Technical Report

UCAM-CL-TR-251
ISSN 1476-2986

Number 251

Multipoint digital video communications

Xiaofeng Jiang

April 1992

This technical report is based on a dissertation submitted December 1991 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Wolfson College.

ISSN 1476-2986

Summary

Ever since the emergence of high speed communication networks and fast signal processing technology, digital video has been attracting increased research interest. However, problems associated with its use in a multipoint communication environment have not been thoroughly investigated. In particular, these include the avoidance of congestion on multicast paths when multiple wideband sources are transmitting simultaneously, and the ability to interchange different format signals properly and efficiently. This dissertation addresses these issues with a two-level communications architecture.

The congestion issue at the network level is dealt with by several stream multicast path finding algorithms which are either centralised or distributed to suit various application environments. Different ways of integrating communication link capacities are investigated for supporting simultaneous transmission of broadband signals with minimum effect on network traffic and maximum success in path finding. Simulation results demonstrate performance improvements over conventional multicast path finding algorithms.

The format issue at the presentation level is dealt with by an intermediate format for general representation of digital video streams. Signals under this scheme are organised in a form to facilitate their interchange and scalable receiving in multipoint communication applications. Issues including frame segmentation and coding description are investigated. An experimental system implementing a simple version of the scheme is presented along with test results on picture quality degradation from conversion of various types and related timing characteristics.

Contents

List of Figures

List of Tables

Glossary

Page numbers in parentheses indicate where the terms first appear.

ASIC	Application Specific Integrated Circuit (88)
ATM	Asynchronous Transfer Mode — a virtual circuit and small fixed size packet based data transfer method (21)
B-list	A list of bandwidth requirements by a multicast group (39)
CAM	Content Addressable Memory (9)
CBR	Cambridge Backbone Ring — a 500 Mb/s slotted ring network (2)
CCIR	International Radio Consultative Committee (11)
CCITT	International Consultative Committee for Telephone and Telegraph (11)
CD-I	Compact Disc-Interactive — a compact disc based video system (14)
CD-ROM	Compact Disc Read Only Memory (12)
CD	Compact Disc (14)
CFR	Cambridge Fast Ring — a 75 Mb/s slotted ring network (2)
CIF	Common Intermediate Format — a video format defined by CCITT Recommendation H.261 (11)
CI	95% Confidence Interval — a statistical error range (46)
CMOS	Complementary Metal Oxide Semiconductor (95)
CMTT	Commission Mixte pour les Transmissions Télévisuelles et Sonores (13)
Codec	COder-DECoder — a device for signal format conversion (16)
DARPA	Defence Advanced Research Projects Agency (30)
DCT	Discrete Cosine Transform (8)
DFT	Discrete Fourier Transform (8)
DPCM	Differential Pulse Code Modulation (7)
DRAM	Dynamic RAM (93)
DSP	Digital Signal Processing (97)
DVI	Digital Video Interactive — a personal computer based video system (15)
ELV	Edit Level Video — a quality level specified by the DVI system (15)
FIFO	First-In First-Out (94)

FLC	Fixed Length Coding (9)
GOB	Group of Blocks — a term used by CCITT Recommendation H.261 referring to a video frame segment (11)
HDTV	High Definition Television (13)
HSLAN	High Speed Local Area Network (25)
IGVR	Intermediate General Video Representation (75)
IRS	Image Resampling Sequencer — a special image processor (97)
ISDN	Integrated Services Digital Network (1)
ISO	International Organisation for Standardisation (12)
JPEG	Joint Photographic Experts Group (12)
KLT	Karhunen-Loeve Transform (8)
MB	Macro Block — a term used by CCITT Recommendation H.261 referring to a video frame segment (12)
MCU	Multipoint Control Unit — a central control and switching equipment for a studio video conference (24)
MPEG	Moving Picture Experts Group (12)
NP-complete	The property that a problem can not be solved in polynomial time (26)
NVST	Network Video Stream Transceiver (91)
PLA	Programmable Logic Array — a circuit device whose function can be programmed after manufacture (9)
PLV	Presentation Level Video — a quality level specified by the DVI system (15)
QCIF	Quarter CIF (11)
RGB	A component signal format of colour television (11)
RS232	A common interface standard for data transmission (92)
SAM	Serial Access Memory (93)
SPN	Steiner Problem in Networks — a mathematical model for multicast path finding (26)
TTL	Transistor-Transistor Logic (95)
VLC	Variable Length Coding (9)
VLSI	Very Large Scale Integration (1)
VRAM	Video RAM — a device combining a DRAM and a static SAM (93)
YCrCb	A component signal format of colour television (11)
YIQ	A component signal format of colour television (15)

Abbreviations for algorithm names:

KMB	A Steiner tree approximation algorithm proposed by L. Kou, G. Markowsky and L. Berman (27)
RS	A Steiner tree approximation algorithm proposed by V. J. Rayward-Smith (28)

Chapter 1

Introduction

1.1 Background

Recent advances in two important areas of technology have had a great impact on communications services. The first is the appearance of fast digital signal processing. Real time processing of digital video images is now possible with special VLSI chips which have been designed since the introduction of several draft international standards on video coding [Ang 91b]. TV quality video can be obtained from a throughput of about 1.5 Mb/s. The second is the emergence of high speed communication networks. Many local area networks are now operating at the order of 100 Mb/s [Hopper 86, Ross 89], while research and experiments on 1 Gb/s (and upwards) wide area networks is currently very active [Greaves 91, Newman 88, Green 91]. Meanwhile in the telecommunications world, broadband ISDN aims to provide communications channels with various bandwidth levels from 2 Mb/s upwards [Handel 89].

The combined effect of these two developments is that a class of new communications services, namely multipoint digital video applications, has become viable in computer dominated networking environments. A typical example is computer workstation based video conferencing. Although digital video, as the most challenging communication medium because of its continuous wide bandwidth occupancy [Ang 91a], has been attracting increased research interests in *multimedia* applications, less attention has been paid to problems involved in its use in a multipoint environment.

The *Pandora* project [Hopper 90] created in 1988, as a joint project between the Olivetti

Research Laboratory at Cambridge and the University of Cambridge Computer Laboratory, is a leading effort to bring digital video into a computer workstation and fast network based communication environment. It implements video processing functions in a special hardware unit called the *Pandora's box* through which the related workstation, network interface, display monitor, and other audiovisual devices are interconnected. Digital video streams may come from a locally attached camera, discs, or a remote source in the network, and travel to reciprocal outputs.

Based on a 75 Mb/s local area network, the *Cambridge Fast Ring* (CFR) [Hopper 88], the system first started with point-to-point video services, such as videophone, video mail, and TV reception. These applications quickly became useful communication tools for daily work in the local community. After this early success, video conferencing as an application of multipoint video communications was added. Meanwhile, for the networking infrastructure, a 500 Mb/s optic fibre wide area network, the *Cambridge Backbone Ring* (CBR) [Greaves 90], was used to interconnect local CFRs. The Pandora system has thus evolved into a multipoint digital video communication environment operating through interconnected high speed networks.

Unlike conventional point-to-point narrowband applications, such a kind of multipoint digital video communication environment has its distinctive properties: (a) the demanding bandwidth requirements of digital video signals, and (b) the manifold nature of data transmission volume in proportion to the number of participants in a communication session. These characteristics introduce a set of new problems such as congestion avoidance and signal interchange which motivate this research.

1.2 Context

To provide mechanisms for supporting multipoint video communications, a two-level architectural approach was adopted in this research.

As multipoint communications traditionally involve a multicast routing problem, *stream multicast* facilities are introduced at the network level to address the associated congestion problem. These take the form of a set of new path finding algorithms with various ways of integrating communication link capacity. Both centralised and distributed schemes are investigated in order to assess suitability for various environments.

Video equipment located at different sites in a multipoint communication session may be heterogeneous. Therefore, interchange of signal formats is unavoidable. This problem is addressed by an intermediate video stream representation scheme at the presentation level, which is shown by analysis and experiments to facilitate easy and scalable interchange of digital video signals.

As outlined at the end of this dissertation, the above two components may be combined with a general purpose protocol engine which together form a complete architecture to facilitate a new range of multi-service broadband communications.

1.3 Outline

Chapter 2 starts with a discussion of practical video coding types and various standards and experimental systems which are essentially hybrid schemes of these basic coding techniques. From the wide spectrum of video signals, it is observed that an intermediate video stream representation method is necessary and the requirements for such a scheme are then presented.

Following the two-level architecture, Chapter 3 moves on to the discussion of multipoint communications environments with the introduction of a case study on different types of video conferencing. Conventional approaches to multicast path finding are surveyed and the problems which they cause in the new environments are investigated. This leads to requirements for a stream multicast mechanism.

The approach to stream multicast path finding is given in Chapter 4. New schemes are based on two typical Steiner tree algorithms. Different ways of integrating communication link capacity are studied by simulation to find their relative performance and effects on networks. The computational complexity of the algorithms are analysed and verified by simulation results. It is found that the new algorithms can also solve some unicast routing problems easily, which is illustrated with examples at the end of the chapter.

Chapter 5 discusses the stream multicast path finding algorithms for the distributed scenario. Loose conditions, such as a global identification method, and knowledge at each node about the available bandwidth on its connected links, are assumed in the distributed environment. The operation of the algorithms is based on packet exchange between neighbouring nodes. Link capacity integration methods and algorithm complexity are again pursued and similar results to those of the centralised case are shown from the simulations.

Having addressed the stream multicast mechanisms, Chapter 6 returns to the higher level issues of signal formats. An intermediate general representation scheme of video streams is presented with discussions on frame segmentation, coding description, and various packet formats. A general decoder model for the scheme is described and some implementation issues are discussed. The features of the scheme are analysed with respect to the requirements stated in Chapter 2.

Chapter 7 presents an experimental system for video stream processing. Its primary function is as a multipurpose stream transceiver to allow experimentation with the video representation scheme described in the previous chapter. Design decisions for various parts of the system are discussed along with the functions and performance of the system. Experimental results are reproduced regarding quality deterioration associated with conversion of various pictures and timing features. A variety of possible applications of the system are also illustrated at the end of the chapter.

Chapter 8 completes the dissertation with concluding remarks and suggestions for future work.

Chapter 2

Digital Video

2.1 Digital Representation of Video Signals

Colour video systems are based on the *tri-stimulus theory* of colour reproduction [Benson 86], which states that any colour can be reproduced by appropriate mixing of three *primary colours*. For this reason, the use of three colour channels in a video system is common although there is a variety of choices for the base colour signals.

According to the organisation of its colour channels, a video signal can be identified as *composite* or *component*. The colour channels of a composite signal are mixed in a fixed format, and can not be processed separately without decoding the composite signal first. In contrast, channels of a component signal are always transmitted and stored separately, and can therefore be processed independently. The colour channels are also called colour components.

A digital composite signal is obtained from directly *digitising* its analogue form while a digital component signal is generated by digitising each of the analogue colour components. Because a major benefit of digital representation is easy manipulation, digital component video is naturally advantageous over its composite counterpart, and has been the dominating form in the computer world since its introduction.

Although digital video provides many advantages [Sandbank 90], one major disadvantage is the enormous bandwidth required when compared to the analogue form, as the *Nyquist*

Theorem[1] suggests. Therefore, digital video is often used in a compressed form, especially for large format or high definition systems.

Before starting the discussion of general representation of video streams, it is necessary to consider the basic coding or compression algorithms commonly used and other standards or schemes being proposed. These are the subjects of the following two sections.

2.2 Types of Digital Video Coding

There are numerous algorithms proposed for coding digital video signals [Netravali 80, Musmann 85, Singhal 90], although they fall into a few general classes. It is outside the scope of this thesis to examine all of them. The following subsections present only those currently in active use and regarded as efficient [Lippman 91] for motion pictures. The practical operational schemes and standards, to be discussed in the next section, are combinations of some of them, and are known as *hybrid coders*.

2.2.1 Predictive Coding

Predictive coding is effective when scenes of a motion picture change smoothly. It is based on the assumption that a similar area to that currently being coded exists in either its spatial or temporal neighbourhood, and hence may be used as a reference. The difference between the coded area and the reference is usually very small because of the similarity. Compression is therefore obtained by sending the difference and an indicator of the reference instead of the data in the whole area.

A predictive coder can be expressed generally as

$$D(x) = I(x) - P(x) \tag{2.1}$$

where $I(x)$ is the actual sample value, $P(x)$ is an estimation or *predictive value* from the corresponding sample(s) in the reference, and the difference $D(x)$ is usually called a *prediction error*. Linear predictors are normally employed which calculate a weighted average of the neighbouring samples in the reference as a predictive value:

$$P(x) = \sum_i c_i I(x_i). \tag{2.2}$$

[1]The Nyquist Theorem states that the sampling rate for a digitising process must be at least twice the highest frequency of the signal to be digitised for the proper reproduction of the original signal.

By adjusting predictor coefficients c_i to minimise prediction errors, a predictive coder can be made adaptive to improve its performance. A number of such algorithms can be found in [Jayant 84, Sage 71, Honig 84].

The simplest predictive coding is *differential pulse code modulation (DPCM)* [Jayant 84], where spatial references are used. The area for comparison is reduced to one sample, and $P(x)$ is the value of the sample prior to the current one.

To explore the strong temporal correlation of video signals, *motion compensation* is a typical example of predictive coding, and is commonly used in current standards and research/commercial systems. This technique normally functions on rectangular areas of frames or *blocks*. Unlike other predictive coding schemes, a reference block is not located at a predefined position relative to the coded block, but can be anywhere in a *search space*. A reference block is searched with a block-matching technique, such as the minimising mean square (or mean absolute) difference. Although the number of matching operations is extremely large, hardware implementation is feasible due to the regular computational structure that can be supported by VLSI technology [Yang 88].

The steps following a block search are straightforward. The displacement of the reference block from the original one is referred to as a *motion vector* $\overline{mv}$, and each predictive value of the block is then obtained from

$$P(x) = I(x + \overline{mv}). \tag{2.3}$$

Due to its block-based operation, motion compensation is particularly suited for use in conjunction with other block based coding techniques, such as transform coding to be discussed in the next section. This combination is also the basis of most hybrid coding schemes [Lippman 91].

