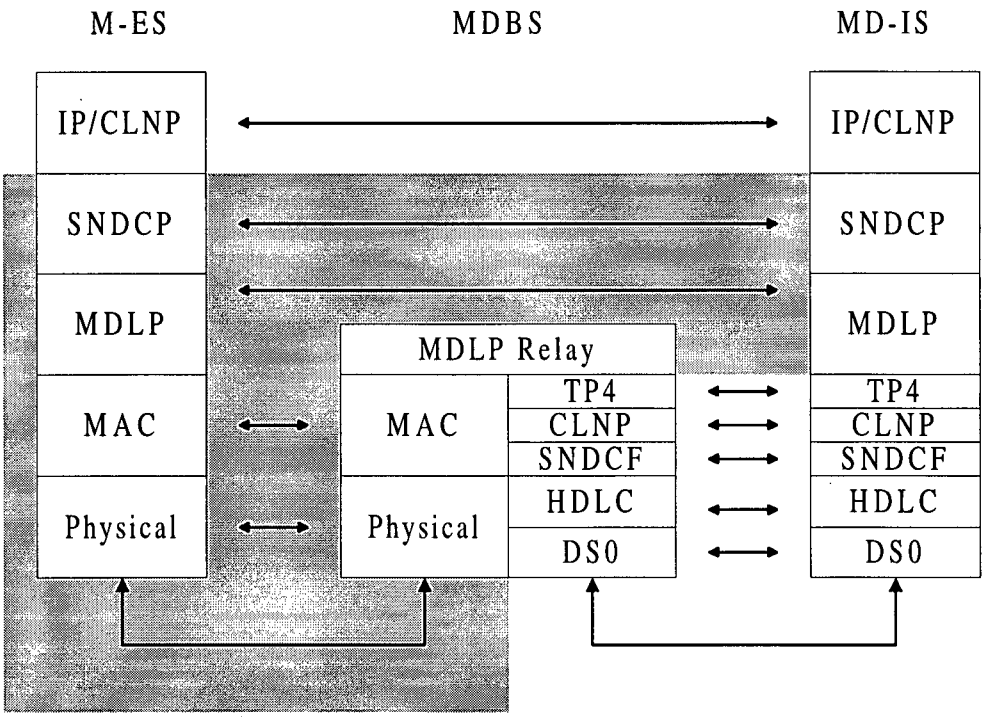


Figure 6: CDPD Air Link Protocol Profile



Shaded Area: Scope of the Air Link

Figure 6 shows the processes used between the M-ES and the MD-IS. The shaded area shows the air link interface part of the CDPD network. The physical connection between the MD-IS to the rest of the CDPD network varies from one service provider to another service provider. As shown in Figure 6, the CDPD network supports the IP network protocol. As data packages enter the Sub-Network layer, up to 2048 bytes of network layer data will be compressed by using the CCITT-V.42 bis compression algorithm. The compression algorithm will compress the data part and the network protocol header part separately. The reason that the network protocol header and the data part will be compressed separately is that the network protocol header does not change very much from one package to the next. Therefore, compression efficiency increases more when the data part and network protocol header are compressed separately. A user can optionally enable and disable the CCITT-V.42 bis compression feature during transmission. After the data packages are compressed, the output will be separated into segments, which are then encrypted. The encrypted data segments will be passed to the Data Link Layer, where they are converted into frames. The framed data will then be passed to the Medium Access Control (MAC) Layer. The MAC Layer will format the framed data into the CDPD reverse channel format and then pass the formatted bit stream to the Physical Layer. The Physical Layer accepts the formatted bit stream and transforms them into modulated waveforms for transmission to the MDBS on the reverse channel. Gaussian Minimum Shift Keying (GMSK) is used for data transmission at a rate of 19.2 kbps. The bandwidth of each RF channel is 30KHz.

1.4 The Reverse Channel Format

The MAC Layer formats the framed data from the Data Link Layer into the reverse channel format. First, a 38-bit dot sequence 101010...1010 is added to the data stream. The dot sequence is mainly for bit timing recovery at the receiver. Then, a 22-bit synchronization word and an 8-bit channel color code are added to the data stream. The synchronization word provides a reference marker within the reverse channel bit stream. The channel color code is for the MDBS to measure the co-channel interference. This is because every MDBS will have a different color code. The MDBS will send the channel color code to all its M-ESs on the forward channel. When an M-ES sends data to the MDBS, the MAC Layer in the M-ES will echo the channel color code back to the MDBS on the reverse channel. When the MDBS receives a data byte stream from the M-ES, it will check the channel color code from the M-ES. If the MDBS discovers that the channel color code being echoed back from an M-ES is not the same as the one it has put on the forward channel, the MDBS will discard all the received blocks. The MDBS will then stop all the M-ESs from using the reverse channel by declaring the channel as being busy. On the other hand, if an M-ES discovers that the channel color code received on the forward channel suddenly changed, the M-ES will stop transmitting data on the current channel and will search for another channel to continue the transmission.

The resulting bit stream will be channel encoded by using the Reed Solomon (63, 47) code. The Reed Solomon (63,47) code can correct up to 8 bits of transmission errors. Each symbol is 6-bit long. Therefore, each block is 385 bit long. Within each Forward Error Correction (FEC) block, a 7-bit continuity indicator will be interleaved among the data symbols, which will be used to indicate if the current block is the last transmission

block or not. Specifically, one bit of the continuity indicator is found in every ninth 6-bit symbol. If the continuity indicator is 0000000, this means that it is the final block. If the continuity indicator is 1111111, this means that more blocks are coming. After transmitting the last block within a transmission burst, the M-ES will ramp down its transmitter within 2.0 millisecond.

1.5 The Forward Channel Format

The forward channel is for the MDBS to communicate with its M-ESs. The data stream from the MD-IS will be encrypted in the Sub-Network Layer. The encrypted data will then be passed to the Data Link and MAC Layers. Finally, it will be put on the forward channel on its way to the M-ES. All M-ESs which are located in the same coverage area of a MDBS will receive the encrypted data. However, only the destined M-ES will have the key to interpret the data.

The data stream from the Data Link layer will first be added with a 35-bit synchronization word that provides a reference marker within the forward channel bit stream. An 8-bit channel color code will also be added to the forward channel data stream, which will be used to determine co-channel interference as described in Section 1.4 . Since the reverse channel is shared by all the M-ESs that are located in the same cell, a 5-bit reverse Channel Busy/Idle Status Flag is found in the forward channel data stream to indicate if the reverse channel is idle or not. If the Reverse Channel Busy/Idle Status Flag is 00000, this means that the reverse channel is idle, and any M-ES can use the reverse channel to send data to the MDBS. If the Reverse Channel Busy/Idle Status Flag is 11111, this means that the reverse channel is busy.

Following the Reverse Channel Busy/Idle Status Flag is a 5-bit Reverse Channel Decode Status Flag. When an M-ES sends data on the reverse channel, the M-ES will also monitor the Reverse Channel Decode Status Flag received from the forward channel. The Reverse Channel Decode Status Flag tells the M-ES that the previous block sent by the M-ES was or was not decoded successfully by the MDBS. If the Reverse Channel Decode Flag is 00000, this means that the last FEC block that the M-ES sent out was decoded successfully by the MDBS. The M-ES will keep sending all the FEC blocks while monitoring the Reverse Channel Decode Flag. If the Reverse Channel Decode Flag is 11111, this means that the last FEC block could not be decoded by the MDBS. The M-ES will stop transmitting, and it will then wait for a random amount of time and then start transmitting again the FEC block that was not decoded correctly. The resulting data stream will then be channel encoded using the Reed Solomon (63,47) channel coder.

Finally, all Reed Solomon encoded bits will be exclusive-ored with a 378 bit Pseudo Random Number (PN) sequence to avoid having a long string of binary ones or zeros. This is because certain types of modulators and demodulators, such as the phased lock loop, may not be able to accurately track these long strings of ones and zeros. The initial PN sequence is 111000101. The generator polynomial is

$$g(D) = D9 + D8 + D5 + D4 + 1.$$

1.6 Principles of Data Transfer

The MDBS senses the reverse channel for the presence of data and sets the Reverse Channel Busy/Idle Status flag on the forward channel. The M-ES transmission access for the reverse channel is managed through a slotted non-persistent Digital Sense Multiple Access with Collision Detection (DSMA/CD) scheme. If the Reverse Channel

Busy/Idle Status flag indicates that the reverse channel is busy, the M-ES will wait for a random number of micro-slots before it senses the channel status again. Each micro-slot is 60- bit long. After time expiration, the M-ES will sense the Reverse Channel Busy/Idle Status flag on the forward channel. If the reverse channel is idle, the M-ES will start data transmission.

When the reverse channel is idle, any M-ES can send data on the reverse channel. Therefore, there is a chance that more than one M-ES will transmit data on the reverse channel at the same time when the reverse channel is idle. If there is a transmission error in a FEC block or more than one M-ES are transmitting at the same time, the MDBS will not be able to decode the data on the reverse channel. The MDBS will set the Channel Decode Status flag to 11111 on the forward channel and discard all blocks subsequently received in the burst. In the event of decoding failure, the M-ES will re-schedule re-transmission. A half duplex M-ES will re-schedule all blocks for re-transmission. A full duplex M-ES will determine the affected frames. The first block to be re-transmitted is the one which was not decoded correctly. It will be presumed that all those preceding have been decoded successfully.

1.7 Data Link Layer Frame Re-transmission

As discussed in Section 1.6 , if an FEC block cannot be decoded by the MDBS, the FEC block will be re-transmitted again. The FEC block re-transmission occurs at the CDPD MAC Layer. However, if a whole FEC block is lost, the MAC layer will not be able to detect the error. Rather, the Mobile Data Link Layer will detect the error and correct it.

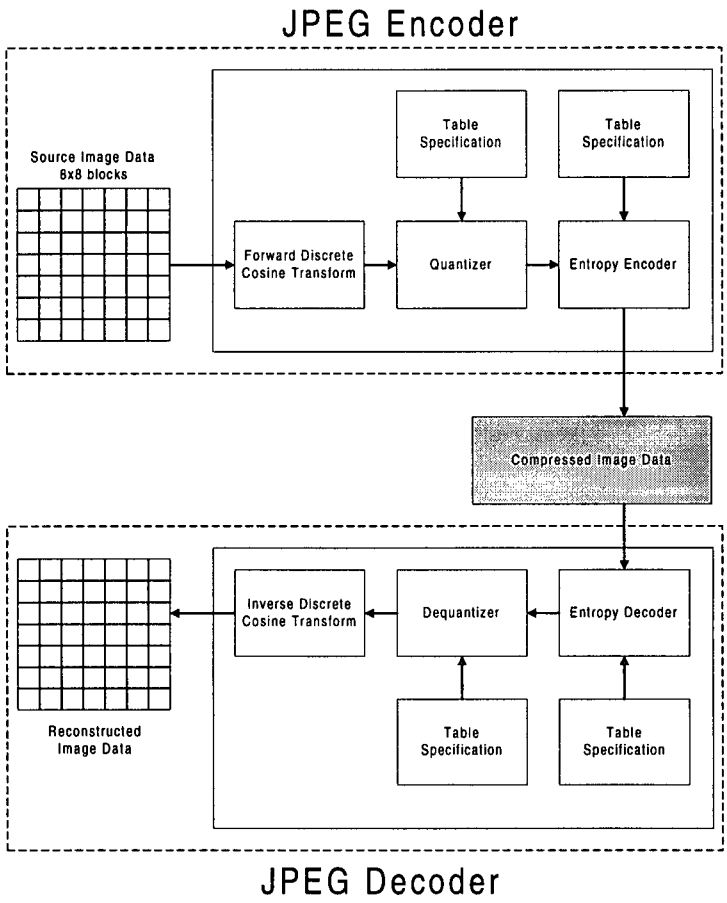
As described in Section 1.3 , the Data Link Layer will divide the encrypted data obtained from the Sub-Network Layer into frames. The Mobile Data Link Protocol will assign a frame number to each frame as well as start a time-out timer for the expected acknowledgement from the MD-IS Data Link Layer. When the MD-IS Data Link Layer receives a frame, it will send a frame acknowledgement back to the M-ES Data Link Layer to confirm that the particular frame has been decoded successfully. If an FEC block is lost, then the frame cannot be re-assembled, and the MD-IS will then not send an acknowledgement for that particular frame to the M-ES Data Link layer. When the M-ES Data Link Layer frame acknowledgement timer expires, the same frame will be re-transmitted again by the M-ES Data Link Layer. Therefore, the CDPD network provides a very reliable mechanism to re-cover from virtually all types of transmission errors.

Chapter 2 The JPEG Compression Standard

The JPEG compression standard [8] has become very popular in the past few years, and JPEG compliant encoders/decoders are now used almost everywhere for transmission of still images. JPEG supports both the sequential and progressive modes. While JPEG's sequential mode is, by far, more popular, JPEG's progressive mode has received much attention in the past 2-3 years. Many image transmission softwares [9], such as Netscape 3.x, can decode JPEG progressively compressed images. The JPEG progressive mode will be used in our Progressive Image Transmission Protocol (PITP) system.

A general block diagram of a JPEG encoder/decoder is shown in Figure 7 [10].

Figure 7: Block Diagram JPEG Encoder and Decoder



In JPEG, an image can be compressed in four different modes [11]:

- Lossless encoding: The reconstructed image is bit-by-bit the same as the original image. However, the compression ratio is much less than in the other three modes. Because of its low compression ratios, the lossless compression mode is not appropriate for wireless image transmission.
- Hierarchical encoding: The image is compressed in multiple resolutions so that a lower resolution image may be accessed and displayed without decompressing the high resolution image.
- Sequential encoding: An image is divided into 8x8 blocks. Then each block will be forward DCT encoded, quantized and entropy encoded. The encoder will process each 8x8 block from left-to-right and top-to-bottom.
- Progressive encoding: An image is divided into 8x8 blocks and encoded in the same order as in the sequential mode. However, after DCT transformation and (optional) quantization, all the DCT coefficients will be grouped into a number of scans, and then each scan will be entropy encoded separately. The progressive encoded image is good for transmission on a low bandwidth channel. This is because a user can view the image in multiple coarse-to-fine stages. The final image quality is dependant on the number of scans received.

To summarize, the sequential, progressive and hierarchical modes are more suitable for wireless image transmission, mainly because they yield higher compression ratios.

The reconstructed image quality for different bit rate ranges is known to be:

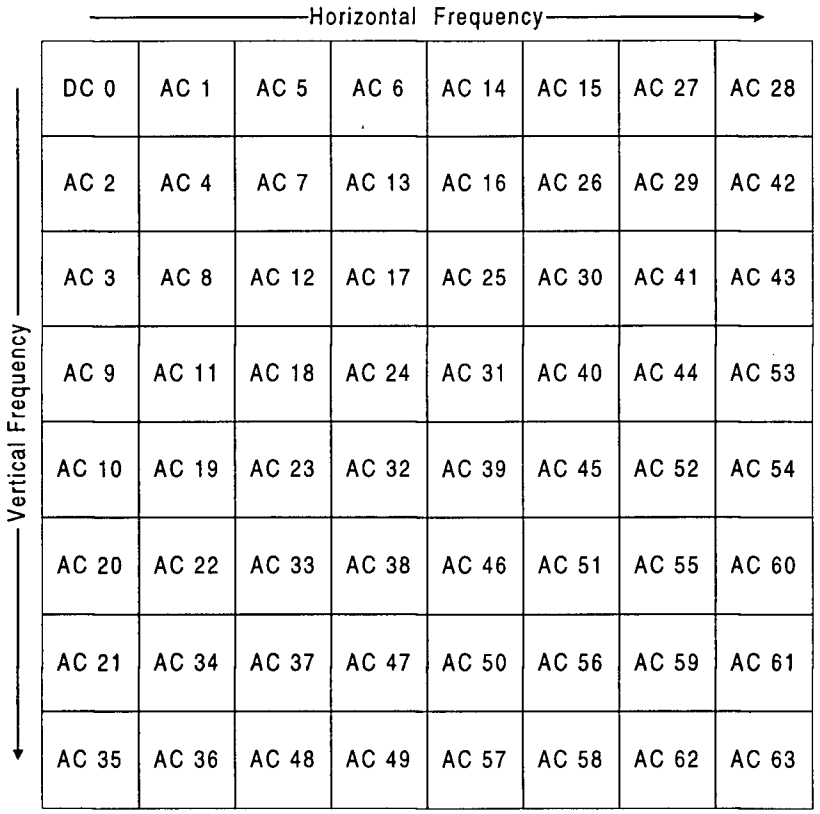
- 0.25 ~ 0.5 bpp: moderate to good quality
- 0.50 ~ 0.75 bpp: good to very good quality
- 0.75 ~ 1.5 bpp: excellent quality

- 1.5 ~ 2.0 bpp: indistinguishable image (visually lossless)

2.1 JPEG Sequential Mode

In the JPEG sequential mode, an image is divided into 8x8 pixel blocks. Then, each 8x8 block is forward DCT transformed. As a result, 64 DCT coefficients are generated for each 8x8 block as shown in Figure 8.