2.2.2 Transform Coding

In transform coding, an image or frame is partitioned into non-overlapping blocks, of which sizes may range from 4×4 to 32×32 samples. Each block is then transformed by the following matrix multiplication into a spectral-like representation,

$$Z = C^T X C \tag{2.4}$$

where X and Z are $N \times N$ blocks of sample data and transform coefficients respectively and C is an $N \times N$ transform basis matrix[2]. When a transform is regarded as *orthogonal*[3], the energy of the signal is concentrated in the "low frequencies" in the transform domain, which is the potential for compression. Among orthogonal transforms, such as discrete Fourier (DFT), discrete Cosine (DCT), Hadamard and Karhunen-Loeve (KLT) [Netravali 88], KLT is the optimum in terms of the mean square error, but not easily implementable.

DCT is close to the optimum if certain loose conditions are met. It also has the ability to reduce *blocking effects*, which refer to the poor performance at edges of blocks, associated with most block transform techniques [Singhal 90]. Therefore, DCT has become the transform of choice. Its real time implementations for motion pictures are supported by recent advances in VLSI technology [Jutand 87, Vetterli 86, Sun 87, Liou 88, Sun 89a].

The transform basis matrix of DCT is defined as:

$$c_{pk} = \begin{cases} \left(\frac{2}{N}\right)^{1/2} \cos\left[\frac{(2p-1)(k-1)\pi}{2N}\right] & p = 1..N,\, k = 2..N \\ N^{-1/2} & p = 1..N,\, k = 1 \end{cases} \tag{2.5}$$

The small block size of $N = 8$ is usually preferred because it is appropriate when using combined run-length and amplitude entropy coding [Wallace 91].

Adaptive schemes are also possible for transform coding if quantisation steps for transform coefficients can be dynamically adjusted to suit particular characteristics of pictures. They are normally used to achieve higher compression rates.

To restore the original sample data, an inverse transform is also calculated by matrix multiplications,

$$X = CZC^T. \tag{2.6}$$

Since the equation has exactly the same form as that of a forward transform, a single architecture can be used for a transform in both directions.

[2] C^T denotes the transpose matrix of C.

[3] Orthogonality means all vectors of a transform basis matrix meet the following condition,

$$\sum_{k=1}^{N} c_{qk} c_{pk} = \begin{cases} 0 & p \neq q \\ A & p = q \end{cases}$$

where A is a constant and c_{pk} is the p-th row k-th column element of C. This property ensures that each basis vector identifies totally different type of detail from the sample data matrix X to be contained in the coefficient matrix Z.

2.2.3 Entropy Coding

The occurrence probabilities of signal levels from a discrete-valued information source are normally different from each other. By exploring this difference of information contents, entropy coding can remove the redundancy of the source without loss of information. This is achieved by means of a *variable length coding* (VLC) procedure which assigns shorter codewords to more frequently occurred signal levels. Taking a 4-level signal source for example, the codewords assigned to the signal levels in decreasing order of occurrence probability would be 1, 01, 001, and 000. If the first code appears more often than the others, the average code length would be less than two, which is the fixed length of binary coding in this case.

As a table look-up method is normally used in the decoding, a problem may arise if a source has a large number of signal levels, which result in an impractical table length. This is solved by combining VLC and FLC (*Fixed Length Coding*): a small number of the most frequently occurring codes are encoded by entropy coding, the rest by normal binary coding.

It is difficult to process a signal further after entropy coding because boundaries between codes are concealed. Therefore, entropy coding is often the last encoding (or first decoding) stage of a hybrid coding scheme.

VLSI implementation of entropy coding is feasible by employing a barrel shifter and PLAs (*Programmable Logic Arrays*) or CAM/RAM (*Content Addressable Memory/Random Access Memory*) modules [Sun 89b].

2.2.4 Pyramid and Subband Coding

In pyramid coding, a picture is represented by a series of band-pass sub-pictures each sampled at successively lower rates. In its simplest form, an original picture is subsampled by a factor of two to produce a low-pass version. An error, or band-pass filtered, picture is obtained by first interpolating the low-pass picture to its original resolution and then subtracting it from the original. This procedure is repeated recursively to generate a sequence of band-pass pictures until usually only one pixel is left in the low-pass picture. Entropy coding is normally applied to the pyramid stack of the low-pass and band-pass pictures to achieve bit rate reduction.

Subband coding is similar except that the band-pass pictures are obtained from a set of parallel band-pass filters. Compared with pyramid coding, this method trades more coder complexity for higher operation speed. The parallel processing of subband coding requires more hardware devices but generates coded signals more quickly.

Pyramid and subband coding have received much attention in recent years because of their hierarchical representation of pictures, which can be exploited to save communication bandwidth and receiver resources in *scalable* systems [Lippman 89]. A picture at the receiver is progressively reconstructed from its band-pass sub-pictures. Once the receiver's resolution is reached, there is no need to send more data from the source.

2.2.5 Region Coding

Region coding is based on the assumption that a picture can be analysed into regions expressed from a set of image primitives, such as rectangular areas of constant colour, smooth shaded patches and textures. The coded parameters are the location of the region and the index of the applied primitive, and can normally be transmitted or stored more efficiently than the original image data, thus reducing the bit rate. The decomposition of a picture into regions is usually performed using a binary- or quad-tree algorithm where the picture is successively divided into smaller regions until a primitive that meets either the bandwidth constraints or the quality desired can be fitted.

Region coding is a highly "unbalanced" coding technique in that significantly more processing is required for encoding than for decoding. This is primarily due to the *region growing* step in the encoding operation where adjacent image patches that are distinct leaves of a tree are combined into a single patch. For this reason, there is no hardware implementation for a region coder yet. However, it is possible to exploit the unbalanced nature in this coding technique, which means a region decoder is relatively simple to build. This property is commensurate with the idea of designing the DVI system (to be introduced in a later section of this chapter) where region coding has been used as a key element for its video coding scheme [Keith 88].

2.3 Digital Video Standards and Proposals

The above motioned coding methods are the basic types. In practice, they are seldom used alone, but rather in combinations to obtain higher coding efficiencies, as seen in the following recently emerged standards and systems.

2.3.1 CCIR Recommendation 601

The International Radio Consultative Committee (CCIR) Recommendation 601 specifies a family of encoding parameters for digital television studios. It aims to establish an agreement on a digital code that is compatible with both 625 and 525 line systems, the two dominant television broadcasting standards in the world.

Video signals defined in the recommendation are of component form consisting of a luminance (Y) and two colour-differences (C_r and C_b). As in most colour component systems, chrominances are not necessarily maintained at the same resolution as luminance. C_r and C_b are therefore sampled at the half frequency of Y, resulting in a frame of 576×720 active Y samples and 576×360 active C_r and C_b samples[4]. Both luminance and colour-difference samples are encoded into 8-bit words.

The relationship between YC_rC_b and RGB signals is as follows,

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.511 & -0.428 & -0.083 \\ -1.905 & -0.339 & 0.511 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.7}$$

This is the basic standard for digital component video, and provides a common exchange format for studio production. However, since no compression schemes are specified, the video rate is high (166 Mb/s), and no practical programs or devices for network video applications use the standard directly.

2.3.2 CCITT H.261 — p×64 kb/s

H.261 is a video coding standard recommended by the CCITT Specialist Group on Coding for Visual Telephony. Because its purpose is to provide ubiquitous video services over the ISDN using current telephone transmission standards [CCITT 90], the data rates are defined as a hierarchy of p×64 kb/s (p = 1,2,...,30).

The video formats recommended are the Common Intermediate Format (CIF) and Quarter-CIF (QCIF). Because chrominance signals are sampled at a quarter rate to the luminance signal, a CIF frame has 288×352 Y samples and 144×176 of each of C_r and C_b samples. A QCIF frame is of a quarter size of a CIF frame.

To fit into the p×64 kb/s hierarchy, a hybrid compression scheme is applied to the raw video signal. CIFs and QCIFs are successively decomposed into Groups of Blocks (GOBs),

[4]Different parameters are used in the North American region. Details are shown in Table 2.1.

Macro Blocks (MB), and Blocks for the efficiency of the compression algorithm. A Block covers a rectangular area of 8×8 samples. An MB consists of four Y, one C_r, and one C_b Blocks while a GOB consists of 3×11 MBs. A CIF has therefore 12 GOBs, and a QCIF has 3.

There are two stages in the H.261 compression algorithm. The first is a combination of DCT and DPCM with motion estimation. By searching a reference MB in the previous frame, a predication is made for a MB of the current frame. The difference between the current and predicted MB is transformed into DCT coefficients, which together with the motion vector represent the current MB. An *intraframe mode* is also defined where prediction or motion estimation is bypassed so that DCT is performed directly on the original MB. This is useful for the first picture of a stream and others after a change of scene.

The second stage uses entropy encoding to further reduce the bit rate. Quantised DCT coefficients and various side information are coded by variable length coding.

A particular transmission rate in the hierarchy is obtained by adjusting the step size of the linear quantiser for the DCT coefficients. It is believed that QCIF with $p = 1$ or 2 is suitable for videophone applications, while a video conferencing service requires CIF with $p \geq 6$ [Liou 91].

2.3.3 JPEG and MPEG

JPEG (Joint Photographic Experts Group) is a joint effect of CCITT and ISO to develop a compression standard for continuous-tone still images. The Moving Picture Experts Group (MPEG) was formed later to examine video compression schemes with an acceptable quality in the range of 1 to 1.5 Mb/s, which are suitable for storage and retrieval on digital storage media such as CD-ROM. Because of the close relationship of the two groups, the compression algorithms are similar, and the rest of this section concentrates on the MPEG standard only.

The coding procedure is very similar to that used by the CCITT Recommendation H.261 because of the consideration for interchange of basic components. However, the relaxed bit rate (1.5 Mb/s compared to 64 kb/s in H.261) increases the possibility of defining a more manipulable and accessible stream, a major concern of the MPEG. This is mainly reflected by more picture types in the standard, namely *Intra (I), Predicted (P)*, and *Bidirectional/Interpolated (B)* pictures.

An I-picture is self-decodable, i.e. no information from other frames is used to encode the picture. A P-picture is encoded from the prediction errors in reference to a past I- or

P-picture, while a B-picture is predicted from both a past and a future I- or P-picture. Trade-offs are made by selecting a proper frequency of B-pictures in a stream so that the target bit rate is achieved while a reasonable granularity of random accessibility is maintained. It is found that for a large class of scenes a reasonable choice is to have I- or P-pictures at about 0.1 second interval [Le Gall 91]. This results in a frame sequence of the following pattern

I B B P B B P B B ... I B B P B B.

It is observed that the high compression gain on B-pictures is obtained at the price of long coding delay and large buffer requirement at both encoding and decoding sides. For the sequence in the above example, the coding delay is 4 frames and a buffer of 5 frames is required. These figures are twice those for H.261 where only forward prediction is allowed. Both parameters become even larger as frequency of B-pictures increases.

2.3.4 CMTT Recommendation

CMTT (Commission Mixte pour les Transmissions Télévisuelles et Sonores) is a joint CCIR/CCITT specialists group for digital transmission of television and sound. Its activities focus on coding digital video signals for contribution-quality applications in the bit rate range of 30 to 45 Mb/s [CMTT 89]. Its draft recommendation is mainly specified for broadcast purposes and has a basic feature of being able to deal with CCIR Recommendation 601 standard video input without subsampling.

The coding algorithm is again similar to H.261 with two important differences due to its different application area. Firstly, a *non-compensated inter-field* mode is defined because of interlacing in the input TV format. At the prediction stage of this mode, the reference block is taken from the previous field without motion compensation, i.e. the motion vector is zero. Secondly, macro blocks, to which motion compensation is applied, are composed of two (instead of four) adjacent 8×8 luminance blocks and two spatially corresponding colour-difference component blocks due to the different subsampling ratio of luminance and chrominance.

2.3.5 HDTV Proposals and Open Architecture Systems

High definition television (HDTV) has been widely regarded as the future television standard. To avoid reappearance of the incompatibility of the current multiple TV systems, a single HDTV standard has been sought for years. Two schemes, the Japanese and the European proposals, have now become candidates, but neither of them has yet been adopted as the unique worldwide standard.

The basic parameters of the two systems are slightly different. The Japanese proposal employs an interlaced system of 30 frames (60 fields) per second while the European uses progressive scanning for 25 frames per second.

No compression method is defined by the proposals, but several experimental systems have been developed and demonstrated to provide good subjective quality at transmission rates of 70 to 140 Mb/s [Cominetti 90, Guglielmo 91].

HDTV has also been explored as a possible form of a common cross-industry standard for easy interchange of video signals generated from diverse sources. This is based on the belief that the development of a multi-standard or standard-independent decoder would only cost slightly more than that of a dual-standard receiver, but offer the extra benefit of being able to handle many different types of signals [Slater 91].

In addition to standard-independency, the *Open Architecture HDTV* project emphasizes *receiver scalability*, which demands that signal transmission match receiver's display and processing capabilities. This objective is achieved by applying the pyramid coding technique so that signals are organised in a hierarchical form and only those parts required by the receiver are transmitted. Such an approach is no doubt more efficient for point-to-point communication. In a multipoint communication environment, however, signals may still need to be transmitted in full when receiver requirements vary.

2.3.6 Other Commercial and Research Systems

In addition to the standardisation activities mentioned above, digital video schemes are also studied at commercial and research organisations. Such work is mostly associated with multimedia applications based on high performance computer workstations, digital storage media, and high speed networks, rather than directly involved in telecommunications.

Compact Disc-Interactive (CD-I)

CD-I was initiated by Philips, and later developed jointly by Philips, Sony, and Matsushita. It was intended to provide full motion video on current television screens, and therefore, its uncompressed streams comply with the CCITT Recommendation 601. The MPEG compression algorithm is directly used to reduce the bit rate of video streams to comply with the fixed throughput of a CD player, i.e. 1.2 Mb/s excluding the bandwidth allocated to audio signals.

Because of the high compression ratio (about 140), a stream is first subsampled both horizontally and vertically, each by a factor of two. The MPEG algorithm is then applied on the resulting 360×288 frames.

Although a single VLSI chip can be implemented easily for real time decoding [Sijsterman91a], the encoding process is computationally expensive. For a benchmark of 50 second full motion video, a parallel computer system consisting of 100 Motorola 68020 processors runs at about 2 seconds per frame, and using the VAX 8800 computer the process is 32 times slower [Sijsterman91b].

Digital Video Interactive (DVI)

DVI is a commercial system designed to bring a live video stream from a CD-ROM to a personal computer. A set of proprietary algorithms has been designed to achieve the high compression rate required for the stream, and a special VLSI video signal processor [Harney 91] has been developed by Intel to perform the algorithms in real time.

To shift a digital RGB signal of 512×480 samples (at 8 bits for each component) to and from a CD-ROM device, a compression factor of over 160 is needed. This is achieved by transforming colour representation formats, subsampling, region coding with motion compensation, and variable length coding.

The conversion of the RGB to YIQ formats makes it possible to take advantage of the fact that human visual perception is not as sensitive to chrominance resolution as to luminance. Therefore, only one in every four I or Q samples horizontally and vertically is used. A further stage of subsampling by a factor of two is then taken in both directions. The resulting frames contain 256×240 Y samples, and 64×60 I and Q samples.

The region coding is combined with motion compensation in which a reference in the previous frame is searched for coding the current region. The difference of the two regions are represented in vectors which are then Huffman coded, the most commonly used entropy coding technique.

Since the DVI coding scheme is highly asymmetrical in which encoding is much more difficult than decoding, two types of compressed streams are defined for real time and non-real time encoding. The *Edit Level Video* (ELV) is encoded without using the computationally intensive parts of the algorithm so that the process can be performed on a DVI development system[5] in real time. The cost of this approach is that the resolution and the frame rate have to be further reduced, and the quality is therefore much lower.

The *Presentation Level Video* (PLV), on the other hand, is encoded by the full DVI

[5]A DVI development system consists of a personal computer, such as the IBM PC-XT, with a plug-in DVI board containing a special video processor.

algorithm, which has to be run on a large computer system on a non-real time basis. The compression takes about 3 seconds per frame on a Meiko parallel processor CPU, which has 64 Transputer processing nodes [Luther 91]. The speed is still about 90 times slower than real time, but the quality is the highest that can currently be achieved on a computer screen with such a high compression ratio.

Both ELV and PLV can be decoded in real time by a DVI development system. The current displaying resolution is 256×240 due to the speed of the video processor.

Pandora Video

The Pandora project [Hopper 90] was among the first to experiment with real time video applications in an integrated environment of high speed networks and personal workstations. Although the video format is only greyscale at the current development stage (while the other schemes discussed so far are colour), it has been successfully used in daily work and inter-personal communications [Hopper 91].

The coding algorithm for the Pandora video is relatively simple because communication bandwidth in a high speed network such as the Cambridge Fast Ring [Hopper 86] is regarded as sufficient for the early experimental small video streams. For both *normal* (256×240) and *small* (128×120) video, a process of subsampling by a factor of two in each direction is applied. The resolution-reduced streams are then DPCM encoded at 4 bits per sample. The overall compression is therefore fixed to a factor of 8.

Frame segmentation is used in the Pandora system. Each frame is vertically broken into several blocks which are then encapsulated into *video segments* with a header containing the frame number, segment position, pixel aspect and other information.

It was decided early in the design that a computer workstation should be free from the heavy computation of video processing. This led to the design of a special piece of hardware, the Pandora's box, where all the processing of video streams is handled.

2.4 Observations on Video Coding Schemes

From the above survey, it is found that due to the simplicity and coding efficiency some basic coding types have been widely employed by (draft) international video coding standards. In predictive, transform, and entropy coding, only simple multiplication and accumulation are required. Normally, a factor of 4 can be obtained from predictive coding for bit rate reduction, and 9 from a combination of transform (DCT) and entropy coding. On the other hand, the rest of the coding types either involve complicated codec design

	Rec. 601		H.261		HDTV	
	NTSC	PAL	CIF	QCIF	Japanese	European
Frame rate	30	25	7.5–30 [a]		30	25
Lines/frame	525	625	288	144	1125	1250
Active lines/frame	480	576	288	144	1035	1152
Samples/line	858	864	360	180	2200	2304
Active samples/line	720	720	352	176	1920	1920
Y:Cr:Cb	4:2:2		4:1:1		22:11:11	
Bits/sample	8		8		8	
Aspect ratio	4:3		4:3		16:9	
Bit rate (Mb/s)						
uncompressed	166		36	9	954	885
compressed	30–45 [b]		0.064–2		70–140	

[a]Applications can select one of the four frame rates, 30, 15, 10 and 7.5 frames/sec.

[b]Compression of the CCIR Recommendation 601 video is specified by the CMTT.

Table 2.1: Telecom Digital Video Systems

	MPEG [a]	CD-I	DVI	Pandora	
				Small	Normal
Frame rate	30	25	30	12.5	25
Lines/frame	576	576	240	120	240
Samples/line	720	720	256	128	256
Colour format	Component	Y:Cr:Cb=4:2:2	R:G:B=4:4:4	Greyscale	
Bits/sample	Variable	8	8	8	
Bit rate (Mb/s)					
uncompressed	149 [b]	166	44	1.5	12
compressed	1.86	1.2	1.2	0.2	1.5

[a]Parameters in this column are given for a "core" bit stream defined by the MPEG.

[b]Assume that there are three colour components (Y:Cr:Cb = 4:1:1) in the signal and each of them has 8 bits per sample precision.

Table 2.2: Computer Digital Video Systems

(pyramid encoding/decoding and region encoding) or unbalanced coding procedures (region coding), although some of them may produce a high coding efficiency, such as a 10 to 15 times bit rate reduction from region coding.

Video coding standards and systems are summarised in Tables 2.1 and 2.2. For the use of video streams in a multipoint communications environment, the following points can be observed:

- All of the standards proposed over the years share a small set of practical coding techniques. Some commercial or research systems may use other coding schemes for special purposes but they are far less influential than the international standards.
- Most of the standards are designed to accommodate video signals of varying quality using the same coding algorithm. The interchange of signals which have been coded differently has not been seriously addressed.
- Dynamically changeable receiving quality from a single stream is only possible in the open architecture HDTV system. This is regarded as a *scalable* system which is an important feature for multipoint communications and will be discussed further in the next section.

2.5 Requirements for Intermediate Stream Representation

The observed characteristics are somehow "inherent" in the type of coding schemes discussed so far due to their design objectives. For multipoint video communication applications, a new kind of signal representation is required which organises signals in an intermediate format to facilitate their easy interchange and scalable reception. To illustrate the issue, requirements for such a scheme are discussed in detail as follows:

- **Genericity**

  It is important that the representation be independent of any particular coding algorithm and format while taking into account the requirements of those algorithms it intends to support. A feasible approach is to provide a set of "descriptors" upon which various forms of video streams can be constructed or described.

  The descriptors need to be primitive enough to provide only basic function blocks for the construction of coding algorithms. When new coding methods emerge in the future, it should be possible to accommodate them as long as the basic coding types they use are supported.

Descriptors need not be "orthogonally organised", as argued in [OHRS 90], because the gains on flexibility outweigh bandwidth savings in most applications. An example is some form of coding algorithm identification, which can be included in a stream along with the description of the algorithm. This serves as a quick indicator to a receiver of how the incoming stream is coded.

- **Fidelity**

  Video streams may exist in a particular coded or compressed form through diverse capturing facilities, existing processing hardware, and multiple coding/decoding passes. Although cross-conversions with no quality loss are not always possible because of coding noise [Lippman 91], the scheme should be designed to minimise such loss.

- **Interchangeability**

  A video stream presented in this scheme should be "universally" understandable. Within the target range of coding types, any video stream should be decodable from the general structure defined by the scheme. Decoding may be performed either by software programs for non-real-time applications or hardware modules for real-time processing.

- **Multiplexing Ability**

  Due to the fact that services such as desktop video conferencing and other multipoint communications are important application areas, it is desirable to have multiple streams received and processed simultaneously at a single receiver. This feature is different from the conventional meaning of channel multiplexing in that the multiplexing of streams is done at a destination rather than on a communication path. This implies that the difference between individual blocks of different streams should be presented in the blocks themselves rather than by any lower level communication protocols.

- **Scalability**

  Scalability is a feature embedded in a scheme, wherein processing power and/or storage space required for a stream is based on the demand at a receiving point in order to make effective and economical use of varying quality video processing facilities. The receiver's demand can be either spatial or temporal depending on the requirements of a particular application environment.

  A typical multimedia workstation user may use the full display to get the details of his/her video mail, while in a video conference where multiple streams are displayed at the same time, each stream is restricted to a smaller area of the screen. For the latter, the processing power and memory space of the receiver for each stream should be reduced accordingly to prevent overrun of the receiver.

There are two possible approaches to the design of a scalable system. Firstly, coding steps and resulting signals are strictly organised into a stream so that subsets of the original signal with degrading quality levels can be extracted directly [Bove 91, Lippman 91, Verbiest 88]. Since the extraction is straightforward from special coding algorithms applied to the signal, this approach can also make efficient use of network resources by sending only that part of the signal required by a receiver. The drawback is that the use of other coding types is precluded. Alternatively, by shifting complexity to the receiving side, a scalable scheme is still achievable with the added flexibility of accommodating a variety of coding techniques. Although transmission efficiency may be lost, this is in fact undesirable for applications such as video conferencing because the requirements of the receivers are not necessarily the same and source signals are best transmitted in full.

- **Manipulability**

  Lack of accessible manipulation tools and technologies for an average user is a significant weakness for the video medium today [Liebhold 91]. This is partly due to restrictions of current video representation techniques.

  The basic capabilities of handling video streams include fast forward and reverse scan, slow-speed replay, still operation, and random access, even if a stream is in its compressed form. Above those, there are standard cut-and-paste editing features. The granularity of accessible editing units is an important characteristic for video stream manipulation as well as the maximum delay to any frame in a stream which measures how quickly access operations can be performed. It is also important to ensure that an edited video sequence will not break the time relationship between transmission and display, which is a serious problem for some coding methods [Lippman 91].

  Although the functions are straightforward for uncompressed video streams, they may become extremely complicated for compressed ones, or even conflict with other requirements in some cases. For example, to facilitate rapid access and fine granularity of editing, groups of relatively-coded successive frames must be kept as small as possible, which limits the coding efficiency [Le Gall 91]. A trade-off is therefore inevitable. In this respect, a generic representation scheme accommodating multiple coding types would be beneficial so that applications may select suitable ones for the situation among those available.

- **Efficiency**

  While preserving the maximum fidelity of the original video signal and other features discussed above, a stream in a general representation form should not have excessive bandwidth expansion in comparison to that directly coded by the same algorithm. This implies that the extra descriptive information should be kept to a minimum, and the organisation of streams should be efficient.

- **Simplicity**

  As a general representation scheme is expected to be implemented in hardware as well as software to suit different types of applications, it should be simple enough for its encoders and decoders to be constructed at reasonable cost.

- **Robustness**

  Different levels of robustness are needed for various parts or packets of streams. Few communication channels are error free. In an ATM network, for example, corruption and loss of packets may occur for a number of reasons [De Prycker88]. Due to the high speed nature of video streams, it is widely believed that packet loss is unavoidable but has little impact on a receiver's performance. Such an assumption, however, has to be reconsidered when different types of packets are transmitted in the single general stream.

  For a packet containing only a block of video signals, protection or error recovery is often unnecessary or even impossible because of the real time nature of such signals. But for packets carrying essential information for a whole stream, such as a coding structure, a colour transform matrix, and any spatial or temporal parameters, error-free transmission must be guaranteed since the absence of any of these packets would prevent the whole stream being properly received. There may be other packets which also provide control information, but are either not essential or overridden frequently. These packets may be protected with another level of error recovery, a compromise of the above two extremes.

2.6 Summary

Since digital video signals by nature occupy much wider bandwidth than their analogue counterparts, extensive research has been carried out for years to find an efficient way to code them. This has resulted in a number of signal formats, although none of them are considered to be the best and the most satisfactory. For multipoint video applications, this means that the interchange of different format signals becomes essential.

Despite a large amount of effort to standardise the coding of video signals, none of the emerging schemes are designed for the special requirements of multipoint video communications. Interchanging different signals is often very difficult, and the scalable reception issue is usually omitted. Such situations call for a new intermediate representation scheme for video streams especially for multipoint digital video applications.

Chapter 3

Multipoint Communications

As discussed in the previous chapter, signal interchange can be handled by an intermediate stream representation scheme at the presentation level. However, congestion must be addressed at the network level to provide a complete architecture supporting multipoint digital video communications. This chapter examines a typical multipoint broadband application example to illustrate the problems involved. This is followed by a detailed discussion of conventional multicast path finding algorithms, the reasons for their inadequacy in the new environment, and the requirements for a stream multicast path finding mechanism.

3.1 Internet Multicast — A Case Study of Video Conferencing

Video conferencing is a typical example of multipoint broadband communications, and has developed into two categories:

- *Studio video conferencing* is a traditional service which has been in use in the telecommunications world for years.
- *Desktop video conferencing* is a new type which appears in the high speed local area network (LAN), wide are network (WAN), and internet environments.

Multicast requirements are very different between these two categories, because of differences in conference architecture.

A typical studio video conference system uses a star configuration as shown in Figure 3.1. Participants are gathered in some *Studios* where proper audiovisual facilities are provided by telecommunication authorities. A *multipoint control unit (MCU)* [Clark 90] is located at the hub of the star as the central control and switching equipment. Its basic function is to determine the current speaker according to conferees' voice activity and distribute the audio and video signals to the others. Connections between MCU and studios can be either terrestrial telephone lines or satellite links depending on distance. Since this configuration is a simple tree where no alternative routes can be found between any two end points, there is no need to apply a routing policy.

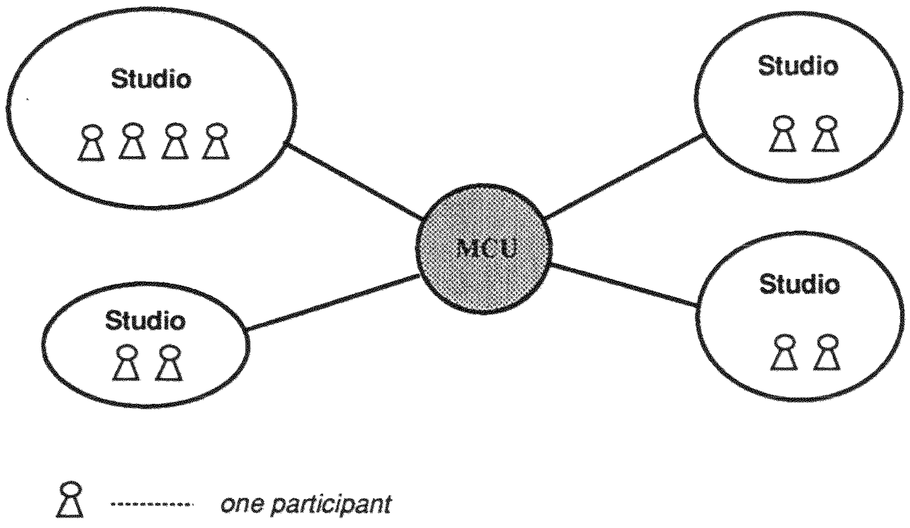


Figure 3.1: A Studio Video Conference

Desktop video conferencing, on the other hand, has been propelled by the increasingly popular multimedia technology and growing communication bandwidth provided by high speed LANs and WANs. Multimedia workstations [Lazar 87, Hopper 90, Hayter 91] have made it possible to have video conferences through every conferee's personal computer rather than particularly equipped conference studios. This situation is illustrated in Figure 3.2. Audiovisual signals from every conferee's workstation go through a set of interconnected high speed LANs to multiple destinations. Unlike the predefined star-shaped structure of studio video conferences, in which routing is unnecessary, this mesh interconnected topology requires routing decisions for the multicast streams.