Figure 8: DCT Coefficients

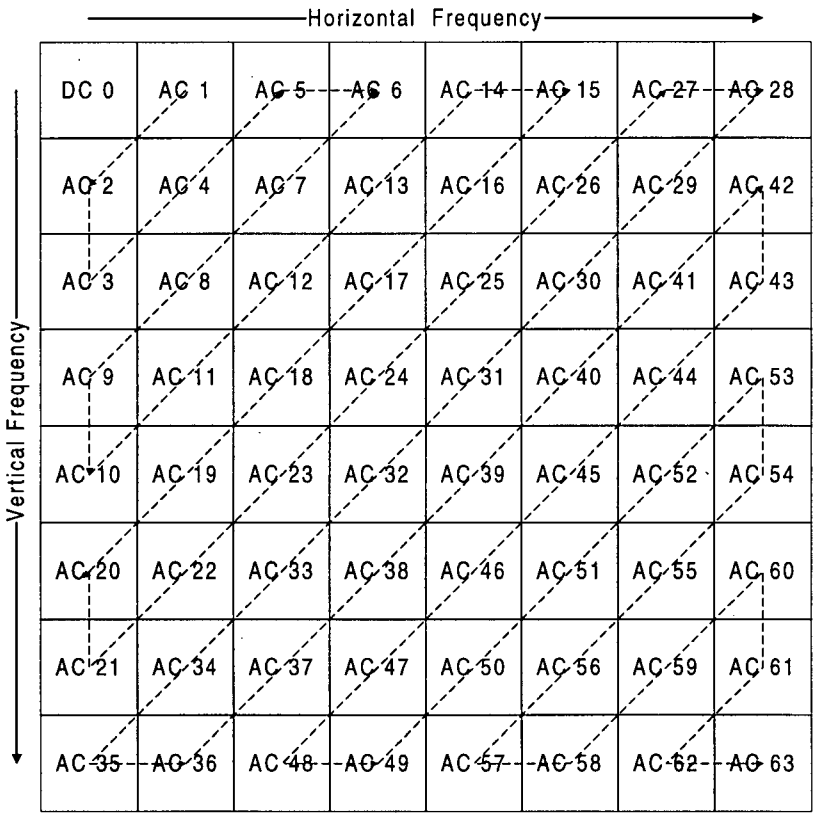


The top left coefficient is called the DC coefficient, and the remaining 63 coefficients are called the AC coefficients. All 64 DCT coefficients are quantized by using a 64-element quantization table specified by the compression kernel. The quantization phase is the largest source of loss in the compression system. However, if

the quantization table is designed appropriately, the loss in information can be made visually unperceivable. The quantization phase reduces the accuracy of the coefficients, hopefully contributing little or nothing to the quality of the reconstructed image. The JPEG standard provides a default quantization table. However, an application can define its own quantization table.

After quantization, all 63 AC coefficients will be arranged into a "zig-zag" sequence for entropy encoding [12] as shown in Figure 9.

Figure 9: Zig-zag ordering of AC coefficients



This "zig-zag" ordering will place the low frequency AC coefficients which are located closer to the top-left corner before the high frequency AC coefficients which are located closer to the lower-right corner. For a typical 8x8 block, most of the AC

coefficients will have values of zero or near zero, and the whole 8x8 block can be compressed to a very small size.

The last phase of the JPEG compression is entropy coding, which provides additional compression by encoding the quantized DCT coefficients into a more compact form. The baseline JPEG standard employs a Huffman-like encoder. JPEG encodes the AC coefficients and the DC coefficient differently.

Entropy encoding of the AC coefficients consists of two steps. The first step is to convert the AC coefficients into intermediate symbols. The second step is to convert each intermediate symbol into a binary sequence by using a set of Huffman-like codes. The intermediate symbol consists of Run Length, and Size and Amplitude of the non-zero coefficient [13][14]:

- Run Length is the number of consecutive zero-valued AC coefficients preceding the non-zero AC coefficient. The Run Length symbol is 4-bit long. If there is more than 15 zeros preceding the non-zero valued AC coefficient, then the Run Length and the Size symbols are (15,0), and the encoder will start counting zeros again.
- Size is the number of bit that is required to represent the Amplitude value in binary format.
- Amplitude is the amplitude of the non-zero AC coefficient.

For example, if the AC coefficients are 0,0,0,0,0,0,0,123, then the symbols will be 7,8,123. The number '7' implies that there are seven zeros preceding the nonzero AC coefficient '123'. The number '8' indicates that 8 bits are needed to represent the nonzero AC coefficient '123'. The number '123' is the amplitude of the nonzero AC coefficient. The symbols 7,8 and 123 are then variable-length encoded.

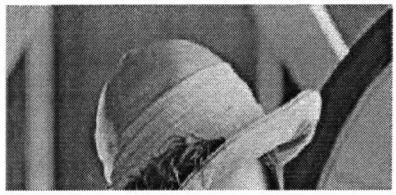
The DC coefficient is differentially encoded. The reason is that the DC coefficient is actually the average value of each 8x8 block, and there is usually a strong correlation between the DC coefficients of the adjacent 8x8 blocks. Therefore, coding the difference of DC coefficients between the adjacent blocks will yield higher compression performance.

The difference between decoding a JPEG sequentially encoded image and a JPEG progressively encoded image is that while decoding a JPEG progressively encoded image, a user can see a rough approximation of the original image during the first stage, while still being able to reproduce higher quality images. A JPEG sequentially encoded image can only be encoded to reproduce partial information of the full image in one stage. Figure 10 shows the decoding process of a JPEG sequentially encoded image.

Figure 10: Decoding process of a JPEG sequentially encoded image

1

2



3

4

In the sequential mode, all 63 AC coefficients of an 8x8 block are entropy encoded sequentially at the same time. Therefore, the whole 8x8 block will not be decoded if any data inside an 8x8 block is corrupted during transmission.

2.2 JPEG Progressive Mode

The JPEG progressive mode is used to encode an image in multiple scans. A user can decode a rough approximation of the original image in the first few stages. As more scans are decoded, the reproduction image quality is improved. In the past 2-3 years, the JPEG progressive mode has received much attention in many image transmission

applications. For example, Netscape 3.x already supports JPEG progressive decoding of still images. In many applications, when a large image is transmitted over the network, the decoding process may take several minutes. In this case, the JPEG progressive mode is much more suitable, a user can quickly decode an acceptable quality image, while still decoding higher quality images if needed.

The compression kernel for the JPEG progressive mode is very similar to that of the JPEG sequential mode except that the DC coefficient is not differentially encoded with the left-adjacent 8x8 block. After transformation and (optional) quantization, the DCT coefficients are stored in a buffer for grouping. After grouping, each group of the DCT coefficients will be entropy encoded, resulting in scans. In the standard, simple methods are suggested to group the DCT coefficients. They are spectral selection and successive approximation.

In the successive approximation method, each scan consists of some bits of the DCT coefficients. Since the most binary-wise significant bit of all DCT coefficients normally carries the most information, such bits are encoded first and the output is sent as the first scan. The less significant bits of all coefficients are encoded in the subsequent scans. For example, an image can be grouped into 4 scans as follows:

- scan 1: all bits of the DC coefficient
- scan 2: bit 7 (most binary-wise significant bit) to bit 6 of all the AC coefficients
- scan 3: bit 5 to bit 3 of all the AC coefficients
- scan 4: bit 2 to bit 0 of all the AC coefficients

Figure 11 shows the decoding process of a JPEG progressively encoded image by the successive approximation method.

Figure 11: Decoding process of a JPEG progressively compressed image by the successive approximation method

Scan 1:

Scans 1 – 2:

Scans1 – 3:

Scans 1 – 4:

In the spectral selection method, the DCT coefficients are grouped into several spectral bands. Each band is then entropy encoded, resulting in a scan. Typically, the lower frequency DCT coefficients are grouped together, entropy encoded and sent first, so when the receiver decodes the first scan, the user can see the low-frequency information of the image. The high frequency DCT coefficients are then grouped,

resulting in other bands that can be transmitted if needed. For example, the image shown in Figure 12 is grouped into 10 spectral bands as follows:

- band 1: DC coefficient only
- band 2: AC 1 (first AC coefficient along the zig-zag direction) and AC 2
- band 3: AC 3 to AC 5
- band 4: AC 6 to AC 9
- band 5: AC 10 AC 14
- band 6: AC 15 to AC 20
- band 7: AC 21 to AC 27
- band 8: AC 28 to AC 35
- band 9: AC 36 to AC 42
- band 10: AC 43 to AC 63 (last AC coefficient in the zig-zag direction)

Figure 12 shows the decoding process of a JPEG progressively encoded image by the spectral selection method.

Figure 12: Decoding process of a JPEG progressively encoded image by the spectral selection method

Stage 1: DC coefficient only
File size up to this stage: 3556 bytes

Stage 2: DC – AC 2
File size up to this stage: 6337 bytes

Stage 3: DC – AC 5
File size up to this stage: 8771 bytes

Stage 4: DC – AC 9
File size up to this stage: 10186 bytes

Stage 5: DC – AC 14
File size up to this stage: 12287 bytes

Stage 6: DC – AC 20
File size up to this stage: 14534 bytes

Stage 7: DC – AC 27
File size up to this stage: 19948 bytes

Stage 8: DC – AC 35
File size up to this stage: 24185 bytes

Stage 9: DC – AC 42
File size up to this stage: 26841 bytes

Stage 10: DC – AC 63
File size up to this stage: 28110 bytes

The data contained in each image shown in Figure 12 is cumulative. For example, stage 1 contains band 1 only, and stage 2 contains band 1 and band 2. As shown in Figure 12, more details of the decoded image appear as more scans are decoded. Moreover, note that after stage 5 has been decoded, the decoded image contains most of the important information of the original image, which is obtained after scan 10 has been decoded. Therefore, if some of the less significant scans, such as scan 6 to scan 10, are corrupted or missing, the user can still have a decoded image with acceptable quality. In addition to this, some of the post image processing techniques can be applied to the decoded image to enhance the image quality. Looking at the importance of each scan from a different viewpoint, Figure 13 shows the decoded images when the individual scans have been corrupted.

Figure 13: Decoded images when a scan has been lost or corrupted

Scan 1 is missing
PSNR: 15.06 dB

Scan 2 is missing
PSNR: 30.33 dB

Scan 3 is missing
PSNR: 35.08 dB

Scan 4 is missing
PSNR: 41 dB

Scan 5 is missing
PSNR: 35.32 dB

Scan 6 is missing
PSNR: 35.18 dB

Scan 7 is missing
PSNR: 34.33 dB

Scan 8 is missing
PSNR: 36.79 dB

Scan 9 is missing
PSNR: 40.03 dB

Scan 10 is missing
PSNR: 43.5 dB

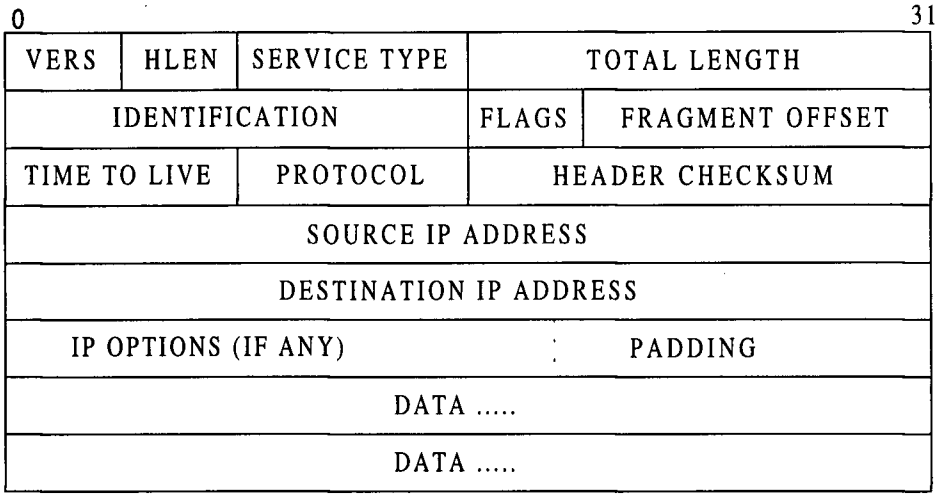
As shown in Figure 13, the first few scans are more important and they affect the reproduction image quality substantially more than the last few scans. Figure 13 shows that the first scan, which contains all the DC coefficients, is the most important scan. Without this scan, the decoded image is almost not usable. Scans 1 to 5 are also important, but they do not contribute as much as the DC coefficient. If they are missing, we can probably use some image post processing techniques to enhance the received image. Scans 6 to 10 are not important. If they are missing, the decoded image still appears visually the same as the original image. Finally, as discussed later, the PITP supports both the spectral selection and the successive approximation methods, or a combination thereof.

Chapter 3 UDP/IP vs TCP/IP

The proposed PITP employs the IP protocol as the network layer protocol. The IP network layer provides a connectionless and unreliable delivery service to the transport layer [15][16]. The IP network layer will treat each IP datagram as independent. Each IP datagram contains its source address, destination address and a checksum that only covers the IP datagram header. All other data security checks, such as checksum for the data part and package acknowledgement, have to be provided by the transport layer. If the IP header is found to be erroneous, the network layer will discard the whole IP datagram, and the IP network layer will assume that the transport layer or the application layer at the sender side will recover from the errors.

Most networks can handle a maximum package size called the Maximum Transmission Unit (MTU). If a package is longer than the MTU, the IP network layer will separate the package into two or more smaller packages. The process is called Fragmentation. On the other hand, the IP does not impose a maximum package size, but states that all networks shall be able to handle datagrams of at least 576 bytes. When a datagram is fragmented, all the fragments will have the same header, which is basically copied from the original datagram. Each of the resulting fragments will be treated as a normal IP datagram. However, if one of the fragments is lost during transmission, the original datagram is considered lost. This is because the IP does not re-transmit any loss fragments, and all the related fragments will be discarded by the destination host's network layer. The IP Datagram format is shown in Figure 14.

Figure 14: IP Datagram Format



The IP datagram header is a minimum of 20 bytes long. Each section of the header is defined as follows:

- VERS: the version of the IP protocol.
- HLEN: the header length of the IP datagram expressed in 32-bit words. The IP datagram header is 20 bytes long if the IP Options field is empty. The length does not include data field. All fields in the header are fixed in length except for the IP Options and the Padding fields. The most common header, which contains no options and no padding, has a header field length that is equal to 5.
- SERVICE TYPE: the 8-bit Service Type field is further re-defined to a 3-bit Precedence field and a 4-bit Type of Service field. The last bit is not used. The Precedence field indicates the nature and priority of the datagram. The possible values of the Precedence field are:

  000: Routine
  001: Priority
  010: Immediate
  011: Flash

100: Flash Override
101: Critical
110: Inter-Network Control
111: Network Control

The possible values of the Type of Service are:

1000: Minimize delay
0100: Maximize throughput
0010: Maximize reliability
0001: Minimize monetary cost
0000: Normal Service

- TOTAL LENGTH: the total length of the datagram in bytes that include the IP datagram header and the data.
- IDENTIFICATION: a unique number assigned by the source IP layer to help to assemble a fragmented datagram. Fragments of a datagram must have the same identification number.
- FLAGS: various control flags.
- FRAGMENT OFFSET: used with a fragmented datagram, so the destination host can use it to re-assemble the fragmented datagram.
- TIME TO LIVE: this value specifies the number of seconds that the datagram is allowed to travel. The parameter prevents any undeliverable datagram from staying in the network forever. When a datagram passes a router, the router will subtract his processing time from this field. When a router finds a datagram with a value of 0 in this field, the router will not route the datagram and will discard it.
- PROTOCOL: this field shows the type of the transport layer protocol.
- HEADER CHECKSUM: this checksum is for the header part only. It does not include the data field part.

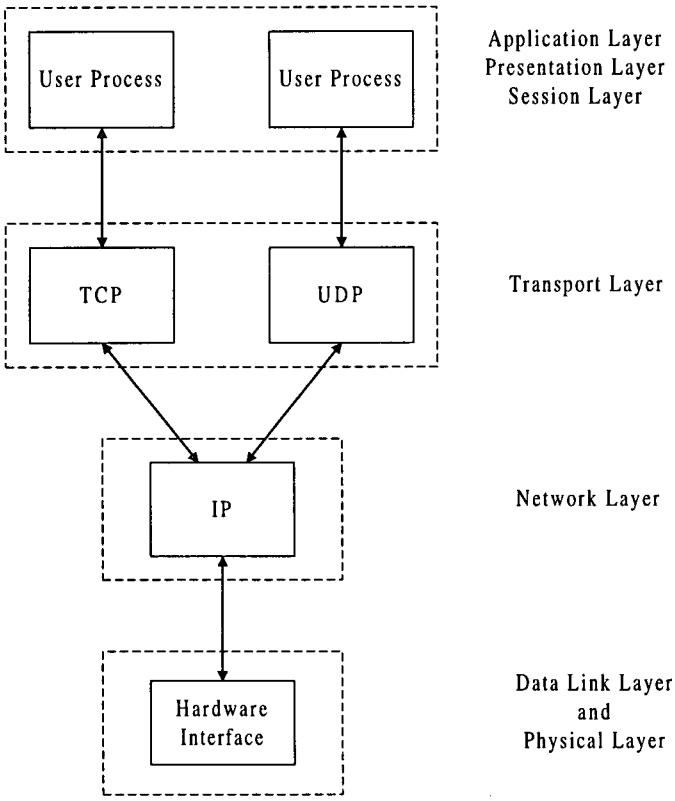
- SOURCE IP ADDRESS: the 32-bit IP address of the host sending this datagram.
- DESTINATION ADDRESS: the 32-bit IP address of the destination host.
- IP OPTIONS: the IP Options are vendor specific.
- PADDING: if an option field is used, the datagram is supposed to be padded with zeros up to the next 32-bit boundary.