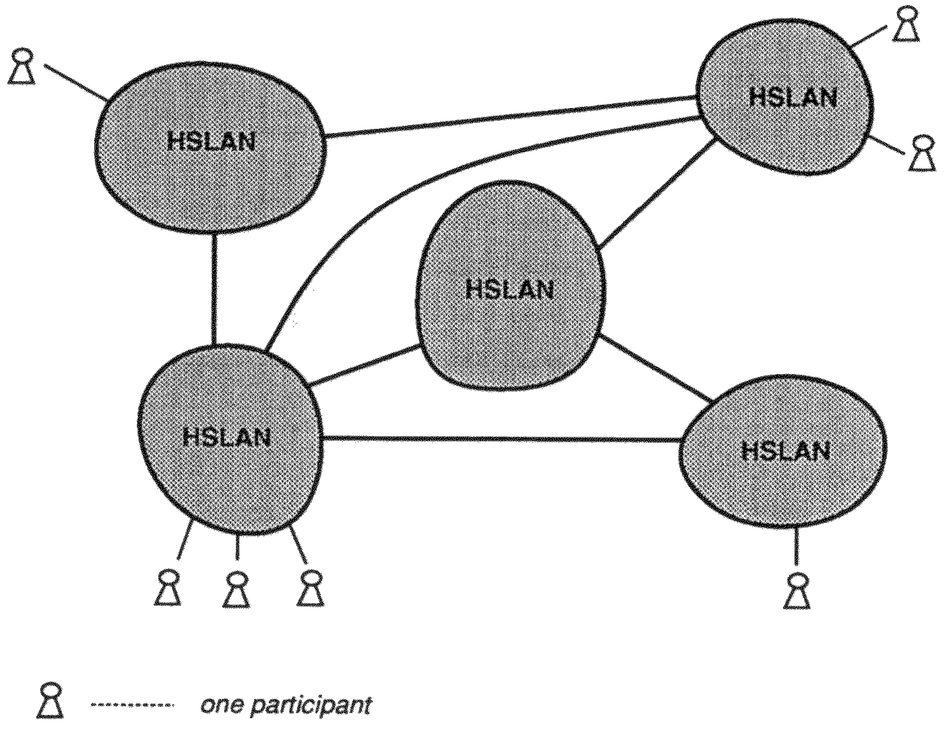


Figure 3.2: A Desktop Video Conference

In [Wall 80] three modes were proposed to describe broadcast groups based on their member's behaviour. Since broadcast is a special case of multicast, such classification is also suited for cataloguing multicast organisations with a slight change in meaning:

- An *autocratic* group allows one member to send while the others can only receive and if necessary respond with acknowledgement. Studio video conferencing belongs to this type because at any particular moment there is only one active speaker in a conference.

- In *democratic* groups any member has the equal right to send to and receive from others. However, non-members are not allowed to participate in the conference in either sending or receiving signals. Desktop video conferences are mainly of this type.

- *Civil service* means that all of the restrictions for the above two types are eliminated. Hence, even non group members can multicast to the group. But applications of this type still remain in data, rather than video, communications.

Due to the fact that desktop video conferencing employs a democratic multicast mechanism, signal streams in this environment have in their nature *temporal continuity* and *broad bandwidth occupancy.* They consume a large portion of bandwidth on network links for a relatively longer period (usually in terms of minutes or even hours), and normally connection-oriented communication channels are preferred to ease receiver buffering of incoming signal packets. Also, every participant's audiovisual signal may be sent to others rather than only the current speaker's, as in an autocratic studio video conference. The bandwidth requirement is proportional to the number of conferees. These new characteristics imply that the traditional multicast routing policy of using a single spanning tree path with little care of communication link capacities, which has been developed for narrow-band single shot data communications, is no longer suitable because paths found by such a mechanism are very vulnerable to congestion from wide-band video streams.

3.2 Steiner Problem in Networks

The mathematical model for multicast path finding is so called the *Steiner problem in networks* (SPN). It is defined as follows:

> For a given undirected graph (network) $G = (V, E, c)$ where V and E are node and edge sets respectively, and c is a cost associated with each edge, and a subset of nodes $X \subseteq V$, find a tree T_H in G such that there is a path between every pair of nodes in X, and the cost of T_H is a minimum.

T_H is usually called a *minimum Steiner tree* for X in G. Nodes in X are referred to as *destination nodes* or *X-nodes* in the context of this thesis. Nodes involved in T_H but not belonging to X are called *Steiner nodes.* A smaller number of Steiner nodes imply fewer extra network resources used.

Since SPN has been proved an NP-complete problem [Karp 72], and remains NP-complete even if G is planar [Garey 79], only heuristic algorithms are of practical interest. These heuristics can be classified as either *centralised* or *distributed* according to the assumptions made. A centralised algorithm assumes the network topology and traffic on each communication link is known to any network node, while a distributed one considers that knowledge of a network node is limited to its immediate neighbourhood.

Two characteristics are considered to be important for heuristic solutions. The first is computational complexity which is the time spent on finding a Steiner tree in the worst case. The second is the worst case error ratio defined as a cost ratio between heuristic and optimal Steiner trees. Both of them will be presented for each approximation Steiner tree search algorithm discussed in the following sections.

3.3 Centralised Solutions

Among a number of centralised approximation algorithms for the SPN proposed in the literature [Winter 87], two basic types of *tree pruning* and *tree merging* can be identified. The former involves a spanning tree (normally a minimum cost one) search on the whole graph in an initial stage, which is then followed by pruning unwanted leaves of the tree progressively until a Steiner tree is created. The latter uses an opposite approach. It starts with all the X-nodes, or single node trees, with no edges to connect them. A function or criterion is then employed to select edges or nodes successively from the graph to make the trees grow and therefore merged each other until a complete (Steiner) tree is formed.

Although the two solutions differ, both types of SPN heuristics exhibit the same worst case performance. Their computational complexity is also similar as the following survey shows.

3.3.1 Tree Pruning Based Heuristics

The algorithm proposed in [Kou 81] represents a typical example of tree pruning based heuristics. It is formulated as follows:

KMB Algorithm:

Find a Steiner tree T_H for an undirected distance graph $G = (V, E, d)$ and a set of nodes $X \subseteq V$.

Step 1: Construct a complete distance graph G_1 from G and X where $cost_{G_1}(u, v)$ is the length of the shortest path between u and v in X.

Step 2: Find a minimal spanning tree T_1 of G_1.

Step 3: Convert the edges in T_1 to paths in G to form a subgraph G_S.

Step 4: Find a minimal spanning tree T_S of G_S.

Step 5: Construct a solution T_H from T_S by deleting its edges so that all the leaves in T_H are X nodes. □

The algorithm is fairly straightforward and was also given independently by [Plesnik 81]. Its overall worst case computational complexity is $O(|X||V|^2)$ and the worst case error ratio is bounded by $2(1-1/l)$, where l is the number of leaves in the optimal tree [Kou 81, Waxman 88b].

Another tree pruning based SPN heuristic was given in [Plesnik 81] and independently in [Sullivan 82], which is, in fact, an improved version of the KMB algorithm. It expends the minimum spanning tree search base to all of the complete distance graphs induced by $Q \cup X$ with $Q \subseteq V|X$ and $|Q| \leq q$, where q is a given number from $[0, |X|-2]$. Among all the T_H's obtained by the KMB algorithm, the one with the minimum cost is selected. The worst case error ratio of this heuristic tends to $2 - q/(l-2)$ which is better than that of the KMB for $q \geq 2$. However, its worst case computational complexity also increases significantly as q becomes larger. It is expressed as $O((|V|-|X|)^q|X|^2 + |V|^3)$. Note that in an extreme case, when $q = |X|-2$, the heuristic becomes a precise SPN algorithm as given in [Lawler 76], which is not solvable in polynomial time even though an optimal Steiner tree is guaranteed.

[Mehlhorn 88] also suggested an improvement in the first step, which is the most time consuming part, of the KMB algorithm to make it faster. It divides the graph G into *Voronoi regions* each of which consists of one X-node x and the set of its neighbouring nodes in V closer to x than to any other X-nodes. The auxiliary graph G_1, upon which a minimum spanning tree search is performed, is then replaced with the one in which a pair of X-nodes are connected only if an edge between their Voronoi regions exists in E. This reduces the number of edges in G_1 to $O(|E|)$ so that the minimum spanning tree search can be carried out in $O(|X|log|X| + |E|)$ time. Since the partition of Voronoi regions can be computed in $O(|V|log|V| + |E|)$ time [Mehlhorn 88], the overall computational complexity is also reduced to this order.

3.3.2 Tree Merging Based Heuristics

The approximation algorithm given in [Rayward 83, Rayward 86] illustrates how the tree merging based heuristics find Steiner trees on graphs.

RS Algorithm:

Find a Steiner tree T_H for an undirected distance graph $G = (V, E, d)$ and a set of nodes $X \subseteq V$.

Step 1: Construct a set of subgraphs $T = \{(\{x\}, \emptyset) : x \in X\}$. Find a shortest path between any pair of nodes u, v in G as $R(u, v)$ and let $d(u, v)$ be the length of the path.

Step 2: For each $v \in V$ evaluate its proximity function

$$f(v) = \min_{1 \leq r \leq k} \left\{ \sum_{i=0}^{r} d(n, t_i)/r : t_0, t_1, ..., t_r \in T \right\}.$$

Find node w with minimum f-value.

Step 3: Let $t_{w.0}$ and $t_{w.1}$ be the two closest trees in T to w. Join $t_{w.0}$ and $t_{w.1}$ by a shortest path through w to form a new tree $t' = t_{w.0} \cup R(w, t_{w.0}) \cup R(w, t_{w.1}) \cup t_{w.1}$.

Step 4: Delete $t_{w.0}$ and $t_{w.1}$ from T and insert t'.

Step 5: If $|T| = 1$ then $T_H = T$. Otherwise, for each $v \in G$ evaluate $d(v, t')$ and $R(v, t')$ and go back to step 2. □

The function $f(v)$ provides a measure of the shortest average distance through v from a tree $(t_{v.0})$ closest to v to subsets $\{t_{v.1}\}$, $\{t_{v.1}, t_{v.2}\}$, ..., $\{t_{v.1}, t_{v.2}, ..., t_{v.k}\}$. This can be intuitively taken as a reasonable measure of proximity of node v to the trees $t_{v.0}, t_{v.1}, ..., t_{v.k}$. Consequently, each iteration of the algorithm joins a pair of closest trees in the tree list T and after $|X| - 1$ iterations, T becomes a single tree spanning all X-nodes.

The worst case computational complexity of the heuristic is $O(|V|^3)$, and its worst case error ratio is the same as that of the KMB algorithm. However, it has been shown that for very sparse graphs, the RS algorithm tends to generate more accurate results than the KMB on average [Rayward 84].

Although not directly related to the SPN, it is worth mentioning that the RS algorithm has an extra feature of yielding optimal solutions in two "boundary" cases. The first is that when $X = V$, it reduces to a well known minimum spanning tree algorithm [Kruskal 56]. The second is that when $|X| = 2$, it finds the minimum cost path between the two X-nodes.

There are several improved versions of the KMB algorithm which should be classified into the second type of SPN heuristics, because the original minimum spanning tree building procedure has been broken and resolved into Steiner tree growing processes. The algorithm given by [Wu 86] and later further improved by [Widmayer 86] is one such example. It is similar to the RS algorithm in the first step in that a tree set T is constructed. A priority queue is employed to hold all the nodes in the increasing order of their shortest distance to a subtree in T. Operations are then performed repeatedly on the queue to find and merge any trees of T being brought together at certain node until $|T| = 1$. This algorithm requires $O(|E|log|V|)$ time which is much better than the KMB algorithm when the graph is not very dense and X-nodes only represent a small portion of all nodes. The worst case error ratio, however, remains the same as that of the KMB algorithm.

Another tree merging base SPN approximation algorithm was suggested by [Takahashi 80]. It starts with a randomly chosen X-node as the initial tree. Then, one of the rest X-nodes which is closest to the tree is added together with the minimum cost path from the node to the tree. The process repeats until all the X-nodes are connected to the tree. The algorithm is very simple and intuitive. Both its worst case error ratio and computational complexity are the same as those of the KMB algorithm. But akin to the RS algorithm, its solutions on very sparse graphs are more accurate than the KMB's [Rayward 84].

The approximation algorithm proposed by [Plesnik 81] involves a different process of tree growing. It is based on a more general idea of the following graph contraction property:

> If a graph G contains an X-node u, and the shortest edge incident to it has another X-node v as the other endpoint, then the edge (u, v) belongs to a Steiner minimum tree, and it suffices to consider the SPN arising after the contraction of G along (u, v).

The actual contraction employed in the algorithm is performed recursively on derived subgraphs of G, upon which the previous SPN heuristic is used to find a Steiner tree. The recursion continues until the subgraphs merge to a single one which covers the whole graph, and the Steiner tree on it becomes the final solution. The detailed procedure, which can be found in [Plesnik 81], is rather complicated with many auxiliary definitions, and hence not included here.

Since the edges absorbed at each step of contraction are in fact Steiner tree branches to connect X-nodes, or the initial subtrees, this algorithm is still classified as a tree merging based SPN heuristic. Both its worst case error ratio and computational complexity are the same as those of the RS algorithm, i.e. $2(1 - 1/l)$ and $O(|V|^3)$, respectively.