The User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are the two most popular transport layer protocols that work with IP. They are usually referred to as the UDP/IP and the TCP/IP to indicate that they both use the IP protocol. The main difference between the TCP and UDP is that the TCP provides a connection-oriented, reliable, full duplex, byte stream service to an application program. The UDP, on the other hand, provides a connectionless, unreliable datagram service. Table 1 summarizes their features. Figure 15 shows the UDP, TCP and IP in OSI layers.

Table 1: Features of IP, UDP and TCP

	IP	UDP	TCP
Connection-Oriented?	No	No	Yes
Message Boundaries?	Yes	Yes	No
Data Checksum?	No	Yes	Yes
Positive Acknowledgement?	No	No	Yes
Time out and re-transmit?	No	No	Yes
Duplicate detection?	No	No	Yes
Sequencing?	No	No	Yes
Flow Control?	No	No	Yes

Figure 15 : OSI Layers Model showing UDP, TCP, and IP



3.1 User Datagram Protocol (UDP)

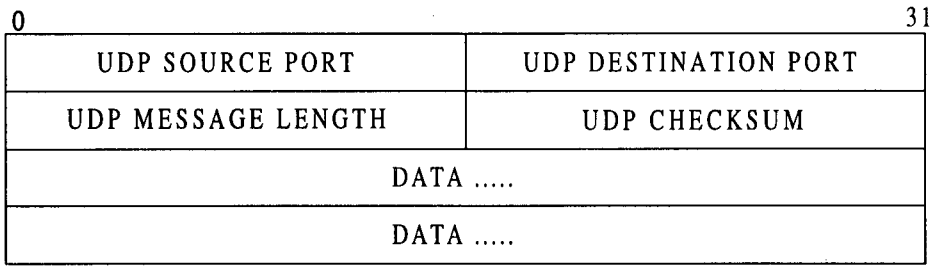
The UDP is not a connection-oriented protocol. Unlike the TCP, each UDP datagram is an independent datagram, which implies that each datagram can be delivered to a different computer host in a different application. Each UDP datagram contains both the source port number and the destination port number, which makes it possible to deliver the datagram to the correct receiver and for the receiver to send a reply. However, the UDP datagram does not provide the source or the destination IP address. The UDP relies upon the IP in the network layer to transport a message from one machine to another, because the IP header contains the source and the destination machine IP addresses.

The UDP provides unreliable datagram delivery service. It does not require datagram acknowledgement from the receiver. It also does not provide any flow control. Each UDP datagram can be lost, duplicated, or it can arrive out of order. Therefore, a UDP application has to provide a mechanism to handle all the above issues. On the other hand, the UDP is simple and flexible to use, and it is a popular protocol for wireless communication applications. This is because a UDP application will not use as many computer system resources as an application which uses the TCP. For example, if a TCP application is communicating with 1000 users, then the application will have 1000 connections, TCP sockets, opened to each different user all the time even when there are no activities between the users and the application. This is because the TCP is a connection-oriented protocol. However, for the UDP, a single UDP socket is enough for communicating to all the users. Moreover, the UDP also provides low overhead in terms of the size of the transport layer protocol header.

3.1.1 Format of UDP Datagram

Figure 16 shows the format of a UDP Datagram.

Figure 16: The Format of UDP Datagram



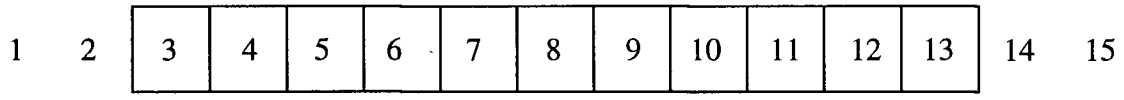
- UDP SOURCE PORT: this is the port ID of the sending process.
- UDP DESTINATION PORT: this is the port ID of the receiving process.
- UDP MESSAGE LENGTH: this is the total length of the UDP datagram including the header.
- UDP CHECKSUM: the checksum is optional. If the value of the checksum is zero, it means the checksum is not implemented. The checksum includes the IP addresses of the source and the destination. Since the UDP header does not include the source and the destination IP addresses, the UDP will prepend a pseudo-header to the UDP datagram when it calculates the checksum. The pseudo-header includes the source and the destination IP addresses. The reason that we include the source and destination IP addresses to the UDP datagram checksum is that the transport layer on the receiving side can use the checksum to verify that the datagram has reached the correct destination host and the port.

3.2 Transmission Control Protocol (TCP)

The TCP provides reliable delivery service to an application. The TCP is a connection-oriented transport layer protocol. It provides error recovery and flow control, and it also guarantees that all packages arrive in the correct order. The TCP uses a positive acknowledgement (ACK) approach for recovery of lost packages. When a sender sends a package to a receiver, the receiving transport layer will have to acknowledge the package. If the sender's transport layer has not received the acknowledgement within the timeout interval, the sender's transport layer will assume the package is lost, and it will re-transmit the package again automatically.

The TCP assigns a sequence number to every package that it sends out. The transport layer of the receiving side will use the sequence number to re-arrange the packages when they arrive out of order, and to eliminate duplicated packages. As indicated in Table 1, the TCP does not provide a message boundary. The sender's transport layer will buffer up the incoming data bytes before passing them to the IP network layer. Therefore, when an application passes data to the TCP, this does not mean the data will be sent out immediately. The TCP transport layer will queue up the incoming data until the queue reaches a certain size, then it will send out all the queued data. If an application wants the TCP to send the data out as soon as the TCP gets it, he has to issue a PUSH command to flush the TCP queue buffer. The TCP is full duplex. Unlike the UDP, in which each UDP datagram is independent and the connection will be closed after the UDP datagram is sent, the TCP keeps the connection open until an application explicitly closes the connection. The TCP uses a sliding window scheme for flow control as shown in Figure 17.

Figure 17: TCP Sliding Window



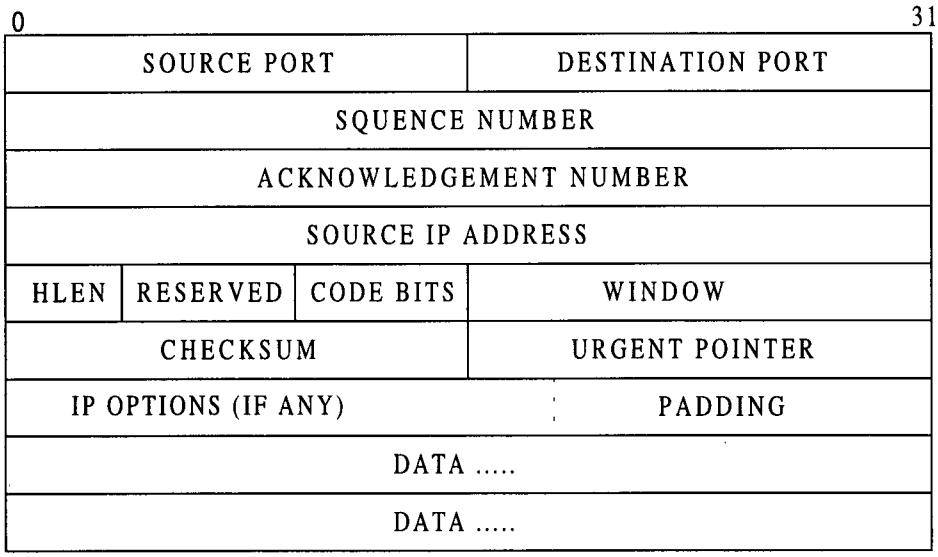
In this example, packages 1 and 2 have been sent and acknowledged. Packages 3 to 13 have been sent but not acknowledged. Packages 14 and 15 have not been sent. If Package 3 is acknowledged, the window will move to the right by one, and Package 14 will be sent. However, if package 3's acknowledgement is lost or never acknowledged, Package 14 will never be sent. When the acknowledgement timeout timer expires,

Package 3 will be re-transmitted, and Package 14 will not be sent until Package 3 is acknowledged by the receiver.

3.2.1 The TCP Segment Format

Figure 18 shows the TCP segment format.

Figure 18: TCP Segment Format



- SOURCE PORT: the sender's port ID, which will be used by the receiver to reply.
- DESTINATION PORT: the port ID of the receiving process.
- SQUENCE NUMBER: the sequence number of the first byte of this segment.
- ACKNOWLEDGEMENT NUMBER: this field contains the next sequence number that the receiver expects to receive, which means the packages with sequence numbers smaller than this number have been received successfully.
- HLEN: this field indicates the length of the TCP segment header.
- CODE BITS: the bit value of the CODE BITS field is defined as follows:

Bits (left to right):	Meaning if bit is set to 1:
URG	Urgent pointer field is valid
ACK	Acknowledgement field is valid
PSH	This segment requests a push
RST	Reset the connection
SYN	Synchronize sequence numbers
FIN	Sender has reached end of its byte stream

- WINDOW: this field indicates the number of bytes of data that the receiver can take.
- CHECKSUM: this checksum covers the TCP segment header, TCP data as well as the Pseudo header as described in the UDP Datagram header.
- URGENT POINTER: points to the first byte of the urgent data. This field is only valid when the URG bit in the CODE BITS is set.
- OPTIONS: this field specifies the TCP options that have been set for this TCP segment.

3.3 UDP and TCP Comparison

For data communication, TCP is a very attractive transport layer protocol, because it takes care of most of the error checking for the application. However, for JPEG image transmission over a wireless network, the UDP protocol may be more suitable. This is because the UDP has lower overhead in terms of the protocol header size, and it also provides selective re-transmission capability. As shown in Figure 12, the decoded image at stage 5 has almost 95% of the important image information. Therefore, if scans between stages 6 to 10 are corrupted, this does not affect the reconstruction image quality very much. If we send each scan by the UDP, any missing scans between stages 6 to 10 will not need to be re-transmitted. However, if we send each scan using the TCP, any missing scans will be automatically re-transmitted by the

transport layer of the sender, and the application does not have the control on which missing scan should be re-transmitted. If the wireless channel error rate is high, it will take a long time for an application to send an image by using the TCP, simply because every missing scan will be re-transmitted.

To summarize, when the UDP is used in conjunction with the JPEG progressive mode, an application only needs to send a few scans, saving transmission time. If the user wants to have a better image quality, the user can ask the image server to send more scans.

Chapter 4 Progressive Image Transmission Protocol (PITP)

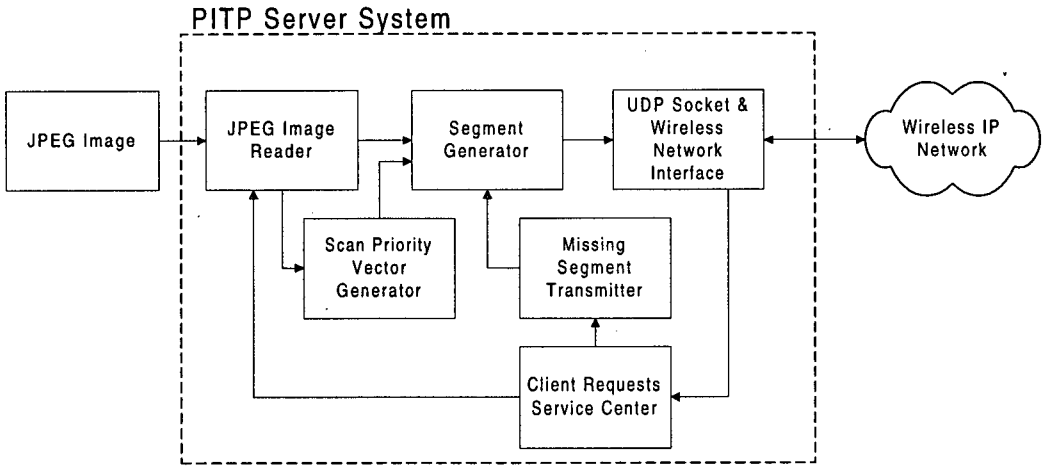
A Progressive Image Transmission Protocol (PITP) is proposed to transmit JPEG files over the CDPD wireless network using the UDP transport layer protocol and the JPEG progressive compression mode [17] [18]. The proposed PITP system consists of two parts. One part will be the image server called the Server PITP, and the other part will be a mobile computer called the Client PITP. The Client PITP will issue the image transmission request to the Server PITP. Then the Server PITP will transmit the JPEG image by sending each scan one by one over the CDPD network using the UDP. A scan priority is assigned to each scan, and each scan will be separated into a number of small segments. When a segment of a scan is missing, the Client PITP will use the Segment Re-transmission algorithm, discussed later, which uses the scan priority and the current channel conditions to determine if the missing segment or scan should be re-transmitted.

The Server PITP will time stamp each segment sent to the client. When the client receives a segment, he will compare the receiving time with the sending time found in the segment. The difference is the end-to-end delay for the server to send a segment to the client over the CDPD network. If the CDPD network is not busy and if the bit error rate of the network is low, the delay will be small. If the CDPD network is busy or the bit error rate of the network is high, the delay will be large. This is how the client program determines the CDPD network channel condition, which is one of the factors that the Segment Re-transmission algorithm will take into account to determine if a missing scan or segment should be re-transmitted.

4.1 Server PITP System

The main task of the Server PITP system is to send the requested JPEG progressively encoded image to the Client PITP system over the wireless IP network. All available images are pre-compressed and stored in the image server. When the Server PITP receives a transmission request from the Client PITP system, it will read the JPEG file and send each scan one by one over the wireless network by using the UDP/IP. Figure 19 shows the Server PITP System.

Figure 19: Server PITP System



The Server PITP system will first open an UDP socket [19] [20] and wait for an image transmission request from the PITP Client System. When it receives an image transmission request, it will read the requested JPEG progressively encoded image file into the memory. It will then separate the JPEG file into the frame header data and the scan data. As mentioned before, the reason that we want to send the frame header and the scan data separately is because the frame header data is very important to the JPEG image decoder. If the JPEG image decoder does not have the frame header data, it

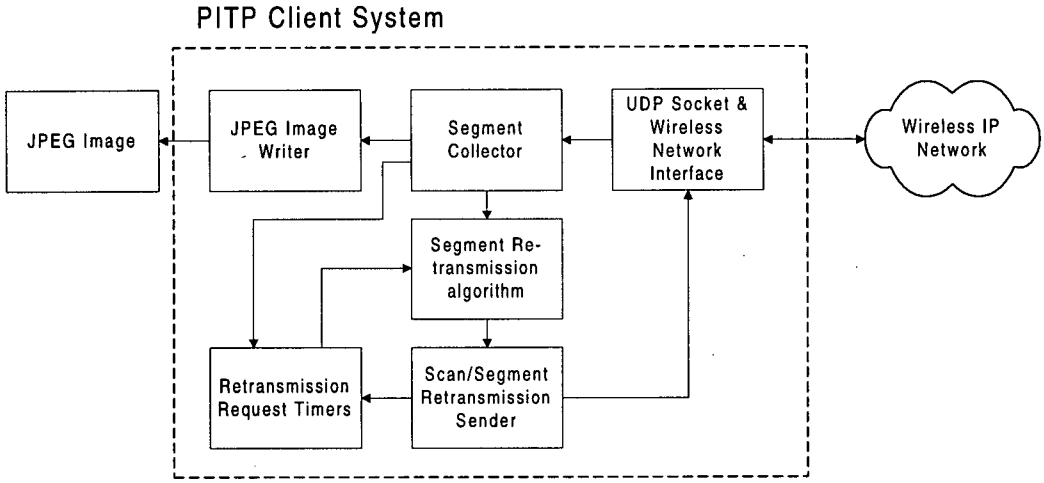
cannot decode the scans even the Client PITP receives all the scans. If the frame header data is lost in the transmission, the Client PITP will request the server to re-transmit the frame header data. Except for the JPEG frame header data, the Client PITP also needs the scan priority vector for the Segment Re-transmission algorithm.

As illustrated in Figure 13, a scan in a JPEG progressively encoded file contributes energy differently to the final decoded image. The first few scans will contribute more energy and they are more important. The last few scans will contribute less energy and they are less important. Each scan is assigned a priority. Therefore, the Segment Re-transmission algorithm in the Client PITP can use the scan priority and the channel condition to determine if a missing segment should be re-transmitted. Since the channel condition and the scan priority are different in nature, the Server PITP will need to normalize the channel condition and the scan priority. The priority value of each scan will be normalized from 1 to 10. A scan with a priority value of 10 is the most important scan, and a scan with a priority level of 1 is the least important scan. The normalization of the channel condition will be discussed in Section 4.5 .

4.2 Client PITP System

The Client PITP system is a mobile PC. A user can make an image transmission request through the Client PITP system. The Client PITP system will communicate with the Server PITP system through a wireless modem. Figure 20 shows the Client PITP system.

Figure 20: Client PITP System



The Client PITP will send the requested JPEG image file name to the Server PITP system. When the Client PITP receives scans from the Server PITP, it will verify the number of scans received so far as well as check if the previous scan has been completely received. If a segment of the previous scan is missing, the Segment Re-transmission algorithm will be used to determine if the missing segment should be re-transmitted. The Segment Re-transmission algorithm will be discussed in Section 4.5 . If a missing segment fails the Segment Re-transmission algorithm, the missing segment will be re-transmitted from the server. Otherwise, the missing segment will not be re-transmitted and the whole scan associated with the missing segment will be discarded.