3.4 Distributed Solutions

Few distributed SPN algorithms, or multicast routing algorithms, have been proposed in the literature. They were mainly developed from an internet broadcast algorithm [Dalal 78] which has been widely used in distance-vector routing [Ford 62, Bellman 57] environments, such as the Xerox PUP Internet [Boggs 80] and some DARPA Internet core gateways [Hinden 82]. Each node in this environment knows the shortest distances to any other nodes and the immediate neighbours on the corresponding shortest paths from periodical exchange of routing information between neighbouring nodes.

The broadcast algorithm, referred to as *reverse path forwarding*, is based on the fact that a shortest path between a pair of nodes is true in either direction. Hence, while a source node broadcasts a packet to all its incident edges, any other node re-broadcasts the packet (except to the edge from which the packet arrives) if and only if it is received from the node on the shortest path to the source node, i.e. from a path in the minimum spanning tree.

Several modified versions of the broadcast algorithm were suggested by [Deering 90] to limit the packet travelling region to a Steiner tree for multicast applications. The best among them is called *reverse path multicasting* which requires non-X-nodes on leaves of the spanning tree to send non-membership reports to their parent nodes of the tree. The spanning tree is continuously pruned while such reports are generated until a Steiner tree is formed. This algorithm has also been proposed as an internet protocol for routing multicast datagrams [Waitzman 88].

Although approaching in a distributed way to multicast path finding, the approximation algorithm is rather similar to the KMB heuristic in that a minimum spanning (broadcast) tree is first constructed and non-X-node leaves are then pruned. Therefore, its worst case error ratio can also be expected as $2(1 - 1/l)$. The worst case computational complexity, on the other hand, cannot be compared with those of the centralised heuristics because the most time consuming part of the algorithm, the broadcast or minimum spanning tree search, is prepared beforehand.

[Wall 82] proposed a distributed version of the KMB algorithm. It suggests that Steps 2 and 4 of the KMB algorithm may be implemented by the above broadcast algorithm. Step 3 can be absorbed into Step 2 since its only purpose is to inform the Steiner nodes of their existence on a Steiner tree. Step 5 can also be eliminated by a consistent tie-breaking rule given together with the new algorithm. However, it assumes that the distance graph constructed in Step 1 is already available in the network, and therefore this dominant complex step is not distributed.

3.5 Problems with Conventional Algorithms

For data communication oriented (narrow-band and casual) multicast applications, the above surveyed path finding heuristics provide rather satisfactory solutions. Although the worst case error ratios of most algorithms have been proved as $2(1 - 1/l)$, experiments have shown that their average error ratios are far below that level, i.e. near optimal. This accuracy issue has in fact been considered to be of prime importance [Winter 87] in such application environments.

However, for the multipoint digital video communication environment, these Steiner algorithms exhibit certain problems due to the lack of integrated consideration of communication bandwidth throughout their tree search processes. These can be clearly illustrated with a simple network example shown in Figure 3.3(a).

Suppose there is a multicast group member at each node of v_1 to v_4 in the graph or network, i.e. all the nodes are involved in the multicast group. The number beside each edge or link indicates its capacity in terms of the number of multicast streams it is able to accommodate, supposing that each stream requires the same amount of bandwidth. It is further assumed that the distances between the nodes are exactly depicted in Figure 3.3(a), i.e. $d(v_1, v_2) < d(v_2, v_3) < d(v_3, v_4) < d(v_4, v_1)$.

Both the tree pruning based centralised heuristics and the distributed algorithms will find a solution as shown in Figure 3.3(b) because this is the minimum spanning tree in the graph. The tree merging based centralised heuristics will start with edge (v_1, v_2), and then add (v_2, v_3) and (v_3, v_4) subsequently, resulting the same Steiner tree. Note that since the capacity of the tree is only 2, the four multicast streams will definitely result in congestion on the tree path. Furthermore, when the tree path has been reserved (or taken out of the graph), the remaining network is as that shown in Figure 3.3(c) which provides no alternative routes for further tree searching. This is a serious drawback because the available bandwidth in the network is not effectively used.

In fact, the network does provide two trees as shown in Figure 3.3(d) and (e) which jointly are able to support four multicast streams from the group. Thus, a well designed Steiner tree search algorithm for multipoint digital video communications should not fail in this situation.

3.6 Requirements for Stream Multicast

Based on the above observation, a new multicast mechanism, referred to as *stream multicast*, is required for applications of multipoint broadband communications. A path finding algorithm for this type of multicast has the following distinctive characteristics:

- Tree capacity, or available communication bandwidth, should be included as an objective of the algorithm so that its solutions are able to accommodate required traffic volume, and the network will not be congested.

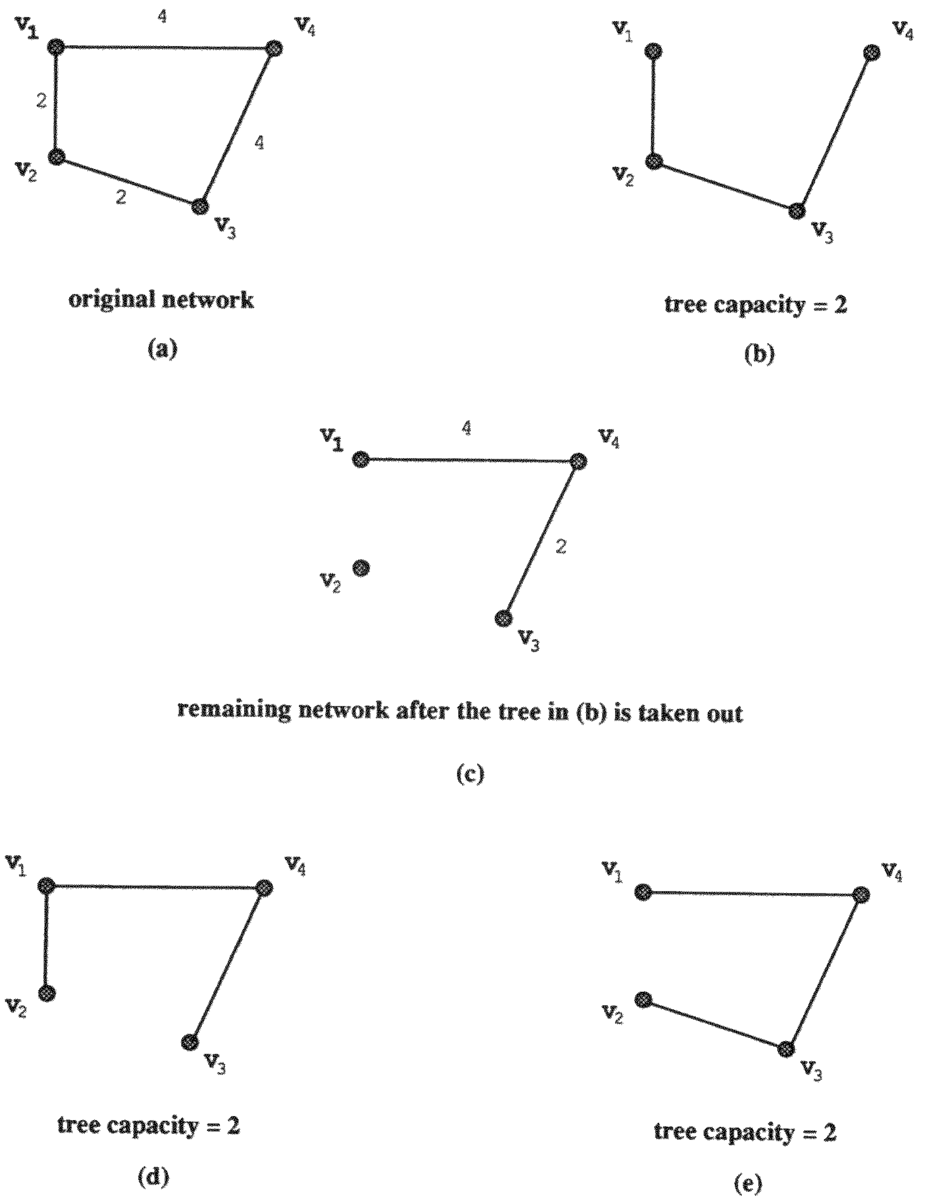


Figure 3.3: A Network Example

- It is permissible that a heuristic algorithm fails to find proper multicast paths in some situations where such paths do exist. But such algorithm failures should be kept to a reasonably low percentage although it is difficult to define a precise level of this ratio which may depend on types of application environments.
- It is desirable that multicast paths found by the algorithm have least effect on the network concerned. This not only requires that the extra network nodes, i.e. Steiner nodes, included in the paths should be as few as possible, but also that the network still be able to function in balance to a maximum degree. The latter implies that the algorithm should try to involve those edges with plenty of unused communication capacity in preference to those small capacity links.
- As an algorithm designed for practical use, it should certainly be executable within polynomial time. In addition, its computational complexity level should be comparable to the above "near optimal" Steiner tree search heuristics.

3.7 Summary

Desktop video conferencing presents problems associated with all broadband multipoint communications. Since digital video streams occupy a great amount of bandwidth, exhibit temporal continuity, and multiple streams are generated within a single multicast group (democratic), congestion becomes a serious issue when considering this type of group communication.

Various approximation algorithms for solving the Steiner problem in networks, which is the mathematical model of multicast path finding, have provided good solutions to data communication oriented multicast applications. However, in the new environment of multipoint digital video communication, these algorithms present a few problems. A stream multicast path finding mechanism is therefore proposed for the new applications. It should take the communication bandwidth into careful account, keep the failure rate low, and restrict the effect of multicast traffic on the network.

Chapter 4

Centralised Stream Multicast Path Finding

This chapter investigates centralised path finding mechanisms for stream multicast. The new algorithms are based on the modifications of well-known conventional Steiner tree search heuristics discussed in the previous chapter. Various ways of integrating the communication bandwidth of network links are examined in order to meet the requirements for such application environments. Verification and comparison of the algorithms to the traditional heuristics are presented with the simulation results.

4.1 Approaches to Stream Multicast

From the target parameter point of view, a stream multicast path finding algorithm differs from a conventional one only in that it has an extra requirement for its solutions, i.e. the communication bandwidth of a multicast path, or *tree capacity*, should meet the intended traffic to be generated from the multicast group. Also, the fact that there are many near optimal centralised SPN heuristics suggests that it may be possible to design new algorithms based on these heuristics by imposing relevant restrictions on their solutions.

Three possible approaches to achieve this objective are discussed in the following subsections.

4.1.1 Single Path Finding

The simplest scheme is to apply original algorithms directly on networks which have been pre-processed. The preparation stage consists of removing all of the network links which are incapable of accommodating the total volume of the intended traffic. This guarantees that whenever a Steiner tree is found, the tree capacity is always enough for the application.

Since this approach lets all signal sources of a multicast group share a single tree, there are expected to be fewer Steiner nodes in its solutions than in those of the other two approaches, and a relatively small proportion of network links are likely to be affected by the multicast traffic. In addition, the computational complexity remains the same as those of the original algorithms, or may be even lower if many of the network links are removed at the preparation stage.

A significant disadvantage is that the failure rate is likely to be high. If the bandwidth of the average video stream is comparable to the capacities of network links, as in the present situation of networks and digital video signals, the preparation stage of removing incapable links leaves few choices for an algorithm to search a tree. This results in frequent failures of path finding, especially for the cases of large multicast groups.

4.1.2 Multiple Path Finding with Link Capacity Check

To reduce the high failure probability of the above approach, it is desirable that those network links, which have less capacity than that required by the total traffic from a multicast group but are still able to accommodate at least one video stream, should not be excluded from the tree search process. Although a resulting path may not have the total bandwidth required, it should support at least one signal stream from the group. If this process repeats on the network where previously found paths are removed, a set of Steiner trees may be obtained the accumulated capacity of which meets the bandwidth requirement.

While the failure rate has been reduced, the drawbacks of this scheme are obvious. The numbers of both Steiner nodes and edges involved are expected to be much larger than those of the previous approach because multiple paths or trees are used for a multicast group. Also, since there is no distinction between large and small capacity links in the network during the process of tree searching, it is impossible to control the selection of links. Thus, the network may be seriously affected due to the excessive use of small capacity links by the multicast traffic.

Note that in the worst case when each path found can only carry one signal stream, $|X|$

(the number of destination nodes) trees are needed. This results in the computational complexity of this approach, although still polynomial, being $|X|$ times higher than that of the previous one, or the original SPN heuristics.

4.1.3 Multiple Path Finding with Link Capacity Integration

The previous approach can reduce the stream multicast path finding failure to a certain extent, but has problems of involving too many nodes and links of the network. An intuitive solution is to consider link capacity as an integrated criterion for node or edge selection throughout the tree searching process [Jiang 91].

When high capacity links are chosen in preference to low capacity ones, the effect is two fold. Firstly, the number of paths is going to decrease because the capacities of trees found tend to be higher and hence the total bandwidth accumulates more quickly. This should reduce the numbers of both extra nodes and links at the same time. Secondly, algorithms should fail less often because when a small portion of bandwidth is removed from a link with a high capacity, that link is still available for the next round of tree searching, and therefore, there is a greater chance of finding a new path. In addition, the network should function in a more balanced way as the multicast traffic has been concentrated on high capacity links.

The computational complexity of this method should be the same as that of the previous one because the same number of paths need to be found in the worst case as long as the integration of link capacity does not introduce more time-consuming operations. However, its average execution time is expected to be shorter since the average number of paths decreases.

In fact, this third approach is a compromise between the first two, having a lower algorithm failure rate than the first, and reducing the effect of multicast traffic on the network (i.e. the extra nodes and links involved) compared to the second. In most cases, it is preferable to the other two for these reasons; however, the previous approaches may still be applied in special situations. For example, the first can be used in a bandwidth abundant network.

Multi-path approaches have also been adopted in the Internet Stream Protocol [CIP 90] which intends to support both point-to-point and multipoint voice and video communications. However, the protocol is used complementarily to the Internet Protocol [Postel 81], and only ensures the correct setup of paths with required characteristics. Routing or path finding is regarded as a separate issue, which is left to a higher level application and is thus not specified.

4.2 Link Capacity-Related Path Finding Algorithms

The survey of centralised multicast path finding algorithms in the previous chapter shows that the KMB and RS algorithms are typical examples of the two types of centralised heuristics, respectively. Most of the other algorithms either have a similar approach or are based on modifications of them. Therefore, these two algorithms are chosen to show how each type of heuristic can be generalised with link capacity integration for use in stream multicast path finding.

4.2.1 Tree Pruning Based

For tree pruning based heuristics, the minimum spanning tree search, together with the auxiliary complete graph upon which the search is performed, are the most important steps of the algorithms. Once a spanning tree is built, there are few choices left for the remaining branch pruning steps. In the case of the KMB algorithm, this consideration is translated into two adjustments:

- Firstly, according to the basic principle that high capacity links must have priority to be searched for before others, a classification is made of network links with respect to their capacities relative to the bandwidth requirements of a multicast group. An auxiliary graph is built by progressively employing classes of links from high to low capacities until it is possible to construct a spanning tree covering all of the $|X|$ nodes on it.
- Secondly, a maximum capacity, instead of minimum cost, spanning tree is searched for on the complete network derived from the above auxiliary graph.

Since each tree searching process concentrates on the highest possible capacity links, either a single tree may support the entire traffic from a multicast group, or these links may remain available for subsequent searches. This ensures both that the algorithm failure rate stays low and that the network is not affected significantly by the multicast traffic.

A drawback of these changes is that some high capacity but long edges may be chosen so that the tree cost increases, if link or tree cost is associated with its length.

By integrating the above adjustments, the modified KMB algorithm proceeds in detail as follows.

Modified KMB Algorithm:

Find a set of Steiner trees, $T_{H_1}, T_{H_2}, \ldots$ with a total capacity of $\sum_{i=1}^{n} B_i$, for an undirected distance graph $G = (V, E, c)$ and a set of nodes $X \subseteq V$, where B_i $(i = 1 \ldots n)$ are the required bandwidth of multicast streams.

Step 1: Build a list of n bandwidth parameters in a descending order, i.e., $B_1 \geq B_2 \geq \ldots \geq B_n$, which will be referred to as a *B-list* in the following description.

Step 2: Classify each edge according to its capacity c into the following groups:

$$c \geq \sum_{i=1}^{n} B_i,$$
$$\sum_{i=1}^{n-1} B_i \leq c < \sum_{i=1}^{n} B_i,$$
$$\vdots$$
$$B_1 \leq c < B_1 + B_2,$$

and discard edges of $c < B_1$. Create a new graph by adding these groups of edges one by one in the above order until all nodes in X are contained in a connected subgraph $G_1 = (V_1, E_1, c_1), V_1 \supseteq X$. Record the value of m of the last group added, $\sum_{i=1}^{m-1} B_i \leq c < \sum_{i=1}^{m} B_i$. If all groups are added and G_1 has not been found, report that no capable Steiner trees can be found and terminate.

Step 3: Remove all the leaves in G_1 which are not in X, and find a Steiner tree T_s for X over G_1 by following similar procedures to the KMB algorithm.

Construct a complete graph $G_2 = (X, E_2, c_2)$ from G_1 of which any edge $e(v, w) \in E_2$ is the path with minimal hops between v and w in G_1. The capacity of each edge is chosen to be the minimum capacity along the corresponding path.

Find a maximum capacity spanning tree T_2 of G_2.

Convert the edges in T_2 to paths in G_1 to form a subgraph G_3.

Find a maximum capacity spanning tree T_3 of G_3. The Steiner tree T_s is obtained by removing all the leaves of T_3 which are not in X.

Step 4: Let c_s be the minimum capacity of all branches in T_s and $c_1 = c_s - \sum_{i=1}^{m-1} B_i$. Delete $B_1, B_2, \ldots B_{m-1}$ from the B-list. For B_i $(i = m \ldots n)$, repeat the following procedure.

If $c_1 \geq B_i$, set c_1 to $c_1 - B_i$, and delete B_i from the B-list.

Let $c_t = c_s - c_1$, and subtract c_t from c of each edge of G in T_s. If the B-list is not empty, reset n to the number of the remaining components on the list, and go back to Step 1. □

4.2.2 Tree Merging Based

In tree merging based approximation algorithms a certain criterion or proximity function is employed to determine edges or nodes to be included in the Steiner tree. It is normally the most critical part of an algorithm, and therefore where the link capacity concerns should be incorporated. There are two possible ways of implementing this integration each with different effects on the resultant trees:

- Replace the cost function of links in the criterion, usually expressed in their distances, with its value after division by the average bandwidth of the required signal streams. If the original cost function is link distances, the new one can be referred to as *link capacity indices.* It reduces the "cost" of a link in proportion to the average number of signal streams which it is able to accommodate. Such links are therefore given priority over those which can carry fewer signal streams when constructing a Steiner tree.
- Alternatively, link costs can be assigned solely as a function of bandwidth, such as the inverse of the average bandwidth of the required signal streams. This is an extreme case where bandwidth is given the full priority in the process of link or node selection for the tree construction. It may be particularly useful in situations where distance is not a significant factor to determine link costs, either because the coverage of the network concerned is limited, or the switching is much more expensive than the transmission.

The above modifications to the link selection criterion ensure that the Steiner tree searching concentrates on high capacity links first so that the link classification process used in the tree pruning based heuristics becomes unnecessary in this case. Both the positive and negative effects of these changes are similar to those made to the tree pruning based heuristics. The relative significance of these effects is to be further investigated by simulation.

Since they can be used in different situations, both of the above two methods of integrating link capacities to tree merging based heuristics are applied to the RS algorithm as two different versions, details of which are given as follows.

Modified RS Algorithm:

Find a set of Steiner trees, $T_{H_1}, T_{H_2}, \ldots$ with a total capacity of $\sum_{i=1}^{n} B_i$, for an undirected distance graph $G = (V, E, c)$ and a set of nodes $X \subseteq V$, where B_i $(i = 1 \ldots n)$ are the required bandwidth of multicast streams.

Step 1: Build a B-list of n bandwidth parameters as $B_1 \geq B_2 \geq \ldots \geq B_n$.

Step 2: Construct a subgraph $G_1 = (V, E_1, c_1)$ where E_1 includes all the edges with $c \geq B_n$. If all the nodes in X are not in a connected graph, report that no capable Steiner trees can be found and terminate.

Remove isolated subgraphs in G_1 except the one containing X nodes, and all the leaves which are not in X. Assign a cost to each edge in G_1 as follows:

$$cost = \frac{distance}{\lceil nc / \sum_{i=1}^{n} B_i \rceil} \qquad (version\ 1),$$

or

$$cost = \frac{1}{\lceil nc / \sum_{i=1}^{n} B_i \rceil} \qquad (version\ 2).$$

Step 3: Find a Steiner tree T_s for X over G_1 by the original RS algorithm where $d(u, v)$ is replaced by $cost(u, v)$.

Step 4: Let c_s be the minimum capacity of all branches in T_s and $c_1 = c_s$. For B_i $(i = 1 \ldots n)$, repeat the following procedure.

If $c_1 \geq B_i$, set c_1 to $c_1 - B_i$, and delete B_i from the B-list.

Let $c_t = c_s - c_1$, and subtract c_t from c of each edge of G in T_s. If the B-list is not empty, reset n to the number of the remaining components on the list, and go back to Step 1. □

Before the simulation gives a quantitative comparison of the new and the original heuristics, it is worth applying the new algorithms on the simple network example described in the previous chapter to check intuitively the effectiveness of the changes.

The modified KMB algorithm first finds a maximum capacity spanning tree as shown in Figure 3.3(d) which is also a Steiner tree having a capacity of 2. It then obtains the second tree as shown in Figure 3.3(e) providing another capacity of 2 to satisfy the total bandwidth requirement of the multicast group.

On the other hand, both versions of the modified RS algorithm start with the edge (v_3, v_4), which is then joined by edges (v_1, v_4) and (v_1, v_2) successively, resulting in the same first

Steiner tree as that found by the modified KMB algorithm. The second Steiner tree can also be checked to be the same so that the modified RS algorithm succeeds in this case as well.

4.3 Computational Complexity

In the modified KMB algorithm, Step 1 is a simple sorting operation which takes no more than $O(m \log_2 m)$ time, where m is the number of the elements in the B-list, using the Heap-sort algorithm [Aho 83]. Step 2 involves classifying all the edges of a graph and building a connected subgraph. The former part obviously takes $O(|E|)$ time while the latter needs $O(n|V|^3)$ because a connectivity check takes $O(|V|^3)$ time using Floyd's algorithm [Floyd 62] to find shortest paths between pairs of nodes, and all n groups of edges have to be used to construct a subgraph in the worst case. Step 3 has only slight changes from the original KMB algorithm, and has the same complexity of $O(|X||V|^2)$ [Kou 81]. The capacity calculation and B-list operation in Step 4 take $O(|V|)$ and $O(n)$ time, respectively.

In the worst case, a set of n Steiner trees, each of which has a capacity of B_i $(i = 1...n)$, has to be found, and the number of components in a B-list is at least one less than its value in the previous round. However, the sorting operation on the B-list is only required at the first iteration of Step 1, when $m = n$. The whole algorithm, therefore, needs $O(n \log_2 n + n^2|V|^3 + n|X||V|^2 + n|E| + n^2)$ computation time. Since $n \geq |X|$, $n|V| \gg |X|$, and $n \log_2 n < n^2$, the above expression can be simplified as $O(n^2|V|^3 + n|E|)$. Also, for a given number of vertices, the maximum possible number of edges found in a complete graph is

$$|E| = \sum_{i=1}^{|V|-1} i = \frac{|V|^2 - |V|}{2},$$

and therefore,

$$n|E| < n|V|^2 < n^2|V|^3.$$

Hence, the complexity of the algorithm is no worse than $O(n^2|V|^3)$.

This is a reasonable value to be expected because it is one degree higher than that of the original algorithm, $O(|X||V|^2)$, with respect to n or $|X|$. The extra factor of $|V|$ is introduced by the connectivity check which is never required for conventional Steiner tree algorithms.

In the modified RS algorithm, the sorting operation in Step 1 again takes $O(m \log_2 m)$ time. Both the subgraph construction and cost calculation parts of Step 2 take $O(|E|+n)$ time while connectivity check needs $O(|V|^3)$ time. Step 3 is basically the same as the original algorithm which takes $O(|V|^3)$ time [Rayward 83], and Step 4 takes $O(|V|+n)$ as analysed for the modified KMB algorithm above. Therefore, the computation time required by the whole algorithm is not longer than $O(n \log_2 n + n|E| + n^2 + n|V|^3)$ in the worst case. Since $n|E| < n|V|^2 < n|V|^3$ and $n \log_2 n < n^2$, the expression can be simplified as $O(n|V|^3 + n^2)$, or $O(n|V|^3)$ since normally $n \ll |V|^3$.

Compared to $O(|V|^3)$ required by the original algorithm, the computational complexity of the modified RS algorithm is also one degree higher with respect to n. Such a result is in fact the best obtainable because of the multiple paths to be searched.

The above analysis suggests that both of the modified algorithms can be completed within a reasonable order of polynomial computation time which meets the last requirement for a good heuristic algorithm for stream multicast path finding.

4.4 Simulation

4.4.1 Network Model

The basic network model is taken from [Waxman 88a] which has some of the characteristics of an actual network. A few changes are needed to introduce capacities and asynchronous traffic to links.

All the network nodes are randomly distributed over an area with a rectangular coordinate grid. The required destination nodes are chosen randomly. It is possible for some of the nodes to be selected more than once because each node represents a subnetwork (HSLAN in Figure 3.2) where a few participants may be connected. This corresponds to the case of $n > |X|$. (However, in the presentation of the simulation results, the *number of destination nodes* is the value of n, rather than $|X|$, to provide a uniform illustration.)

Edges or links are determined by a probability function,

$$P(\{u,v\}) = \beta exp \frac{-d(u,v)}{L\alpha}$$

where $d(u,v)$ is the distance between nodes u and v, L is the maximum distance between two nodes, and parameters $\alpha, \beta \in (0,1]$. An edge $e(u,v)$ exists if and only if a random value generated for it is greater than or equal to $P(\{u,v\})$. By giving different values to α or β, the ratio of short to long edges or the total edge density in a network can be adjusted.

One of the three capacity values, 10, 30, or 100 Mb/s, is assigned to each link randomly, but the percentage of each category is maintained as 60, 30, and 10, respectively. A normally distributed asynchronous traffic load with an average of a quarter of the link bandwidth is then subtracted from the capacity assigned above, and the final remaining value is taken as the current available bandwidth of that link.

4.4.2 Results

Based on the above model, 2500 different 25-node networks were generated randomly with the number of destination nodes as 4, 6, 8, 10 and 12 (500 networks for each). The required bandwidth of each multicast stream was uniformly assumed to be 3 Mb/s to simplify the simulation model while maintaining the generality of the results. Also, such a bandwidth is sufficiently high for most of the current experimental video streams, such as a 1.5 Mb/s Pandora video, a 1.86 Mb/s MPEG core bit stream, or a 2 Mb/s high-end H.261 video stream, as surveyed in Chapter 2.

The simulation was carried out for all of the three approaches to the stream multicast path finding, as discussed in the beginning of this chapter, to compare their relative performance. On all of the 2500 sample networks, each approach is used to find a suitable set of Steiner trees. The results are illustrated in Figures 4.1 – 4.5.

A few definitions are required before examining the results:

- A *normalised link cost* is the ratio of the link length to the average length of all links in a network.
- A *normalised multicast cost* is the accumulation over the tree set of the tree costs weighted by their capacity indices, i.e. the average numbers of streams the trees can carry.

It is shown in Figure 4.1 that the costs of trees found by the modified algorithms (the multi-path approach with link capacity integration) are slightly (about twice at most) higher than those found by the repeated-run versions of the original algorithms (the multi-path approach with link capacity check), but comparable with the single-run versions (the single path approach). This means that the shift of emphasis from link distance to link capacity increases the cost of multicast trees, but not significantly. In addition, the greater this shift, the more the cost increases, as demonstrated by the two versions of the modified RS algorithm.