When a re-transmission request is sent, a re-transmission timeout timer for the missing segment is started. If the missing segment is not received before the timer expires, it will be re-evaluated again using the Segment Re-transmission algorithm based on the latest channel condition. Therefore, a missing segment may be re-transmitted when it is evaluated by the Segment Re-transmission algorithm the first time, but it may

not be re-transmitted when it is re-evaluated the second time, as the channel condition may have changed substantially in-between the two evaluations.

4.3 Segment Generation

As mentioned in Section 3.1 , the UDP provides a message boundary, which means an application can send variable size UDP datagrams to the client system. If a scan is too long, the IP network layer will separate the scan into a number of smaller UDP datagram fragments. This process is called fragmentation. The fragmentation process should be transparent to the client transport layer. This is because the IP network layer in the client system will need to reassemble all fragmented segments into one scan and pass the reassembled scan to the client transport layer. However, if a fragmented segment is corrupted or lost in transmission and the data link layer protocol of the wireless network is not able to correct or detect the error, the client IP network layer will not be able to reassemble the fragmented scans. The client IP network layer will then discard all the fragmented UDP datagrams associated with the corrupted datagram and the scan will be declared missing. If the missing scan fails the Segment Re-transmission algorithm, the Client PITP will send a re-transmission request to the server. If the missing scan is long, it will take a long time for the server and the wireless network to deliver the scan again to the client.

In order to speed up transmission, the Server PITP will separate a scan into number of small segments, and each segment will be considered as an independent UDP datagram. Therefore, if a segment is lost or corrupted in the transmission, the Server PITP will only need to re-transmit the missing segment and not the whole scan to the Client PITP. When the Client PITP receives all the segments, it will reassemble them to

form a scan. As a result, the number of bytes of data sent over the wireless network and the network connection time will be reduced, because we no longer need to re-transmit the long scans, only the missing segments.

4.4 Re-transmission Timeout Timer

If a missing segment is not received by the Client PITP before the re-transmission timeout timer expires, a re-transmission request will be re-issued provided that the missing segment still fails the Segment Re-transmission algorithm under the current channel condition. The value of the timeout interval is therefore a very important parameter. If the Client PITP timeouts too soon, it may cause a lot of unnecessary re-transmissions and it may make the channel even busier. For example, if the forward channel error rate for the wireless network suddenly becomes very high, every segment sent by the base station will likely be re-transmitted many times between the base station and the mobile station. The base station then becomes congested, as all the outgoing segments that include one of the re-transmitted segments will be queued up in the base station. If the Client PITP time-outs too late, it will take the client a long time to discover that the re-transmitted segment is missing again.

One of the methods of estimating the timeout interval is by first choosing an arbitrary initial value [21]. If the re-transmission timer expires and the requested segment has still not been received, then the timeout interval will be increased. The problem associated with this method is that the timeout interval will then not be based on the current channel condition. Since there is no way to update the current channel condition in the process of estimating the timeout interval, the latter will always be

increased. Therefore, when the channel condition improves, the timeout interval may still remain very long.

Another more common way to estimate the timeout interval is to measure the end-to-end delay for a segment to travel from the server to the client. The most common algorithm used to realize the above method is called the Karn's algorithm. The Karn's algorithm is currently being used in some of the TCP/IP implementations. It uses the round trip delay for estimating the timeout interval. The TCP computes the round trip time by subtracting the segment sending time from the segment acknowledgement time. The round trip travel time will then be used to estimate the timeout interval using

$$\text{timeout interval} = \gamma * \text{round trip travel time.} \qquad (1)$$

where the value of γ is typically set to 2. The above estimation becomes more complicated when re-transmissions are involved. For example, consider the case where a segment is sent, but the sender does not receive an acknowledgement before the timer expires. In this case, either the corresponding segment or the segment acknowledgement has been lost during the transmission. The server will then send the segment again. When the server receives the acknowledgement for the segment, it is impossible for the server to find out if the acknowledgement corresponds to the original segment or the re-transmitted segment, as both segments carry the same data. If the acknowledgement corresponds to the original data segment, the round trip time will be long. However, if the acknowledgment corresponds to the re-transmitted segment, the round trip time will be short. Of course, the easiest way to solve this problem is that the server transport layer ignores the round trip time from the segment that involves re-transmission. But the estimated round trip time will not fully reflect the current channel condition anymore,

because some of the round trip time samples are skipped. In Karn's algorithm, the timeout interval is computed by using the formula given in Equation (1). And the algorithm will ignore the round trip time samples that involve re-transmission. If the re-transmitted segment is lost and needs to be re-transmitted, the current timeout interval will be doubled. The TCP will keep an upper limit on the length of the timeout interval to make sure that the timeout interval will not become impractically large. When the sender receives an acknowledgement corresponding to a segment that does not require re-transmission, the new timeout interval will be re-estimated based on the new round trip travel time.

The Client PITP uses a different method to estimate the timeout interval, as the client does not send segment acknowledgments to the server. In our PITP system, the client will use the most current end-to-end delay to estimate the timeout interval. Each segment is time stamped when it is sent. The client can subtract the sending time from the receiving time to get the current end-to-end delay. The timeout interval is estimated based on the following formula

$$\text{timeout interval} = \gamma * \text{end-to-end travel time.} \qquad (2)$$

The TCP requires the γ to be set to 2 and uses the round trip travel time to estimate the timeout interval. For the PITP, the default value of γ is 4, because the PITP uses the end-to-end delay to estimate the timeout interval. Therefore, the re-transmission timeout timer is four times the end-to-end travel time.

4.4.1 Simulation Results

An experiment was performed to compare the performance of our Client PITP timeout algorithm and the Karn's algorithm. The simulation assumes the end-to-end delay is Gaussian distributed. The PITP algorithm always uses the previous end-to-end delay to estimate the current end-to-end delay. Therefore, the estimated end-to-end delay by the PITP algorithm is always one sample behind the real end-to-end delay curve. The Karn's algorithm will also use the previous end-to-end delay to estimate the current end-to-end delay if the previous delay is not from a segment that required re-transmission, and it will skip the samples that are re-transmitted. Therefore, the accuracy of the Karn's algorithm to estimate the timeout interval depends on the number of re-transmissions. If there are no re-transmissions, the performance levels of the Client PITP algorithm and the Karn's algorithm are identical. The more re-transmissions are needed, the lower the performance of the Karn's algorithm will be. This is because the Karn's algorithm skips more samples.

Figure 21 shows the estimation errors of the Karn's algorithm and the Client PITP algorithm. The estimation errors were calculated by subtracting the estimated delay from the actual delay. A new 10000 random samples were used for each re-transmission rate, and then the total accumulated absolute estimation errors were obtained. Figure 22 shows the difference, as a function of the re-transmission rate, between the Karn's algorithm estimation error and the PITP algorithm estimation error.

Figure 21: PITP and Karn's Algorithm Estimation Error

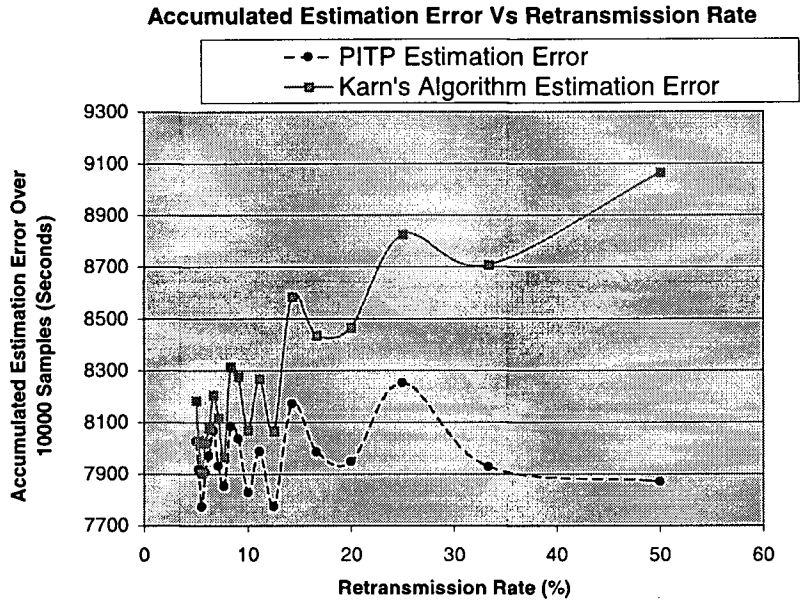
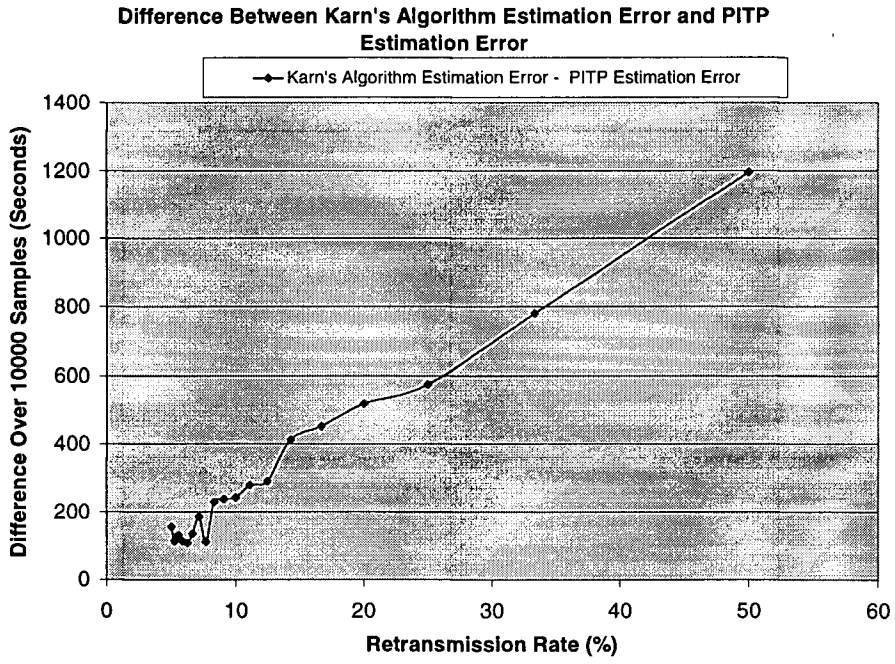


Figure 22: Difference between Karn's Algorithm Estimation Error and PITP Algorithm Estimation Error



As shown in Figure 21 and Figure 22, the accumulated estimation errors of the Karn's algorithm are always higher than those of the Client PITP algorithm. As the re-

transmission rate increases, the accumulated estimation error for the Karn's algorithm increases. The reason is that the Karn's algorithm skips the round trip time samples that involve re-transmissions and uses the old delay for estimating the new timeout. Therefore, the more re-transmissions occur, the more samples that the Karn's algorithm will skip, and the less accuracy of the current end-to-end delay estimation.

4.5 Segment Re-transmission Algorithm

When a segment is lost in the transmission, the client will evaluate the missing segment based on certain criterion to determine if the missing segment should be re-transmitted or not. Since we are sending JPEG progressively encoded images and not data files, the client does not need to receive all the scans. The received image is still useful even if part of image file is lost and not re-transmitted. When we transmit data over a wireless network, we always care about the channel condition or the channel bit error rate. This is because if the channel bit error rate or the channel load is low, we can send data to the destination very fast with a fewer transmission errors. If the channel bit error rate or the channel load is high, we will experience longer transmission time and more transmission errors. Therefore, when the channel condition is bad, we should send as a few scans or segments as possible over the channel.

Each scan of a JPEG progressively encoded image is prioritized. The priority of the scan will tell us how important the scan is and how distorted the decoded image will be if the scan is lost. The more important the scan is, the more information it carries and the less distortion the received image will have. Therefore, if a high priority scan is lost, we will want the server to re-transmit the scan.

Clearly, both the channel condition and the scan priority will have to be considered simultaneously when a missing segment is evaluated to determine if it should be re-transmitted. This is because if the channel condition is very bad (i.e., it takes minutes or even hours to deliver a missing segment), then we may not want to wait for a long time to obtain the missing segment even if it is very important. On the other hand, if the channel condition is so good that it only takes the network a second to re-transmit the missing segment, we may want the server to re-transmit the missing segment to minimize the decoded image distortion.

In order to compare the scan priority and the channel condition factors, which have different effects, such factors need to be normalized. Since the channel delay will have a negative impact on transmission, the longer the normalized channel delay, the lesser the importance that a missing segment be re-transmitted. The channel condition will be normalized in such a way that the channel condition will be compared to the user acceptable channel delay. For example, we may consider that a channel is in good condition if the channel can deliver a segment within 5 seconds. Therefore, if the channel delay is 5 seconds, we want to receive all the scans. If there are missing scans, we want to re-transmit the missing scans because of the good channel condition. The user acceptable channel delay should not be fixed because wireless networks may be implemented differently. For example, the CDPD network in Vancouver, British Columbia may have a relatively short channel delay, because the service provider BC TEL has allocated two cellular channels that are dedicated for CDPD data transmission. For other CDPD networks, the CDPD users may have to share the channels with voice

transmission, so they may experience longer channel delays. In this work, the normalized channel delay will be given by

normalized channel delay =(current channel delay/user acceptable channel delay) – 1. (3)

If the acceptable channel delay is 5 seconds and the current channel delay is 50 seconds, the normalized channel delay will be 9. If the current channel delay is more than 50 seconds, the normalized channel delay will be 10. In other words, the maximum limit on the normalized channel delay will be 10. If it takes the channel 50 seconds to delivery a 1 kbyte segment, we will need to wait 23 minutes to receive 28110 bytes of a JPEG progressively encoded image. Therefore, if the channel condition is bad, most or all missing segments should not be re-transmitted.

The normalized scan priority levels are derived from the JPEG files. In this thesis, two cases are here simulated. In the first case, all scans have the same priority level. For the uniformly prioritized scans, the PITP assumes the first scan is the most important scan with a normalized scan priority level of 10, and the last scan is the least important scan with a normalized scan priority level of 1. In the second case, the scan priority level is proportional to the scan importance factor over the scan size. This latter scheme will be discussed in more detail in the next section.

By combining the normalized scan priority and the normalized channel delay, the Segment Re-transmission algorithm is to evaluate

Score = normalized scan priority - normalized channel delay. (4)

Since the normalized channel delay has the negative impact on the transmission, the normalized channel delay is subtracted from the normalized scan priority. The two

factors cannot be added or multiplied together. Otherwise, the higher the normalized channel delay, the more scores a missing segment will get.

If the Segment Re-transmission algorithm produces a score of 1, or higher, the missing segment will be re-transmitted. In the Client PITP system, the Segment Re-transmission algorithm is slightly different from the equation shown above. The Segment Re-transmission algorithm is to evaluate

$$\text{Score} = \text{scan\_priority\_weight} * \text{normalized scan priority} - \text{channel\_delay\_weight} * \text{normalized channel delay.} \quad (5)$$

We have introduced the scan_priority_weight and the channel_delay_weight into Equation (4). If the scan_priority_weight and the channel_delay_weight are set to 1, this Segment Re-transmission algorithm is the same as that given by Equation (4). A user may sometimes want to impose a certain level of received image quality. The scan_priority_weight and the channel_delay_weight affects whether a missing segment will be re-transmitted. For some wireless networks, the connection is charged by the connection time, so a short transmission time may be more favourable to the users.

Table 2 summarizes the relationship between the magnitude of the end-to-end delay, in terms of the number of times of the user acceptable channel delay, and the number of re-transmitted scans for the uniformly prioritized Lena image.

Table 2: The number of re-transmitted scans for Lena image as a function of the end-to-end delay

Scan	Normalized Scan Priority	Within 1 time	Between 1 time and 2 times	Between 2 times and 3 times	Between 3 times and 4 times	Between 4 times and 5 times	Between 5 times and 6 times
1	10	Yes	Yes	Yes	Yes	Yes	Yes
2	9	Yes	Yes	Yes	Yes	Yes	Yes
3	8	Yes	Yes	Yes	Yes	Yes	Yes
4	7	Yes	Yes	Yes	Yes	Yes	Yes
5	6	Yes	Yes	Yes	Yes	Yes	Yes
6	5	Yes	Yes	Yes	Yes	Yes	No

7	4	Yes	Yes	Yes	Yes	No	No
8	3	Yes	Yes	Yes	No	No	No
9	2	Yes	Yes	No	No	No	No
10	1	Yes	No	No	No	No	No

As shown in Table 2, when the end-to-end delay increases, the number of skipped scans increases and the received image quality starts deteriorating. Then, we can use the scan_priority_weight to compensate the unexpected variation of the end-to-end delay so that a higher quality image will be received. For example, if the end-to-end delay suddenly increases to 3 times of the user acceptable channel delay, then all the missing segments (e.g. scans 8 to 10) with a normalized scan priority less than 4 will not be re-transmitted. If the scan_priority_weight is set to 2, all normalized scan priority level will be multiplied by 2. The result is shown in Table 3.