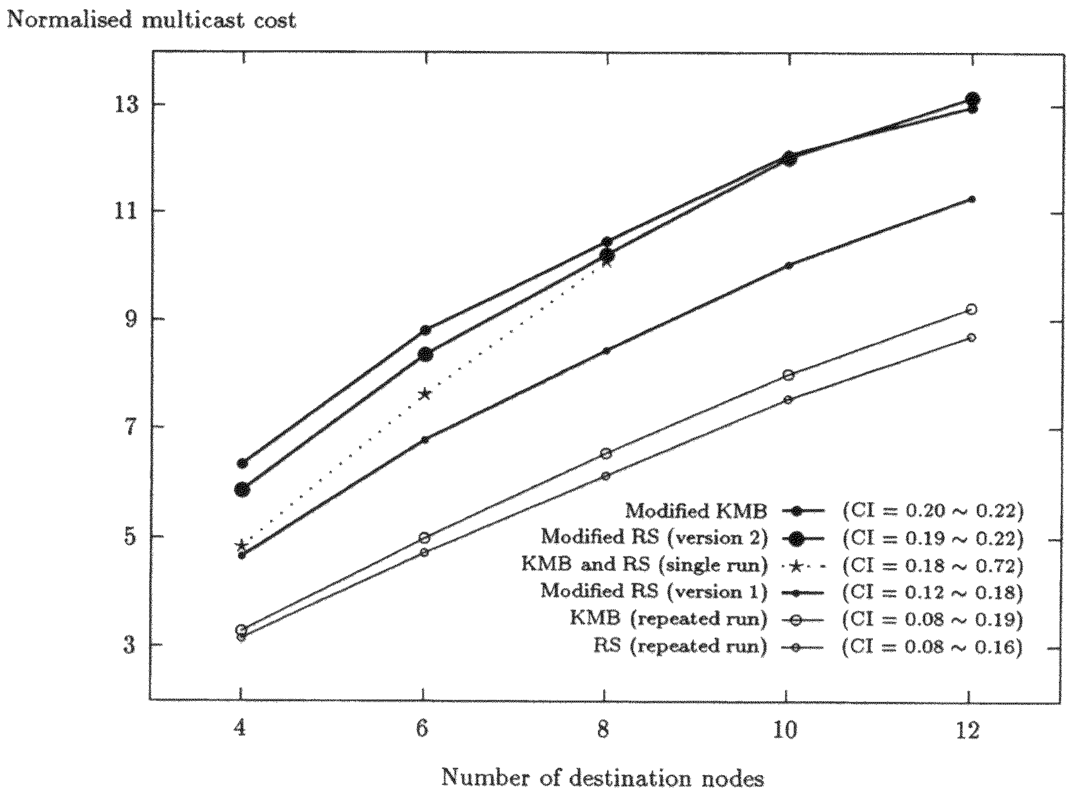


Figure 4.1: Normalised Cost of Multicast Trees

Number of destination nodes	95% Confidence intervals				
	4	6	8	10	12
Modified KMB	0.20	0.21	0.21	0.21	0.22
Modified RS (version 1)	0.12	0.14	0.15	0.16	0.18
Modified RS (version 2)	0.19	0.19	0.19	0.20	0.22
KMB (repeated run)	0.08	0.11	0.12	0.15	0.19
RS (repeated run)	0.08	0.09	0.10	0.13	0.16
KMB (single run)	0.19	0.27	0.71	–	–
RS (single run)	0.18	0.27	0.72	–	–

Table 4.1: 95% Confidence Intervals of Normalised Multicast Costs

The accuracy of a simulation result is usually measured by its *95% confidence interval* (CI), which gives the error range that includes 95% of random samples. Under the assumption that errors have a normal distribution, it can be calculated as follows:

$$CI = \frac{1.96}{\sqrt{m-1}}\sqrt{\frac{\sum_{i=1}^{m}(x_i - \bar{x})^2}{m}}$$

where x_i is a random sample, m is the number of samples, and $\bar{x}$ is the mean value.

Table 4.1 gives the 95% confidence interval for each simulation result presented in Figure 4.1. The values are mostly less than 3% of the corresponding simulation results except in one case where the values of the single run KMB and RS algorithms reach 7% due to a reduced number of successful samples. This suggests that the results are reliable. For the rest of the figures, similar tables are omitted, but the range of the confidence intervals for each curve is indicated.

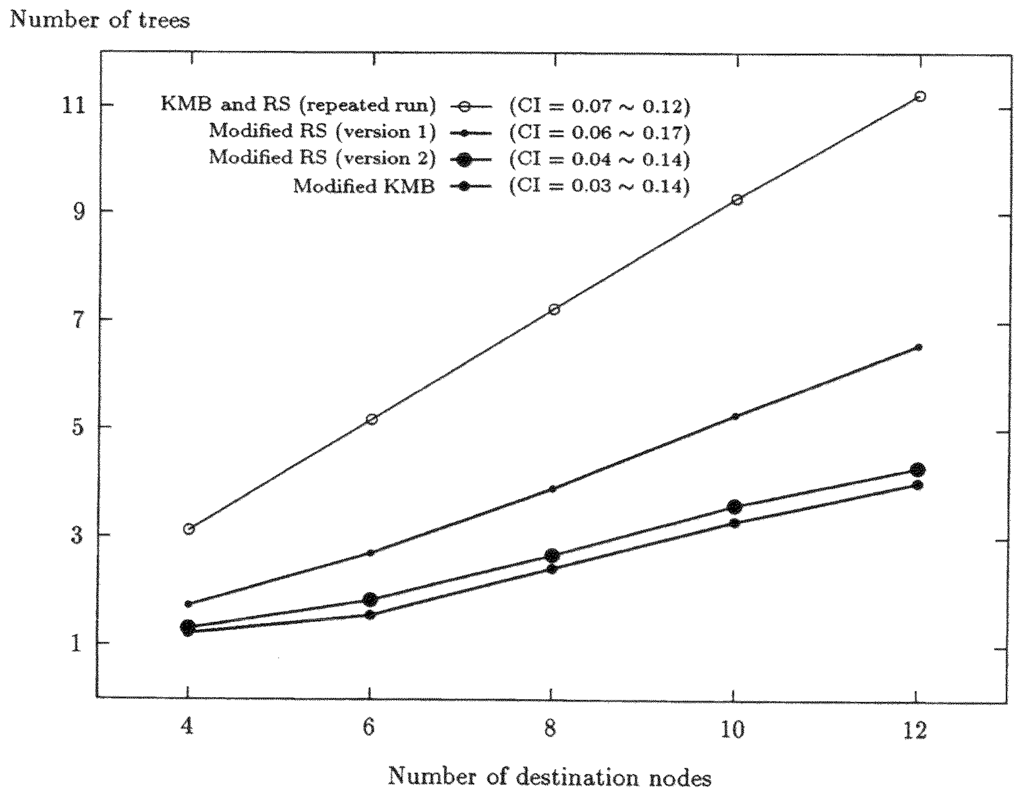


Figure 4.2: Average Number of Trees in Multicast Route Sets

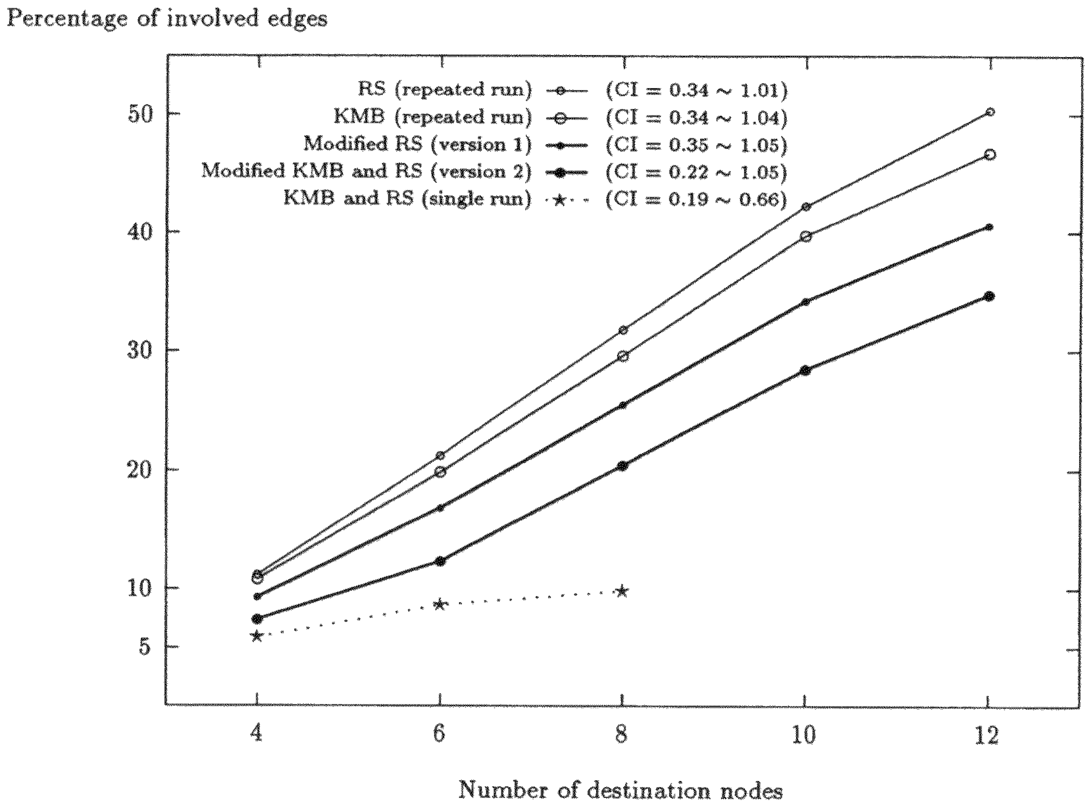


Figure 4.3: Percentage of Network Links Used by Multicast Trees

Results presented in Figures 4.2 and 4.3 reflect the maintenance cost of multicast paths. It is found that the tree sets obtained by the modified algorithms have much smaller numbers of trees and take fewer network links to form the trees than the repeated-run versions of the original algorithms. Undoubtedly, the single-run versions have the best performance in this respect due to the single path nature of their approach. However, their curve stops when more than eight destination nodes are involved in a multicast group because a 100% failure rate is incurred as shown later in Figure 4.4.

The two versions of the RS algorithm again show the effect of balancing the weights of link capacities and distances. Contrary to the previous case on multicast cost, preference of considerations on link capacities gains positive effects: decreasing numbers of Steiner trees and involved edges.

In the multi-path approach with link capacity integration, the modified KMB algorithm gives results very close to those of the second version of the modified RS algorithm. This is because the maximum capacity tree search used in the KMB algorithm is similar to the link-bandwidth-only cost function employed in the second version of the RS algorithm. The same observation can be made for the results of multicast cost in Figure 4.1 and failure rate in Figure 4.4 as well.

Figure 4.4 illustrates the failure rates of the algorithms. These are relative figures because of the difficulty in finding a precise and efficient algorithm for this application. In fact, absolute figures are not necessary for the purpose of comparison. It is therefore assumed that if none of the algorithms can find a satisfactory solution of a Steiner tree set, such a solution does not exist for this sample network.

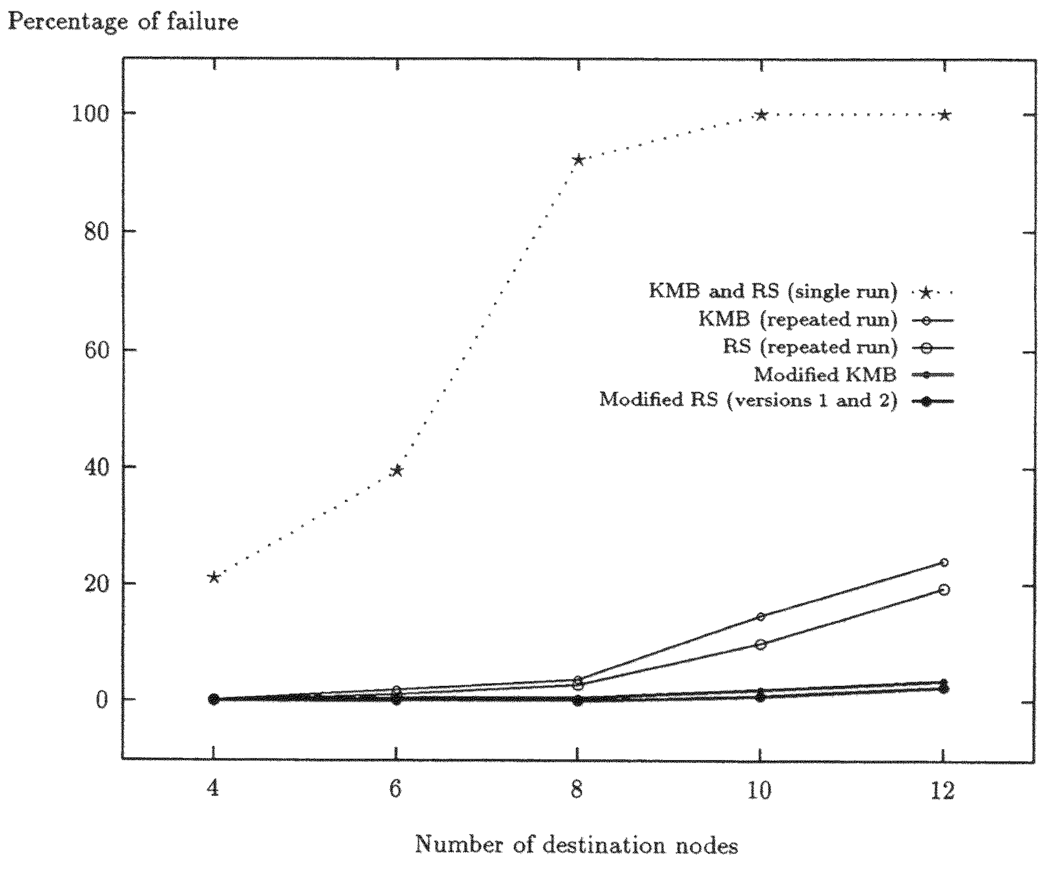


Figure 4.4: Failure Rates of the Routing Algorithms

The result shows that the multi-path approach with link capacity integration has extremely good performance. For the cases in which the destination nodes are a small portion of the total network nodes, the failure rates of the modified algorithms are close to zero. For other cases, the figures are still less than 4% while the repeated run algorithms have a 1 in 4 (KMB) or 5 (RS) chance of failure and the single run algorithms have no success at all.

Figure 4.5 verifies the computational complexity of the algorithms analysed in the previous section. The computation times are normalised to those of the single run KMB and RS algorithms, which happen to have the same computation times (and are faster than any of the modified algorithms). It shows that all of the curves are linear with respect to the number of the destination nodes except the one corresponding to the modified KMB algorithm which is quadratic. These are exactly the same as the analytical results.