Table 3: The effect of the end-to-end delay for Lena image with a scan_priority_weight parameter set to 2

Scan	Normalized Scan Priority	Within 1 time	Between 1 time and 2 times	Between 2 times and 3 times	Between 3 times and 4 times	Between 4 times and 5 times	Between 5 times and 6 times
1	10	Yes	Yes	Yes	Yes	Yes	Yes
2	9	Yes	Yes	Yes	Yes	Yes	Yes
3	8	Yes	Yes	Yes	Yes	Yes	Yes
4	7	Yes	Yes	Yes	Yes	Yes	Yes
5	6	Yes	Yes	Yes	Yes	Yes	Yes
6	5	Yes	Yes	Yes	Yes	Yes	Yes
7	4	Yes	Yes	Yes	Yes	Yes	Yes
8	3	Yes	Yes	Yes	Yes	Yes	Yes
9	2	Yes	Yes	Yes	Yes	No	No
10	1	Yes	Yes	No	No	No	No

When the scan_priority_weight parameter is set to 2, the Segment Re-transmission algorithm can tolerate a higher unexpected variation of the end-to-end delay from within one time of the user acceptable channel delay to two times of the user acceptable channel delay. When the scan_priority_weight increases to a value of 11 or higher, all the missing segments will be re-transmitted. To evaluate the performance of

the Segment Re-transmission algorithm, we have conducted an experiment where a uniformly prioritized and compressed Lena image is transmitted. In this experiment, the channel end-to-end delay is set to 4 times higher than the user acceptable channel delay. Therefore, all missing segments with normalized scan priority level less than 5 would fail the Segment Re-transmission algorithm. We also set the forward channel segment loss rate to 20% so that we can simulate the impact of missing a large number of segments in the forward channel. The reverse channel segment loss rate was set to 0%, as we only want to simulate the loss of segments in the forward channel. The experimental results are shown in Table 4.

Table 4: Received image quality as a function of scan_priority_weight parameter

Scan_priority_weight	Total transmission time (seconds)	Received image PSNR (dB)
1	44	31.71
2	48	40.03
3	53	None of the scans are skipped

As shown in Table 4, the received image PSNR increases as the scan_priority_weight increases. However, at the same time, the total transmission time also increases. This is because more missing segments failed the Segment Re-transmission algorithm and therefore need to be re-transmitted.

The effect of the channel_delay_weight parameter is just the opposite of that of the scan_priority_weight parameter. If we want to have a shorter transmission time with fewer re-transmissions under a highly varying channel delay, and if we are willing to accept a lower quality image, we can set the channel_delay_weight to some values larger than 1. As the channel_delay_weight increases, fewer re-transmissions will occur, resulting in a shorter transmission time and a lower image reproduction quality.

We have conducted an experiment to study the impact of the channel_delay_weight parameter on the total transmission time for a highly varying channel delay. A uniformly prioritized Lena image is used in this experiment. We first simulated a channel delay to match the user acceptable channel delay which was set to 4 seconds, and we set the forward channel segment loss rate to 20%. The reverse channel segment loss rate was set to 0%. Since the simulated channel delay was set within the user acceptable channel delay, all missing segments were re-transmitted. Then, we increased the simulated channel delay to 6 seconds. As a result, some of the missing segments were not re-transmitted. The experimental results are shown in Table 5.

Table 5: Total transmission time as a function of channel_delay_weight parameter

Simulated Channel Delay (seconds)	User Acceptable Channel Delay (seconds)	Channel_delay_weight	PSNR of the received image	Total Transmission Time (seconds)
4	4	1	All scans received	152
6	4	1	All scans received	188
6	4	2	43.50dB	146
6	4	3	35.13dB	153

As shown in Table 5, when the simulated channel delay was set within the user acceptable channel delay, all missing segments were re-transmitted and the total transmission time was 152 seconds. When the simulated channel delay was increased to 6 seconds, the total transmission time increased to 188 seconds, but all the scans were still received. This is because all missing segments have normalized scan priority level that are larger than 1. When the channel_delay_weight was set to 2, the total transmission time decreased to 146 seconds, as some missing segments were not re-transmitted. When the channel_delay_weight was set to 3, more missing segments were not re-transmitted, and the PSNR decreased to 35.13 dB. Therefore, as the channel_delay_weight increased, the transmission time was still at around 152 seconds,

even though the simulated channel delay increased from 4 seconds to 6 seconds. During a connection time, a user will want to have a stable transmission time when the channel bit error rate is increased. By setting the channel_delay_weight parameter to some values larger than 1, a user can have a shorter transmission time and a fewer re-transmissions, of course, at the expense of reduction in reproduction quality.

4.6 Prioritization by Scan Importance Factor and Scan Size

Scans can be prioritized by the ratio of the scan importance over the scan size [22]. In this method, the reconstruction distortion, or the squared error, is calculated for each bit of the DCT coefficients of all the 8x8 blocks of an JPEG image. The overall distortion is then calculated by summing the squared error of the bit at the same position of the DCT coefficients for all the 8x8 blocks. Following this method, the decoder assume that the DCT coefficients of all 8x8 blocks are zero-valued. The 0s are replaced by the actual bit values as the bits are received. For example, if the actual value of the seventh bit of an AC1 coefficient of an 8x8 block is 0, and if the actual value of the bit is missing in the transmission, the squared error is 0. This is because the decoder assumes the bit value to be 0 anyway. Therefore, the missing bit with value of 0 will not effect the reconstruction distortion. However, if the actual value of the bit is 1, and if the bit is missing in the transmission, the square error is equal to $(2^6)^2$. In general, the overall distortion is equal to

$$D_i = \sum_{j=1}^{N} (b_{i,j} 2^{pi-1})^2, \qquad (6)$$

Where i is the bit position in an 8x8 block, i.e. 1, 2, 3, ... 640, N is the number of 8x8 blocks, $b_{i,j}$ is the value of the i^{th} bit of the j^{th} 8x8 block and Pi is the position of the i^{th} bit.

After the overall distortion values and the bit rate are computed, and the bits will be grouped into scans by using the combination of the JPEG successive approximation and spectral selection methods. The scan importance factor is calculated by summing the overall distortion of the bits in a scan. The advantage of the method is that the more important DCT coefficients are always sent first. In this way, the client always receives the best image quality for the number of bytes it has received. Normally, when the receiver decodes 50% of all the scans, the decoded image contains more than 90% of the important image information. Therefore, if the last 50% of the scans are missing, it is mostly not necessary to re-transmit them when the channel bit error rate or the channel load is high. Figure 23 and Figure 24 show two compressed images using this method. Table 6 and Table 7 show their scan importance factors and their scan sizes.

Figure 23: Lena image, File Size: 152511 Bytes

Table 6: Scan importance, scan size and normalized scan priority for Lena

Scan	Scan Importance	Scan Size (bytes)	Normalized Scan Priority
1	223695	1491	10
2	24016	936	7

4	5236	1426	3
5	3874	536	5
6	3867	525	5
7	1566	3418	1
8	966	497	2
9	940	524	2
10	426	6590	1
11	251	785	1
12	246	524	1
13	110	13485	1
14	70	1549	1
15	62	523	1
16	28	32677	1
17	16	522	1
18	9	43980	1
19	4	525	1
20	3	41369	1

Figure 24: Mandril image, File Size 210864 Bytes

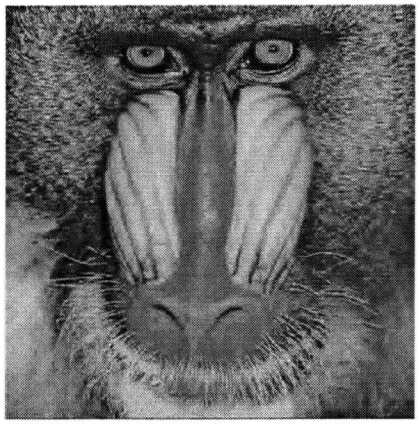


Table 7 : Scan importance, scan size and normalized scan priority for Mandril

Scan	Scan Importance	Scan Size (bytes)	Normalized Scan Priority
1	115945	1356	10
2	31814	103	10
3	16657	540	8
4	15048	797	7
5	14336	522	8
6	5897	4045	2
7	4842	2889	2
8	3941	522	5
9	1696	13746	1
10	1302	2747	1
11	993	525	2
12	487	27135	1
13	248	524	1
14	141	37556	1
15	62	524	1
16	42	41006	1

17	15	525	1
18	12	38784	1
19	4	524	1
20	3	36391	1

The normalized scan priority is expressed as

10 * Log (scan importance/scan size) / scan 1 normalized scan priority. (7)

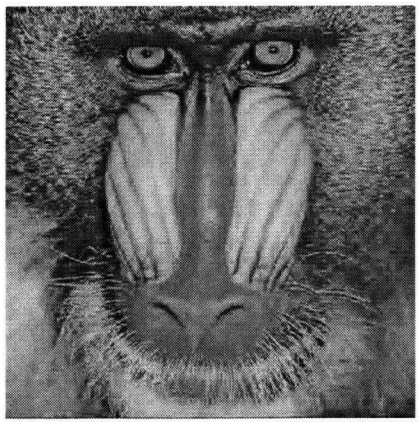
By dividing the scan importance factor by the scan size, we can obtain the average information a byte carries within a scan. The reason that we take the logarithm of the scan importance factor over the scan size is because the dynamic range of the scan importance factor is too large. The result of the logarithm will be multiplied by 10 and divided by the normalized scan priority of scan 1 so that scan 1 is always the most important scan. As shown in Table 6 and Table 7, the scan importance factor decreases as the scan number increases. For the last 50% of the scans, the normalized scan priorities are all 1. Two experiments have been conducted using the images shown in Figure 23 and Figure 24. We have setup the experiment such that if the lowest priority scans are missing, they will not be re-transmitted. The experimental results are shown in Figure 25 and Figure 26.

Figure 25: Received Lena image

Received Image Size: 21002 Bytes
PSNR: 35.27 dB

Original: Image Size: 152511 Bytes
Total Transmission Time: 170 seconds

Figure 26: Received Mandril image



Received Image Size: 16248 Bytes
PSNR: 23.38 dB

Original: Image Size: 210864 Bytes
Total Transmission Time: 225 seconds

As shown in Figure 25 and Figure 26, even if the received image represents only about 10% of the size of the original image, it contains almost all the important details of the original image. Clearly, if the channel error rate is high, this method will save a substantial amount of transmission time.

4.6.1 Optimal Segment Re-transmission Algorithm Coefficients for the images prioritized by the Scan Importance Factor and the Scan Size

The Segment Re-transmission algorithm controls which missing segments should be re-transmitted. By setting different values for the scan_priority_weight, channel_delay_weight and the user acceptable channel delay parameters, the Segment Re-transmission algorithm can be modified to obtain different levels of received image quality. For a JPEG coded image, the reproduction image quality for different bit rate ranges is known to be

- 0.25 ~ 0.5 bpp: moderate to good quality
- 0.50 ~ 0.75 bpp: good to very good quality
- 0.75 ~ 1.5 bpp: excellent quality
- 1.5 ~ 2.0 bpp: indistinguishable image (visually lossless)

For the images shown in Figure 23 and Figure 24, the scan sizes are shown in Table 8 and Table 9.

Table 8: Scan size, accumulated file size and normalized scan priority for Lena

Scan	Scan Size (bytes)	Accumulated File Size (bytes)	Normalized Scan Priority
1	1491	1491	10
2	936	2427	7
3	526	2953	7
4	1426	4379	3
5	536	4915	5
6	525	5440	5
7	3418	8858	1
8	497	9355	2
9	524	9879	2
10	6590	16469	1
11	785	17254	1
12	524	17778	1
13	13485	31263	1
14	1549	32812	1
15	523	33335	1
16	32677	66012	1
17	522	66534	1
18	43980	110514	1
19	525	111039	1
20	41369	152408	1

Table 9: Scan size, accumulated file size and normalized scan priority for Mandril

Scan	Scan Size (bytes)	Accumulated File Size (bytes)	Normalized Scan Priority
1	1356	1356	10
2	103	1459	10
3	540	1999	8
4	797	2796	7
5	522	3318	8
6	4045	7363	2
7	2889	10252	2
8	522	10774	5
9	13746	24520	1
10	2747	27267	1
11	525	27792	2
12	27135	54927	1
13	524	55451	1
14	37556	93007	1
15	524	93531	1
16	41006	134537	1
17	525	135062	1
18	38784	173846	1
19	524	174370	1
20	36391	210761	1

Both images have 512 pixels x 512 pixels. In order to obtain an image of moderate to good quality, the compressed image size should be within the following bounds:

Lower bound: 512 pixel x 512 pixel x 0.25 bit per pixel = 65536 bits or 8192 bytes

Upper bound: 512 pixel x 512 pixel x 0.5 bit per pixel = 131072 bits or 16384 bytes

Therefore, the decoder will need the first 10 scans of the Lena image or the first 8 scans of the Mandrill if a good quality level is to be achieved. This implies that all the missing scans with the lowest priority do not need to be re-transmitted, and we can obtain a decoded image of a moderate to good quality. Figure 27 shows the decoded images by just excluding the lowest priority scans.

Figure 27: Images with/without the lowest priority scans

Original Lena image:
File Size = 152408 Bytes

Image without the lowest priority scans:
File Size = 16574 byte, PSNR = 34.66 dB

Original Mandril,
File size = 210864 bytes:

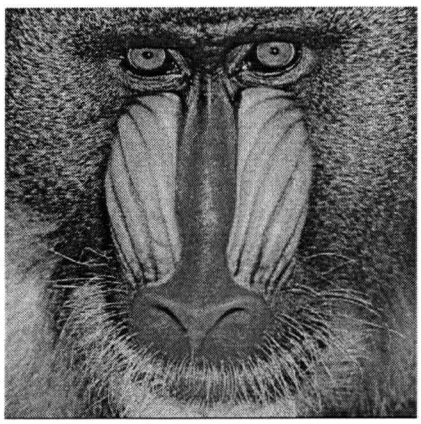
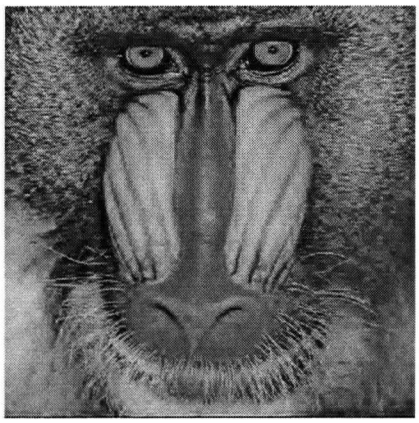


Image without the lowest priority scans:
File size = 10879 bytes, PSNR = 23.02 dB:



As shown in Figure 27, the decoded images without the lowest priority scans contain most of the important information of the original images. Therefore, the user acceptable channel delay should be set to be half of the average end-to-end delay so that all the missing segments with the lowest priority will not be re-transmitted, and we can still obtain a moderate to good quality image.

4.7 Server PITP and Client PITP Time Synchronization

Time synchronization between the Server and the Client PITP Systems is very important. In the PITP, one of the factors that we take into account when applying the Segment Re-transmission algorithm is the channel condition. The PITP subtracts the segment sending time from the segment receiving time to obtain the channel end-to-end delay. If the server and the client are not synchronized in time, any difference in time will affect the result of the Segment Re-transmission algorithm. For example, if the client clock is 10 seconds faster than the server clock, then the end-to-end delay is always 10 seconds longer than the actual end-to-end delay. The error may cause a would-be failing missing segment to pass the Segment Re-transmission algorithm, leading to it not being re-transmitted. We have conducted an experiment to show the impact of the synchronization in time between the server and the client on the received image quality. In the experiment, we have set the user acceptable channel delay to be 2 seconds, then we set the client clock to be 10 seconds faster than the server clock. The received image is compared to the image in which the server and the client are synchronized in time. Figure 28 shows the experimental results.

Figure 28: Received images with/without time synchronization

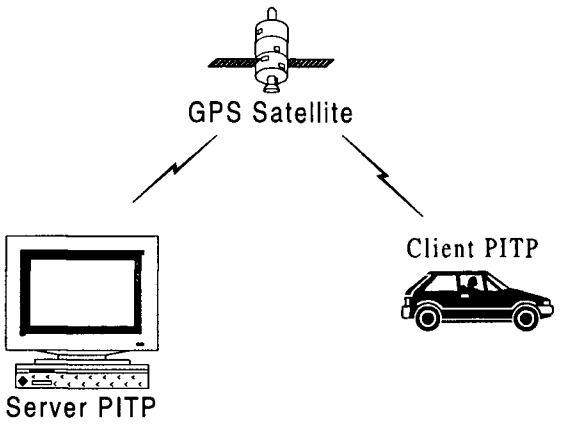
With time synchronization

Without time synchronization

As shown in Figure 28, if the server and the client are not synchronized in time, the client will not receive good quality image. Therefore, the server and client clocks must be synchronized. In this thesis, two methods are proposed:

1. Manually synchronize the clocks: manually synchronizing the clock of each mobile station to the clock of the server is the least expensive method. However, none of the clock timing is constant, and drift is inevitable. Moreover, it is also very easy for someone or a program to accidentally change the times of the stations. Therefore, if manual synchronization is used, every station needs to be re-synchronized periodically.
2. Use a common timing source: the second method is to use a common timing source for the server and the mobile workstation. A Global Positioning System (GPS) trans-receiver can be used to synchronize the timing between the server and the mobile station.

Figure 29: GPS Time Synchronization Scheme



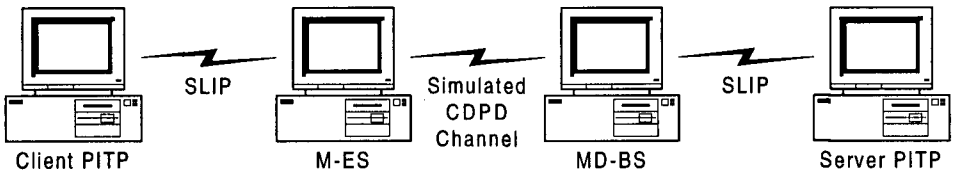
The GPS is a satellite launched by the US government to provide location information in latitude and longitude for any objects that are equipped with a GPS trans-

receiver. Except for the location information, the GPS also provides the current time information. The GPS trans-receiver can interface to a serial COM port of a computer. Therefore, the Server PITP and the Client PITP systems can use their serial COM ports to interface to a GPS trans-receiver and synchronize their timers to GPS satellite. If the timers of the server and mobile station are changed, they will not affect the result of the Segment Re-transmission algorithm. This is because the PITP stations are referencing their timing information from the same source, that is the GPS satellite.

Chapter 5 CDPD Emulator Test Bed for PITP Simulation

A CDPD emulator test bed was setup to test the efficiency of the PITP using the CDPD network. The CDPD emulator test bed consisted of four PCs, two of the PCs held the CDPD emulators: one held the M-ES emulator and the other one held the MD-BS emulator. The other two PCs operated the Server and Client PITP systems. The test bed setup is shown in Figure 30.

Figure 30: CDPD Emulator Test Bed Setup



The PITP PCs and the CDPD emulators were connected through the Serial Line Internet Protocol (SLIP). The simulated CDPD channel was a full duplex channel, with the forward channel for the MD-BS to communicate with the M-ES, and the reverse channel for the M-ES to communicate with the MD-BS. In the experiments, the uniformly prioritized Lena image is used. The objective of this experiment is to illustrate the tradeoffs achieved by the PITP between the transmission time and received image quality. First, we re-transmitted all missing scans. Then, we measured the time it took the network to re-transmit the missing scans as well as the improvement, in terms of PSNR, of the received image quality. Table 10 shows the time it takes the CDPD emulator to re-transmit a missing scan. Table 11 shows the PSNR of the received image, and Table 12 shows the tradeoffs between time savings and PSNR of the received image.

Table 10: Uniformly prioritized Lena transmission time using the CDPD Emulator Test bed

Re-transmitted scan	1	2	3	4	5	6	7	8	9	10
Re-transmission time (seconds)	2	3	3	1	1	4	6	4	1	3

Table 11: PSNR of the received image

Image contain scans	1	1 - 2	1 - 3	1 - 4	1 - 5	1 - 6	1 - 7	1 - 8	1 - 9
PSNR (dB)	25.8	27.7	28.6	28.8	29.9	31.4	34.5	38.4	43.5

Table 12: PSNR of the received image and the total transmission time tradeoffs

	25.8 dB	27.7 dB	28.6 dB	28.8 dB	29.9 dB	31.4 dB	34.5 dB	38.4 dB	43.5 dB
25.8 dB	-	-	-	-	-	-	-	-	-
27.7 dB	3s	-	-	-	-	-	-	-	-
28.6 dB	6s	3s	-	-	-	-	-	-	-
28.8 dB	7s	4s	1s	-	-	-	-	-	-
29.9 dB	8s	5s	2s	1s	-	-	-	-	-
31.4 dB	12s	9s	6s	5s	4s	-	-	-	-
34.5 dB	18s	15s	12s	11s	10s	6s	-	-	-
38.4 dB	22s	19s	16s	15s	14s	10s	4s	-	-
43.5 dB	23s	20s	17s	16s	15s	11s	4s	1s	-
All scans	26s	23s	20s	19s	18s	14s	7s	4s	3s

The experimental results in Table 10, Table 11 and Table 12 show that the PITP provides the user good tradeoffs between the received image quality and the total transmission time. Consider the example of a user who receives an image with PSNR of 29.9 dB. As shown in Table 12, he will save 4 seconds and 18 seconds if he does not request re-transmission of scan 6 and scans 6 – 10, respectively.

We have conducted an experiment on CDPD simulator test bed to test the tradeoffs of the transmission time and the received image quality for different forward channel segment loss rates. We have used the uniformly prioritized Lena image and also the Lena image which has been prioritized by the scan importance factor over the scan size [22]. The experimental result for the uniformly prioritized Lena is shown in Figure 31.

Figure 31: Transmission time and received image quality tradeoffs at different forward channel loss rates

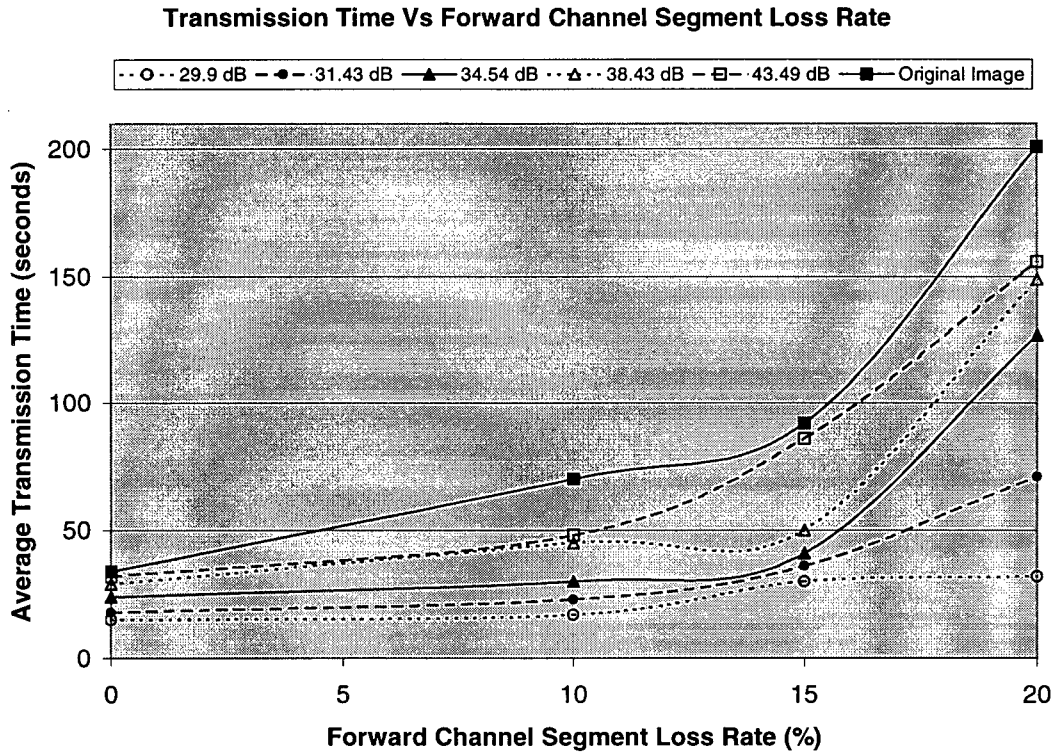


Figure 31 shows the tradeoffs between the transmission time with the received image quality, in terms of PSNR, at different forward channel segment loss rates. For example, if we satisfy the received image quality with PSNR of 34.54 dB at a forward channel segment loss rate of 20%, there is a total saving of 80 seconds in the transmission time compared to the scenario where we re-transmit all the missing scans. The received image with a PSNR of 34.54 dB contains most of the important information of the original image. A comparison of the original image and the received image with a PSNR of 34.54 dB is shown in Figure 32.

Figure 32: Uniformly prioritized Lena image and the Lena image with PSNR of 34.54 dB

Original image

Received image with PSNR of 34.54 dB

Visually, the images in Figure 32 are almost identical. Moreover, as shown in Figure 31, the higher the forward channel loss rate, the more savings that the PITP provides. The same experiment has also been conducted on the Lena image which has been prioritized by the ratio of scan importance over the scan size. The experimental results are shown in Table 13 and Figure 33.

Table 13: Transmission time at forward channel segment loss rate of 10%

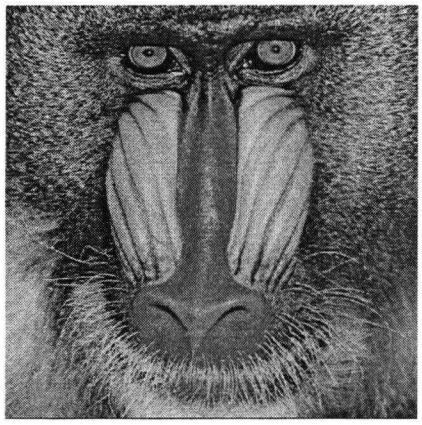
Original Lena	255 seconds
Lena image with PSNR of 35.27 dB	170 seconds
Original Mandril	413 seconds
Mandril image with PSNR of 23.38 dB	225 seconds

Figure 33: Prioritized compressed images, received with some missing scans

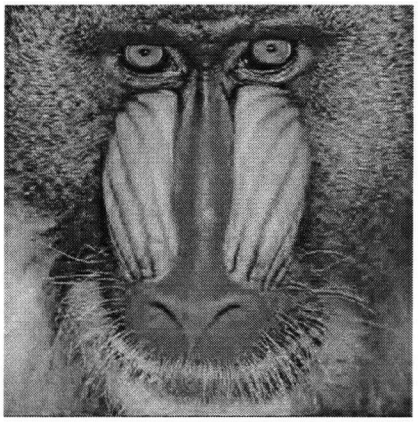
Original Lena image

Received image with a PSNR of 35.27 dB

Original Mandril



Received image with a PSNR of 23.38 dB

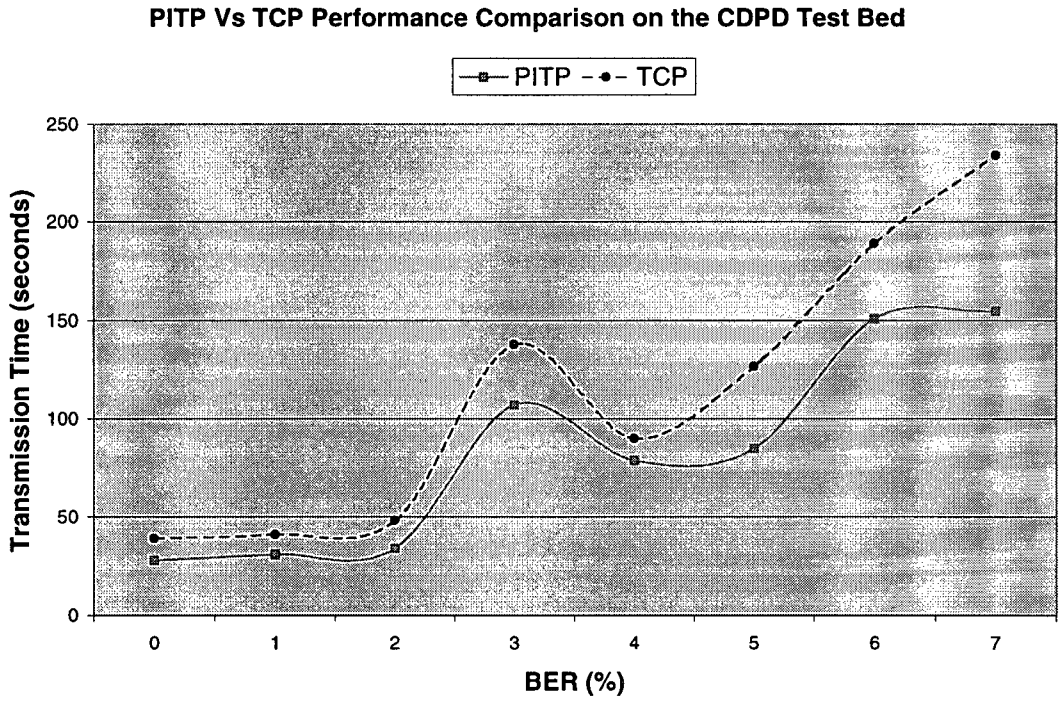


As shown in Figure 33, the received images with the missing scans contain most of the important information of the original images. If we are satisfied with the received image quality, we can have a saving of 85 seconds and 188 seconds, respectively, in transmission times for the Lena and Mandril images at the forward channel segment loss rate of 10%. For many networks, the charge for a connection is based on the connection time. Therefore, the shorter the total transmission time, the less the user will have to

pay. If the application uses TCP to send the image, it will not have such flexibility. In fact, the transmission can take a very long time when the channel bit error rate is high.

We have also conducted an experiment on the CDPD emulator test bed to compare the performance of the PITP systems with a modified PITP system that employs the TCP protocol. In the experiment, we used the uniformly prioritized Lena image. For a fair comparison, we set up the PITP in such a way that all missing segments were re-transmitted. For each experiment, the Block Error Rate (BER) of the CDPD channel was varied so that we could compare the performance of PITP and TCP for different BERs. The experimental results are shown in Figure 34.

Figure 34: PITP Vs TCP Performance Comparison on the CDPD Emulator Test Bed



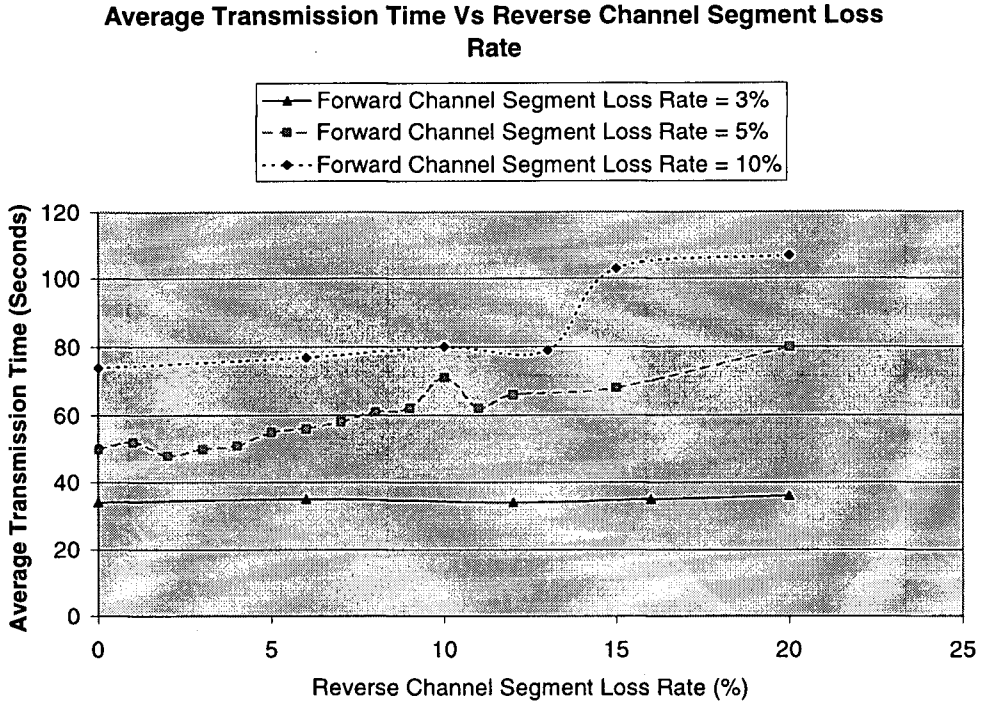
As shown in Figure 34, the PITP system, on average, is about 32% faster than the modified PITP that uses the TCP protocol. In fact, the gain shown in Figure 34 is the performance gain of the UDP over the TCP. The reason that TCP is slower is that the

TCP protocol uses a sliding window for orderly flow control. However, the PITP system uses the UDP which does not have a sliding window or an orderly (sequential) transmission restriction. All segments will be sent to the CDPD modem as long as space is available, and the CDPD modem/network is allowed to control the flow.

5.1 Reverse Channel Error Rate

We have conducted an experiment to test the impact of the reverse channel error rate on the performance of the PITP. We used the uniformly prioritized Lena image. In the experiment, we simulated some missing segments in the forward channel and we have setup the re-transmission algorithm in such a way that all missing segments were re-transmitted. Then we varied the reverse channel segment loss rate to see how it would affect the transmission time. The forward channel segment loss rate could not be set too low, because we would then not have enough lost segments to evaluate the impact. As shown in Figure 35, when the forward channel segment loss rate is 3%, the transmission time stays almost constant at around 35 seconds for the reverse channel segment loss rate between 0 to 20%. Therefore, in the experiment, we set the forward channel segment loss rate to 5% and 10%, respectively. We repeated the simulation 100 times for each reverse channel segment loss rate and averaged the transmission time. Figure 35 shows the experimental results.

Figure 35 : Reverse Channel Simulation Results



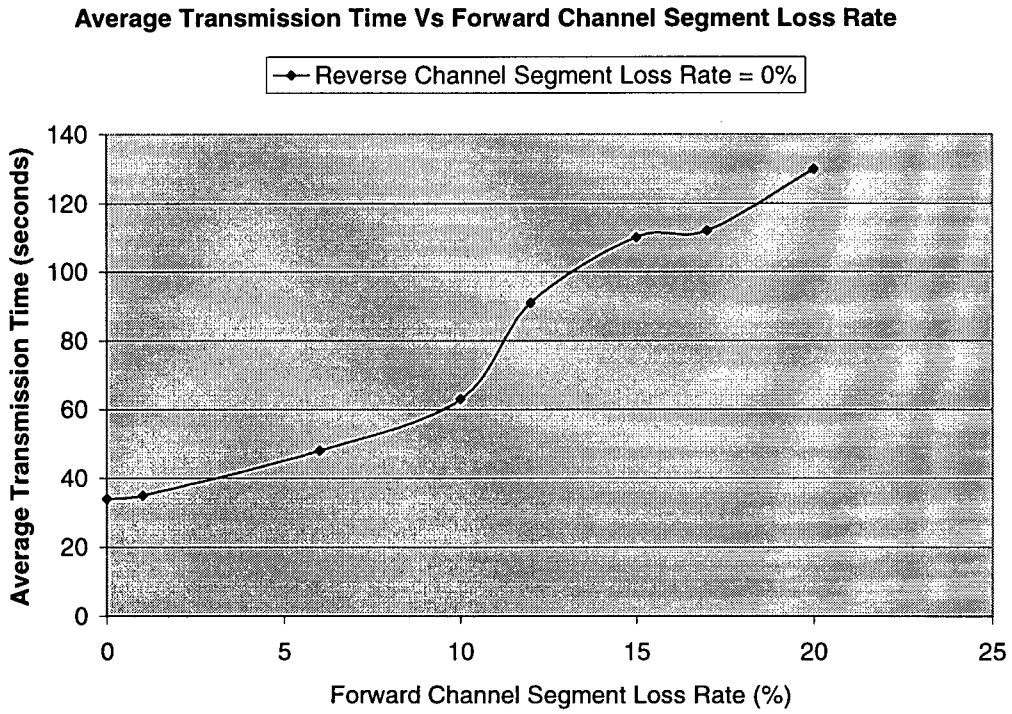
As shown in Figure 35, the performance of the PITP is not strongly affected by the reverse channel segment loss rate. In both cases, the average transmission time only increased by 30 seconds as the reverse channel segment loss rate increased to 20%. Since the PITP is a selective re-transmission protocol, the PITP rarely uses the reverse channel except for sending re-transmission requests. When the forward channel segment loss rate was 5%, the average transmission time increased slightly as the reverse channel segment loss rate increased. When the forward channel segment loss rate was 10%, the average transmission time increased significantly when the reverse channel segment loss rate increased to 15%. This was because the reverse channel segment loss rate was set so high that many re-transmission requests were also lost during the transmission. As a result, the lost re-transmission requests were re-sent following specific timeout periods.

For some wireless networks, the wireless system will not allow users to use the channel when the channel bit error rate reaches a certain level. For example, the CDPD network will ask the M-ES to search for other channels for transmission when the channel BER is higher than 10%.

5.2 Forward Channel Error Rate

The objective of the forward channel error rate experiment is to test the impact of the forward channel error rate on the performance of the PITP. In the experiment, we set the reverse channel segment loss rate to 0% so that only the forward channel segment loss rate would affect the transmission time. We repeated the simulation 100 times for each forward channel segment loss rate and averaged the transmission time. The experimental results are shown in Figure 36.

Figure 36 : Forward Channel Simulation Result

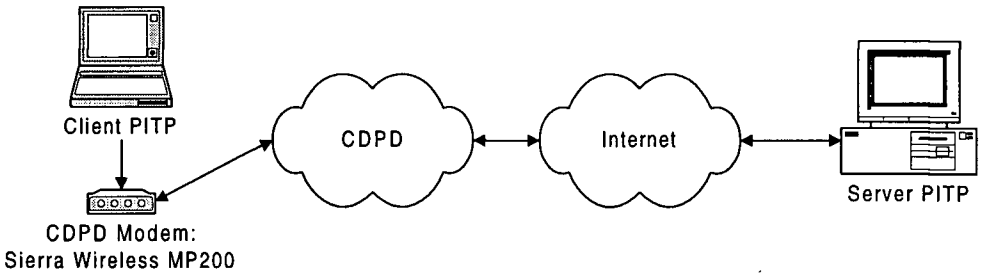


As shown in Figure 36, the forward channel segment loss rate affects the performance of the PITP significantly more than the reverse channel segment loss rate. The average transmission time increased by more than 100 seconds as the forward channel segment loss rate increased to 20%. This is because all scans (including the missing ones) are sent to the Client PITP system on the forward channel.

5.3 PITP Test on the Real CDPD Network

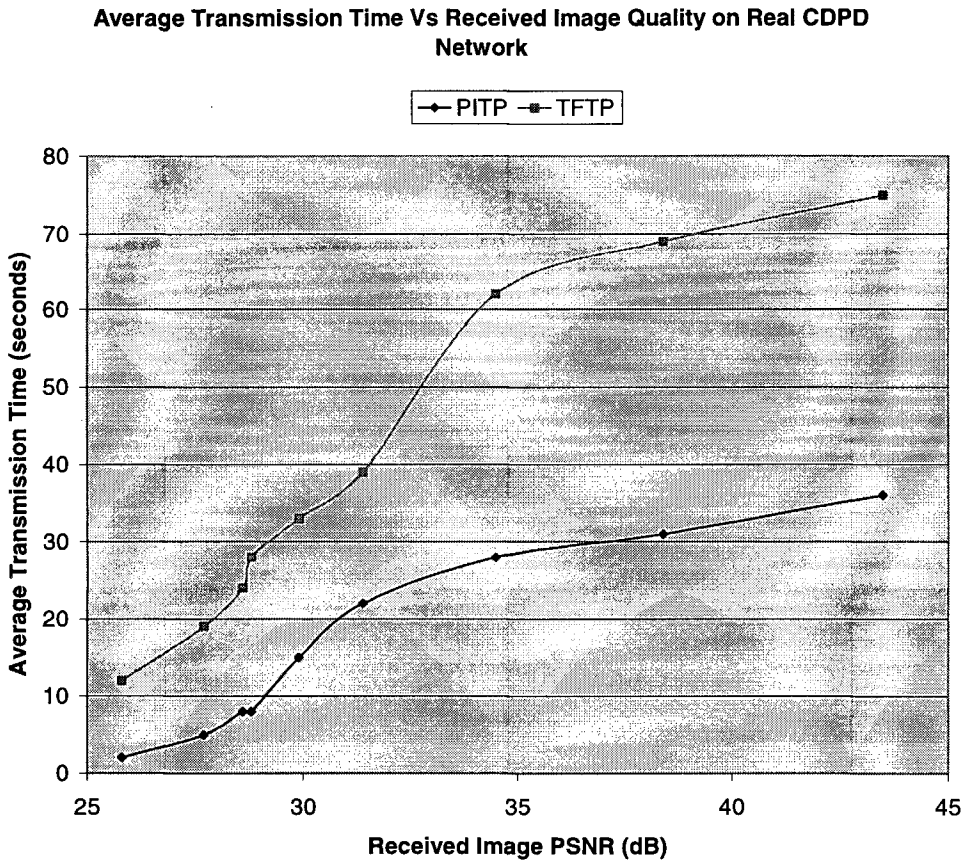
A real CDPD transmission experiment was conducted to evaluate the performance of the PITP system. In the experiment, we used a CDPD modem from Sierra Wireless. The model for the CDPD modem is MP200. The test bed setup is shown in Figure 37.

Figure 37: Real CDPD transmission test bed



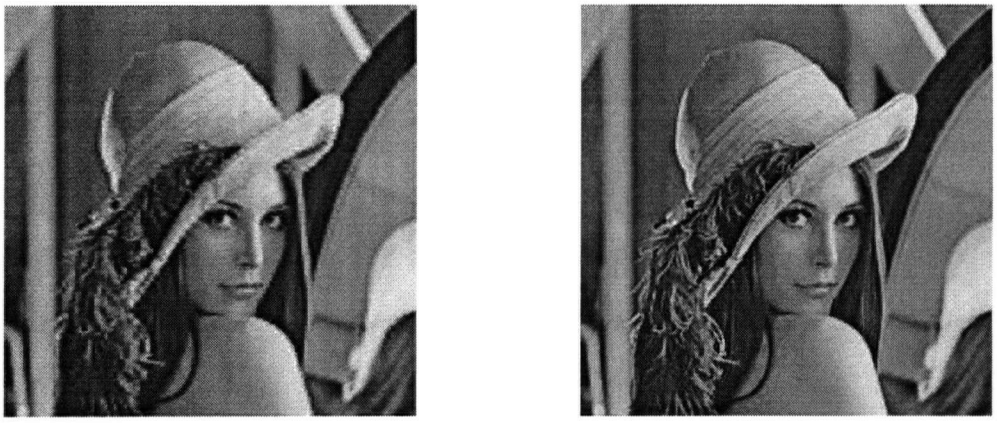
In the experiment, we compared the average transmission time of the PITP with the commercially known FTP software, the TFTP Server 32 Pro, Version 1.0. to transmit the Lena image which has been prioritized by scan importance over the scan size. We have repeated the experiment ten times for each received image quality. The experimental results are shown in Figure 38.

Figure 38: Average transmission time of PITP and TFTP Pro 32



As shown in Figure 38, the PITP has much better performance than the TFTP. With 28 seconds of transmission time, a user will receive an image with a PSNR of 34.5 dB if he/she uses PITP, and he/she will receive an image with a PSNR of only 28.8 dB if he/she uses the TFTP software. The received images are shown in Figure 39.

Figure 39: Received images from TFTP and PITP



The image on the left of Figure 39 is the TFTP received image, and the image on the right is the PITP received image. Clearly, the PITP image has substantially better quality than the TFTP image. If we want to receive an image with a PSNR of 43.5 dB, it will take the PITP and TFTP 36 seconds and 75 seconds, respectively.

The PITP is faster than the TFTP Server 32 Pro due mainly to three bottlenecks associated with TCP transmission: 1) a small CDPD modem input buffer size, 2) a slow CDPD network and 3) a TCP sliding window constraint. As the PITP and the TFTP use the same CDPD modem/network, the TCP sliding window constraint appears to be the only bottleneck. The PITP employs the UDP, which does not maintain sliding window flow control. As long as the CDPD modem has available memory, the application can keep sending data to the UDP socket for transmission. This UDP's efficient use of the CDPD wireless network is responsible for most of the gain achieved by PITP.

Chapter 6 Conclusions

The progressive image transmission protocol (PITP) is well suited for transmission of JPEG compressed images over the CDPD network. The PITP employs a JPEG progressive encoding method, the UDP and the Segment Re-transmission algorithm. The PITP system provides the user with good tradeoffs between image transmission time and reproduction image quality. The Segment Re-transmission algorithm is very flexible in the sense that the performance of the corresponding PITP system can be controlled significantly by optimizing the algorithm's parameters. Simulation results show that the PITP provides a much better performance than TCP/IP. This is confirmed by the real CDPD transmission experiment, where the PITP user can receive a better image than the user of the commercially known FTP software, the TFTP Server 32 Pro.

Bibliography

[1] E.A. Quincy and R.J. Achatz, *Performance Prediction of GSM Digital Cellular Speech and Image Transmission.* IEEE Pacific Rim Conference on Communications Computers and Signal Processing, pp. 26 – 31, 17 - 19 May 1995.

[2] Ricardo L. de Queiroz and K. R. Rao, *HVS - Weighted Progressive Image Transmission Using the Lapped Orthogonal Transform*, J. Electronic Imaging, Vol. 1, pp. 391 - 395, July 1992.

[3] Limin Wang and Morris Goldberg, *Progressive Image Transmission by Transform Coefficient Residual Error Quantization*, IEEE Transaction on Communications, Vol. 36, No.1, pp. 75 – 87, January 1988.

[4] G. Mongatti, L. Alparone, G. Benelli, S. Baronti, F. Lotti, A. Casini, *Progressive Image Transmission by Content Driven Laplacian Pyramid Encoding*, IEEE Proceedings-I, Vol. 139, No. 5, pp. 495 – 500, October 1992.

[5] John Agosta and Travis Russell, *CDPD: Cellular Digital Package Data Standards and Technology.* McGraw Hill, 1996.

[6] CDPD Forum, *Cellular Digital Package Data Implementor Guidelines and System Specification Release 1.1*, January 1995.

[7] John Gallant, *The CDPD Network*, EDN, pp. 40 – 48, October 13, 1994.

[8] Edward R. Dougherty, *Digital Image Processing Methods*, Marcel Dekker Inc, pp. 261 – 325, 1994.

[9] G. Lyengar, S. Panchanathan and M. Goldberg, *A Software tool for Progressive Image Transmission*, Communication '93, Technical Program Conference on Communication, pp. 527 – 531, 23 - 26 May 1993.

[10] Borko Furht, Stephen W. Smoliar, Hongjiang Zhang, *Video and Image Processing In Multimedia Systems.* Kluwer Academic Publishers, 1995.

[11] CCITT, *ISO/IEC 10818-1:1994(E), Information Technology - Digital Compression and Coding of Continuous - Tone Still Images: Requirements and Guidelines*, First Edition, February 15, 1994.

[12] Khalid Sayood, *Introduction to Data Compression*, Morgan Kaufmann Publishers, Inc, 1996.

[13] K.R. Rao, J.J. Hwang, *Techniques and Standards for image, Video, and Audio Coding*, Prentic Hall, 1996.

[14] R.J. Clarke, *Digital Compression of Still Images and Video*, Academic Press, 1995.

[15] IBM Corporation, *TCP/IP Tutorial and Technical Overview*, 1995.

[16] William Stallings, *Data and Computer Communications*, Macmillan Publishing Company, Second Edition, 1988.

[17] Dr. Kallel, Dr. Kossentini, Dr. Rabab Ward, Robert Link, Michael Wong, Jaehan In, Sinclair Chi, Jun Zhou, Shahram Shirani, Tamarasi Dias, *Efficient Image Transmission Strategies for Low Bit Rate Wireless Application First Quarter Progress Report,* Department of Electrical and Computer Engineering, The University of British Columbia, submitted to ASI and Mobile Data Solution Incorporated, April 1997.

[18] Dr. Kallel, Dr. Kossentini, Dr. Rabab Ward, Robert Link, Michael Wong, Jaehan In, Sinclair Chi, Jun Zhou, Shahram Shirani, Tamarasi Dias, *Efficient Image Transmission Strategies for Low Bit Rate Wireless Application Second Quarter Progress Report,* Department of Electrical and Computer Engineering, The University of British Columbia, submitted to ASI and Mobile Data Solution Incorporated, July 1997.

[19] Stardust Technologies, *Windows Sockets 2 Application Programming Interface, An Interface for the Transparent Network Programming Under Microsoft Windows*, Revision 2.2.0, May 1996.

[20] W. Richard Stevens, *Unix Network Programming*, Prentic Hall, 1990.

[21] Douglas E. Comer, *Internetworking with TCP/IP*, Prentic Hall International Editions, 1991.

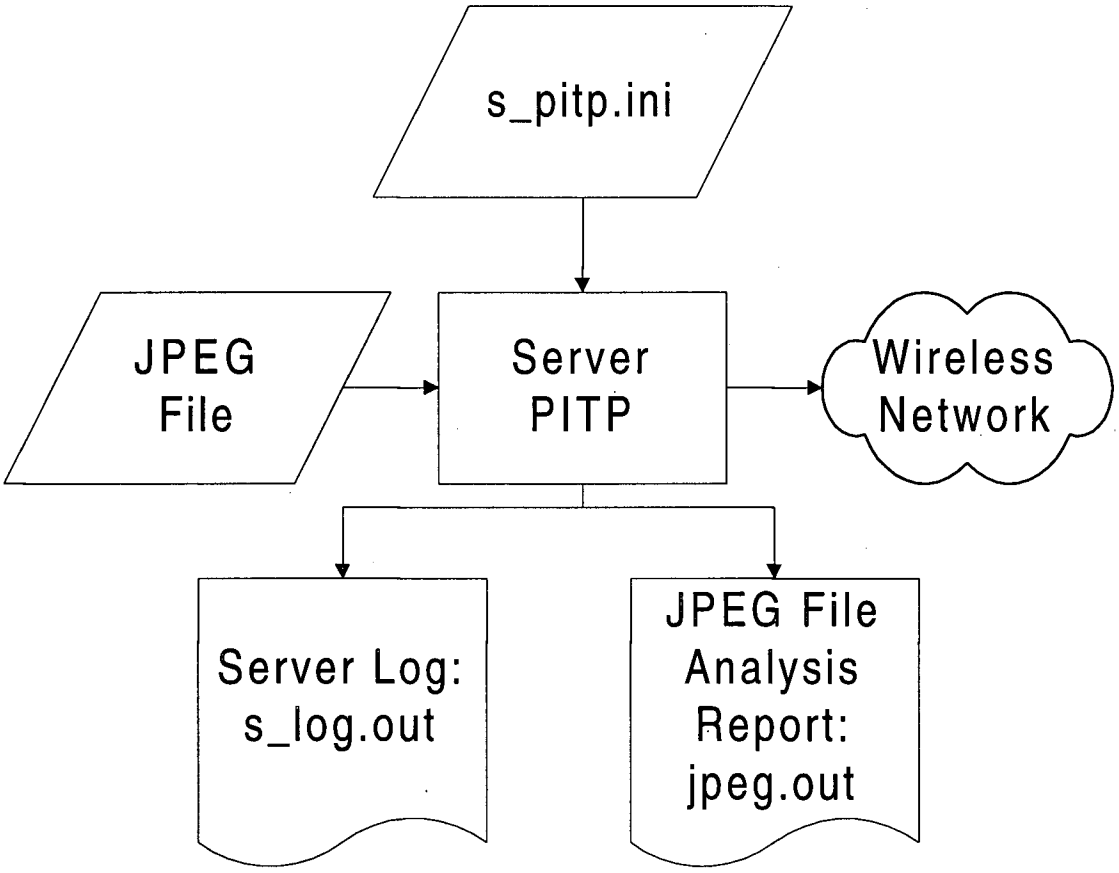
[22] Jaehan In, Shahram Shirani, and Faouzi Kossentini, *On RD Optimized Progressive Image Coding Using JPEG*, IEEE Transaction on Image Processing, submitted in November 1997.

Appendix 1 The Implementation of PITP

Server PITP System:

The Server PITP and the Client PITP Systems have been implemented on a PC platform. Figure 40 shows the high level process diagram of the Server PITP.

Figure 40: High Level Process Diagram of the Server PITP



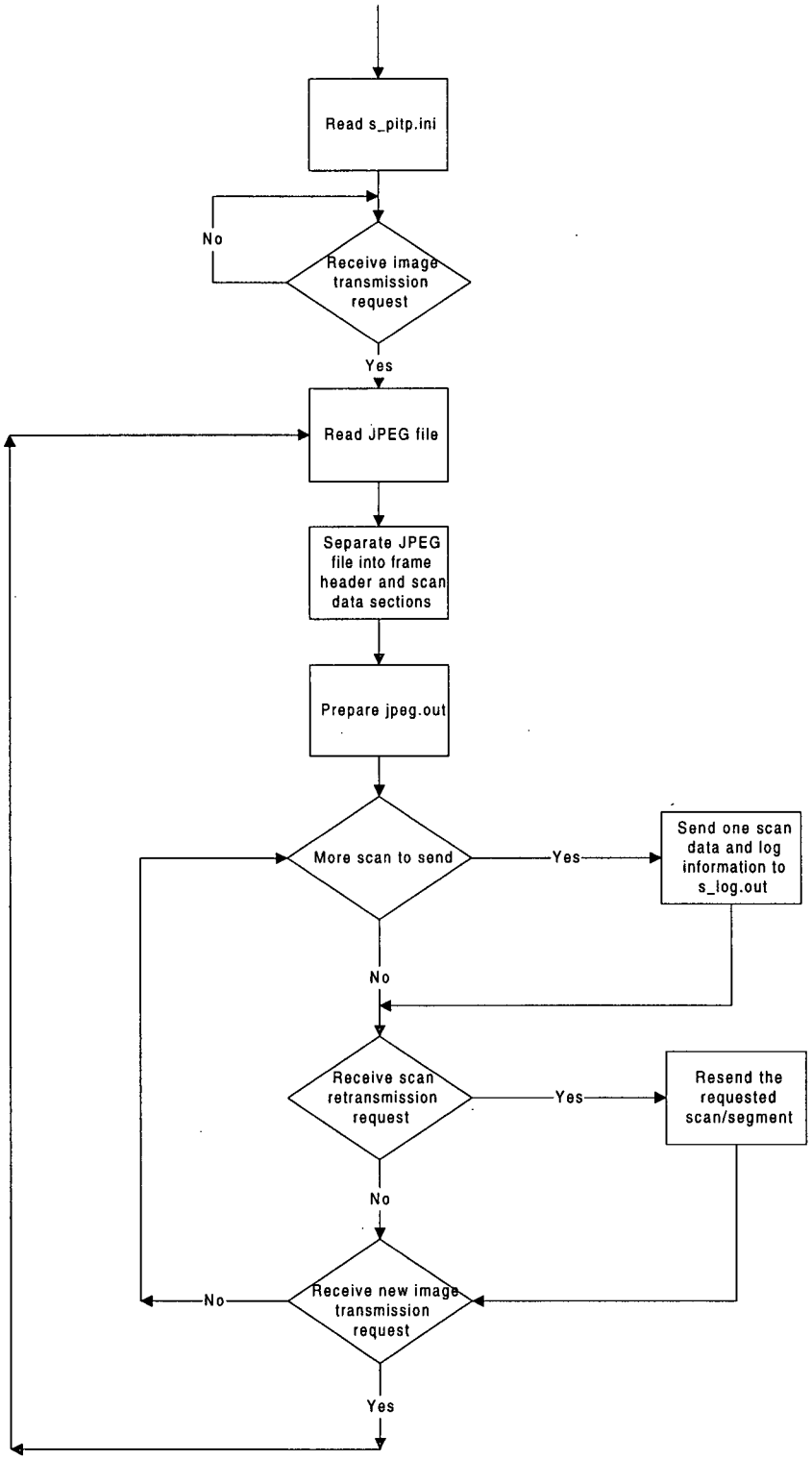
There are several components associated with the Server PITP. The purpose of each component is described as follows:

- JPEG image: this is a JPEG progressively encoded image.

- s_log.out: This file contains all the important process information during the transmission period, such as the total number of scans and their sizes. This file is used mainly for debugging purposes.
- jpeg.out: A JPEG progressively encoded image which consists of a number of scans, each scan separated by a unique marker. The Server PITP separates the JPEG progressively encoded image into a JPEG frame header section and scan data sections. The JPEG frame header section includes the quantization table and other data, such as component-specification parameters. The scan data sections include the Huffman tables for each scan, variable length codes, etc. The Server PITP will store the location of each section and output the offset information to the jpeg.out file. The Server PITP will then send each scan to the client program.
- s_pitp.ini: It contains the port ID that the server should use. In order to send a UDP datagram to the server, the client has to know the server's port ID and its IP address. The IP address identifies the physical location of the server in the wireless IP network. The port ID identifies the Server PITP application process so that the transport layer of the server knows where to deliver the datagram. Therefore, the server's IP address and the port ID as a whole are unique in the wireless network. The Server PITP will first wait for a client to make a connection by sending an image transmission request. Based on the request sent by the client, the server can determine the client's IP address.

Figure 41 shows the detailed flow diagram the Server PITP.

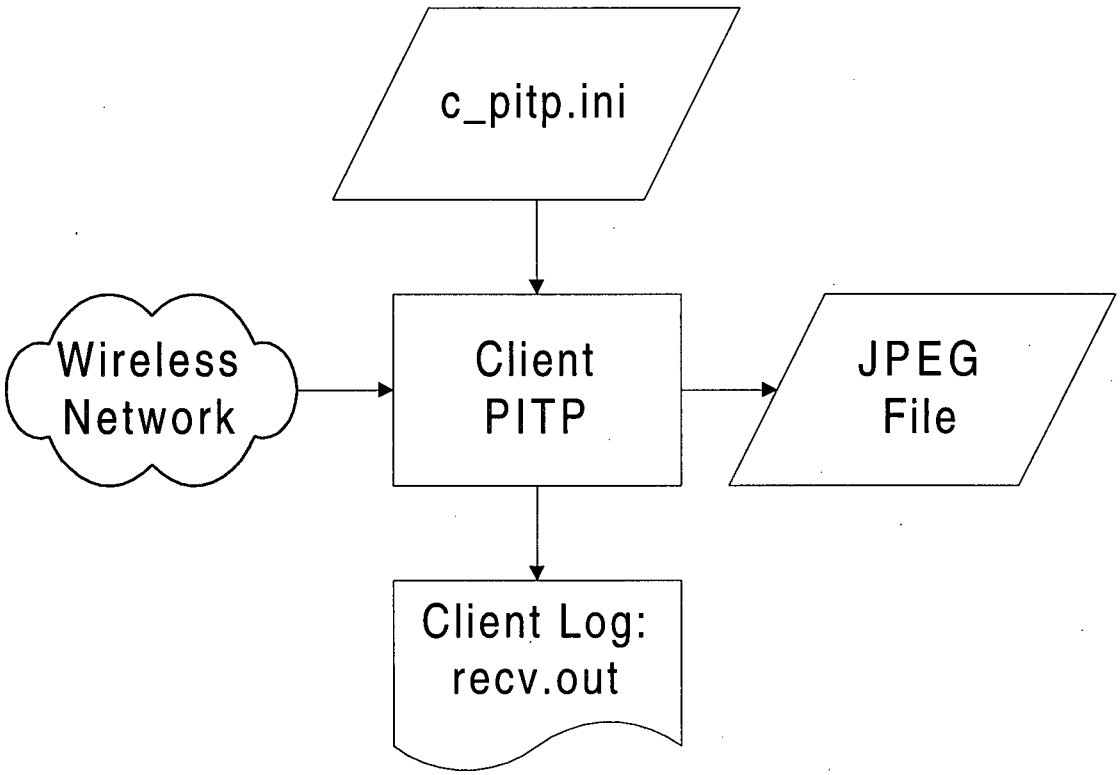
Figure 41: Detailed Flow Diagram of the Server PITP



Client PITP System:

Figure 42 shows the high level process diagram of the Client PITP.

Figure 42: High Level Process Diagram of the Client PITP



There are several components associated with the Client PITP. The purpose of each component is described as follows:

- JPEG image: This is the received JPEG progressively encoded image.
- recv.out: This log file contains all the important information collected during the transmission process, such as the number of scans received and the size of each scan.
- c_pitp.ini: This contains all the run time parameters for the Client PITP. The file contains:

a) tcp_mode flag: If the flag is set to yes, all missing segments or scans will be re-transmitted.

b) segment_timeout_mult: This will multiply with the segment end-to-end delay to estimate segment timeout timer interval, which will be discussed in Section 4.4 .

c) data_timeout_value: This is the data timeout timer, which will be discussed in Section 4.4 .

d) server_ip_address: This is the IP address of the image server.

e) port_id: This is the server port ID, and it should be the same as the port ID as specified in s_pitp.ini file.

f) channel_delay_weight, scan_priority_weight and max_channel_delay: These parameters will be discussed in the 4.5 .

Figure 43 shows the detailed block diagram of the Client PITP System.

Figure 43: Detailed Block Diagram of the Client PITP

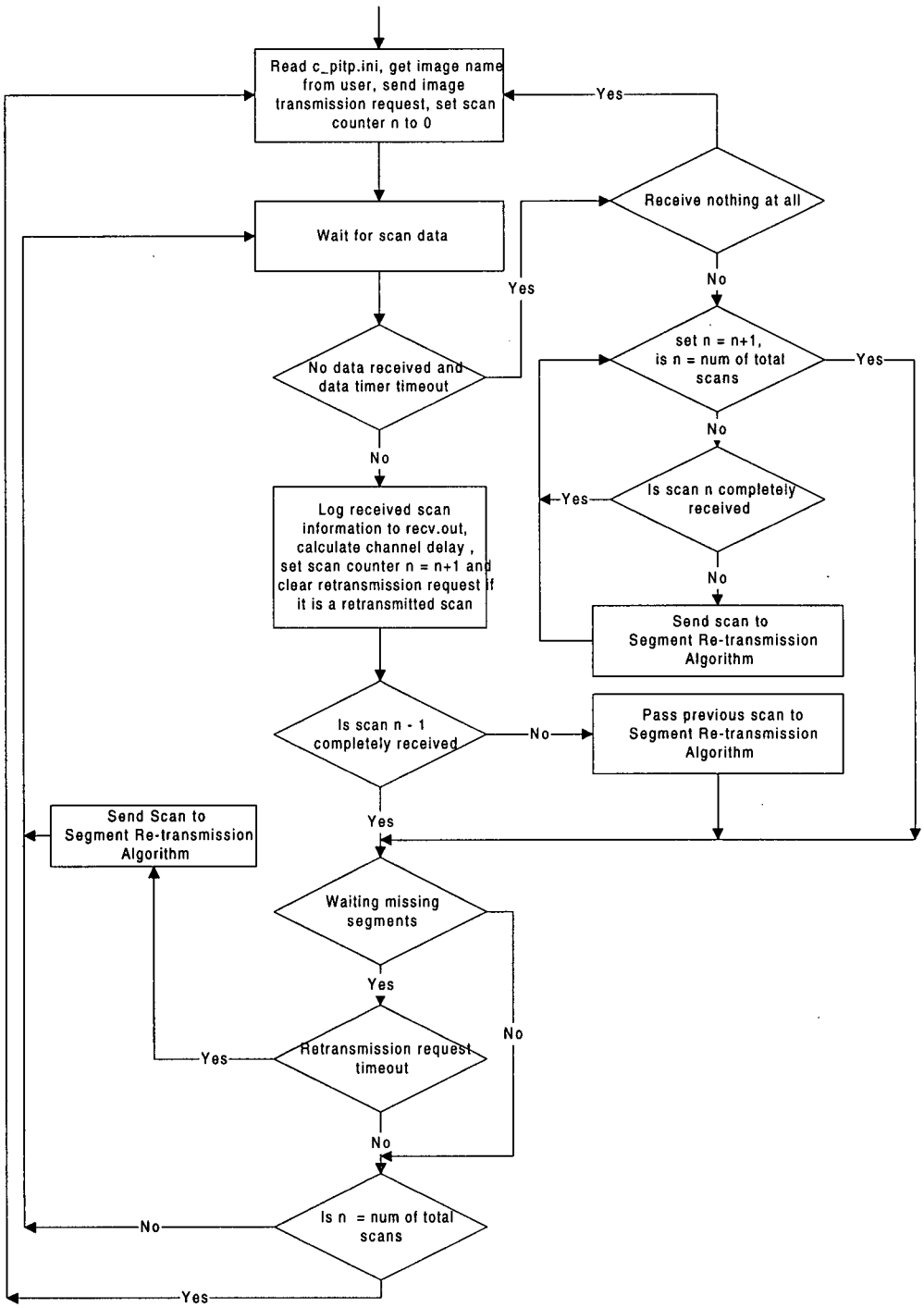
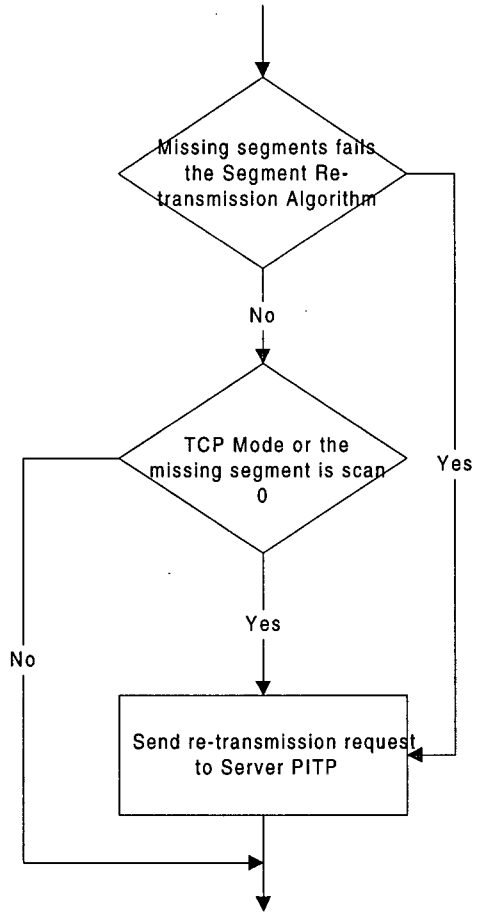


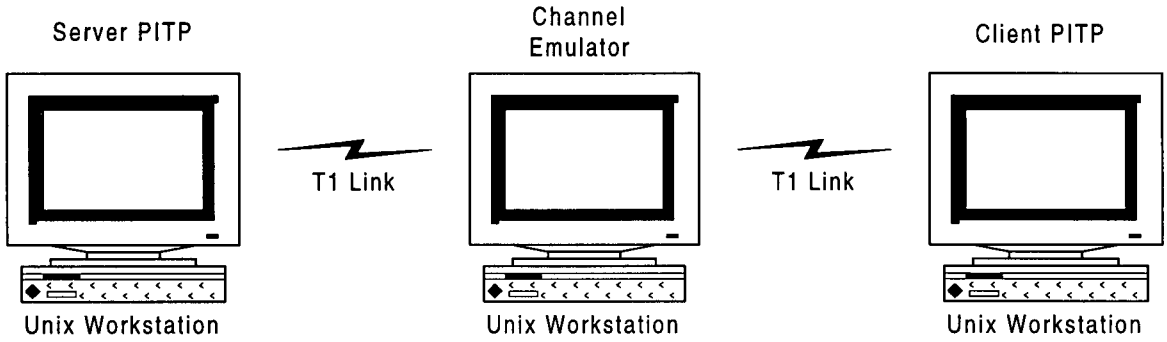
Figure 44: Segment Re-transmission Algorithm



Appendix II CDPD Simulator Test Bed

The CDPD simulator test bed tests the performance of the PITP under different segment loss rates. The test bed configuration is shown in Figure 45.

Figure 45 : CDPD Simulator Test Bed Configuration



The test bed consisted of three Unix Workstations. One workstation was used to simulate the Server PITP, one workstation was used to simulate the Client PITP, and the last workstation was used as a CDPD channel emulator. All three workstations were located in different places and connected by T1 links. These places were about 2000 feet apart of each other. The connections between the PITP workstations and channel emulator were full duplex. Therefore, the channel emulator could simulate errors in both the forward channel and the reverse channel. The channel emulator could inject errors to the segments received from both the Server PITP and the Client PITP workstations for simulating the segment lost.

The channel model parameters are:

- Forward Channel Segment Loss Rate (%): This parameter defines the segment loss rate for the channel that the Server PITP workstation communicates with the Client PITP workstation.

- Reverse Channel Segment Loss Rate (%): This parameter defines the segment loss rate for the channel that the Client PITP workstation communicates with the Server PITP workstation.
- Forward Channel Load Rate (%): This parameter defines how much channel capacity that the Server PITP can use to communicate with the Client PITP. That parameter is just for simulating the forward channel load. The higher the number, the longer the time that the Server PITP can occupy the channel and the smaller end-to-end delay.
- Reverse Channel Load Rate (%): This parameter is the same as the Forward Channel Load Rate except that it applies on the reverse channel.
- Channel Speed (bps): This parameter defines the speed of the wireless network. For example, if you want to simulate the CDPD channel, this number should be set to 19200 bps.

The CDPD simulator reads these parameters from an initialization file, and it simulates the channel characteristics based on these parameters.