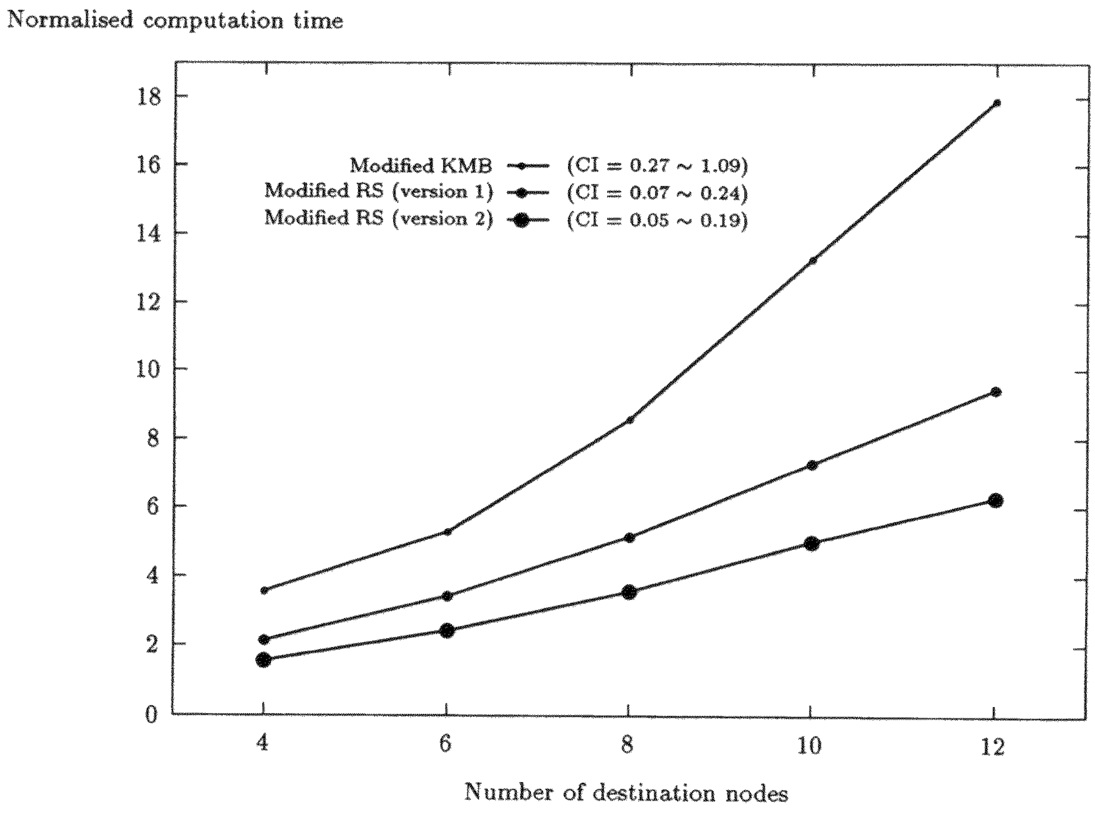


Figure 4.5: Normalised Computation Time of Algorithms

The conclusion from this simulation is that the more the link capacity consideration is emphasised, the better chance there is that a satisfactory solution of a Steiner tree set can be found and the less maintenance effort is required in terms of the numbers of multicast trees and involved edges. The unavoidable negative effect is that the cost of a multicast path increases, if such a cost is related to link distance.

It is noted that a particular application can make a proper trade-off between these effects to suit its environment. For example, single run algorithms can be very efficient in a small area and bandwidth abundant network environment, although they present impractically high failure rates for the sample networks in this simulation.

4.5 Implementation

There have been several network protocol architectures designed for high speed and multi-service communications, such as the *Xpress Transfer Protocol* [Chesson 91] and *Multi Service Network Architecture* [McAuley 90]. A common feature among them is the use of virtual circuit based connections to achieve fast processing of data packets. This makes it possible to implement the path finding algorithms discussed above since the basis for stream multicast is connection-oriented communication.

To implement a centralised stream multicast path finding algorithm, a network protocol engine has to know the state (unused capacity and distance if required) of each link in the network. Propagation of distance information is already required by widely deployed *link state* routing protocols [Perlman 91], such as the ARPANET routing protocol [McQuillan 80], the *Routing Information Protocol* [Hedrick 88], and more recently the *Open Shortest Path First* [Moy 91] and *Intermediate System-to-Intermediate System* [ISO90] protocols. It would be fairly straightforward to include an extra parameter, link capacity, in the propagated information used by these protocols. Problems may arise in a large network requiring a huge storage space for a complete image of the network and heavy computation overhead. Unfortunately, these are inherent characteristics of centralised algorithms, which can be solved by either hierarchical [Tsai 89] or distributed schemes as discussed in the next chapter.

4.6 Other Applications

As a mathematical model, the link capacity integrated path finding algorithms discussed so far can easily be applied in many other situations beside the internet stream multicast. Multi-path unicast routing and multimedia switching subsystems are two examples of these additional uses.

4.6.1 Multi-path Unicast Routing

Multi-path unicast routing is a traffic diversion method which uses more than one route simultaneously between a single source and destination pair, as illustrated in Figure 4.6 where three paths are used between nodes S and D. It has been shown to offer a number of advantages over single path routing in unicast communications [Breslau 91]. These include association of different routing criteria with different routes, network load distribution, and adaptation to network status changes. In particular, when the bandwidth requirement of a connection is too large to be met by any single path, such a routing method becomes a necessity [Jiang 92].

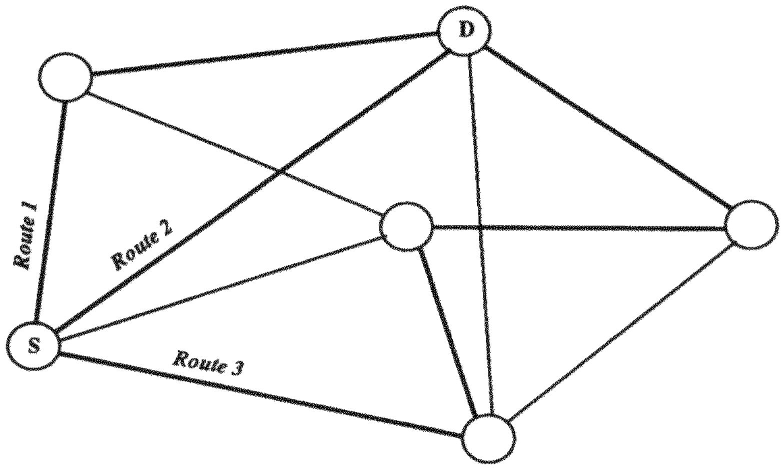


Figure 4.6: Multi-path Unicast Routing

Although many multi-path routing schemes have been proposed in the literature [Attar 81, Lai 85, Gardner 87, Wang 90, Breslau 91], the modified Steiner tree algorithms discussed in this chapter can be used directly for some simple situations where bandwidth limitation is the only reason to use a multi-path approach to unicast routing. In these cases, the problem becomes a special application of the multicast path finding algorithms where $|X| = 2$, and the target is to find a set of routes with capacities of $B_1, B_2, ...B_n$ and $\sum_{i=1}^{n} B_i$ is equal to or larger than the required bandwidth between the pair of transmitting and receiving nodes.

4.6.2 Multimedia Switching Subsystems

A multimedia switching subsystem is illustrated in Figure 4.7. It is an abstracted architecture modelled after two real projects which are currently carried out by the Olivetti Research Laboratory at Cambridge and the University of Cambridge Computer Laboratory. The first is named *Medusa*, a follow-up to the Pandora project [Hopper 90], which aims to provide multiple video cameras at a single location. The second is a fast packet switch based desktop area network [Hayter 91], which attempts to replace the conventional bus oriented interconnection architecture in computer workstations.

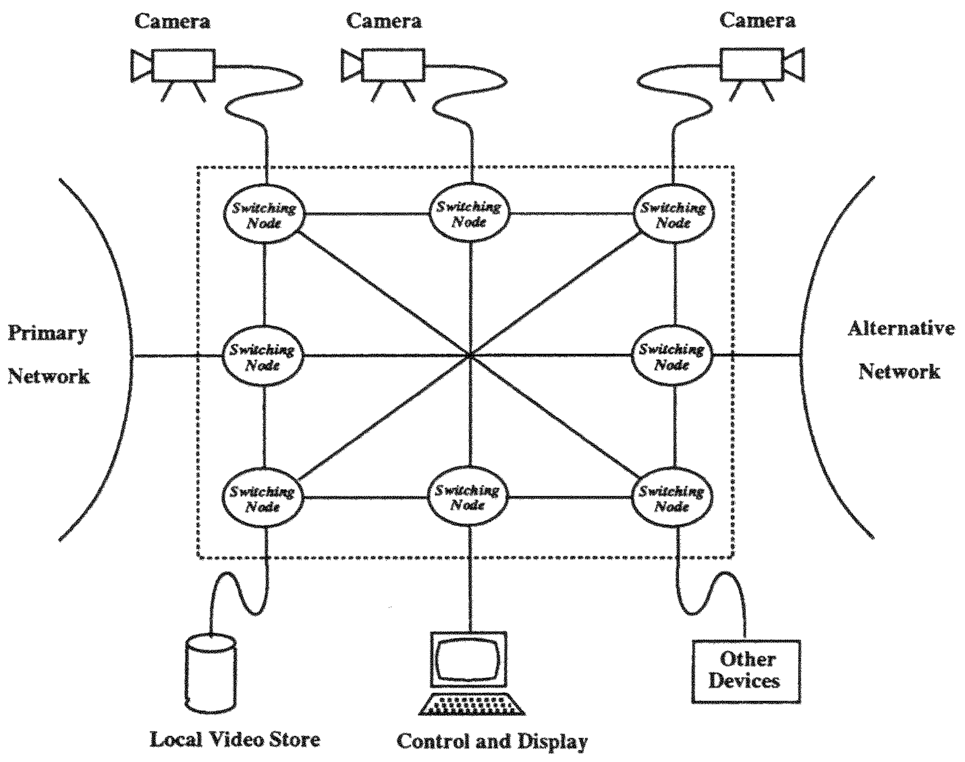


Figure 4.7: A Multimedia Switching Subsystem

When the above two systems are combined, a multimedia switching subsystem is one obvious result. Special characteristics of the new system are that multiple video sources are available both locally and remotely, and these sources are attached to a short distance but possibly high bandwidth mesh network. Since the complete network is encapsulated into a subsystem, a high degree of network connectivity can easily be achieved when there are large numbers of video sources. The system is therefore flexible enough to suit different application requirements.

Stream multicast is probably unavoidable in such a system because both local and remote video sources may be desired by a local display, storage medium, or any other receiver connected through the networks at the same time. In these situations, the system performance would undoubtedly be improved if a proper stream multicast path finding algorithm were incorporated.

Either the single path or the multi-path approach to the stream multicast routing may be employed depending on the relative communication bandwidth available on the interconnection links with respect to the video stream bandwidth. If plenty of communication bandwidth is available on each link, the single path approach is the best choice to obtain simplest multicast paths, while in the opposite case, the multi-path approach becomes more appropriate to avoid congestion and failures.

4.7 Summary

Although conventional Steiner tree algorithms can not be used directly for multipoint broadband communications, such as desktop video conferencing, because of the bandwidth problem, they have been intensively researched over years. As a result, there have been a number of well-known heuristics, such as the KMB and RS algorithms. This situation can be exploited when designing a stream multicast path finding algorithm.

Three approaches to designing such an algorithm based on the conventional Steiner tree heuristics are possible. The single path approach is the simplest one with almost no changes to the original algorithms except checking the communication bandwidth available on each link. It has been shown by simulation that this approach may incur an extremely high failure rate, although its solutions have the simplest forms and require minimum maintenance effort from the network.

Multi-path approaches, on the other hand, display the opposite effects to those of the single path one. However, the major problem with the link capacity check multi-path approach is that its failure rate is still rather high, i.e. up to 25% as demonstrated by the simulation. Link capacity integration is therefore proposed to incorporate bandwidth considerations throughout the path finding process, and is proven effective by a failure rate drop below 4%.

Investigations were made into two different ways of integrating link capacities. Trade-offs among multicast cost, extra links, Steiner nodes and failure rate were found to be inevitable. Therefore, a proper form of link capacity integration can only be decided by applications, based on the simulation result presented.

The adapted stream multicast path finding algorithms remain solvable in polynomial time. The computational complexity is one degree higher than those of the original heuristics with respect to the number of destination nodes, which is acceptable considering multiple paths have to be found. This analytical result has been further verified by the simulation.

Implementation of centralised stream multicast path finding algorithms in a network protocol engine is fairly straightforward and is suitable for an internet environment dominated by link state routing mechanisms. It has also been found that the algorithms designed for the stream multicast path finding can be readily applied to a number of other applications. These include multi-path routing in unicast communications and in a multimedia switching subsystem.

Chapter 5

Distributed Stream Multicast Path Finding

When global knowledge of either the network topology or bandwidth availability is not obtainable by network nodes, centralised path finding algorithms become useless. Such situations demand distributed algorithms which are based on local information only. This chapter presents a distributed scheme for stream multicast path finding based on very limited local information. Simulation results show its performance is comparable with that of the centralised algorithms discussed in the previous chapter.

5.1 Conditions and Approaches for Distributed Path Finding

There are many situations where the knowledge about a complete network available to an individual node is strictly restricted. One reason for this is that the network is very large so that information of status changes cannot be propagated in time. In these cases a distributed, instead of centralised, algorithm must be used because it only requires that each node is aware of the network status in its immediate neighbourhood.

The basic approach which may be used by a distributed algorithm is the communication, usually by exchanging control packets, between neighbouring nodes. There are, however, two different ways of implementing it.

- In a *concurrent* scheme, all of the destination nodes may start to build their Steiner trees simultaneously. Redundant paths are possible, and can be discarded at a final clean-up stage.
- In a *sequential* approach, each destination node constructs its Steiner tree in turn on an ordered basis. The process may stop at any time when enough Steiner tree capacities have been found for the total bandwidth requirement of the multicast group.

The concurrent method needs an overload avoiding or resolving mechanism so that no link is used by multiple paths whose total required bandwidth exceeds the available capacity of the link. On the other hand, the sequential approach demands an arbitrary sequencing mechanism which determines the order in which each destination node constructs its Steiner tree.

The different characteristics of the two schemes mean that the concurrent one may perform very efficiently in a bandwidth abundant network while the sequential one works better in a link capacity limited network. This is because when there is plenty of bandwidth available on each network link, overloads are less likely to happen, and an overload avoiding or resolving process is normally both complicated and time consuming on the distributed basis. The sequential approach also has the extra advantage that destination nodes having higher bandwidth requirements can be arranged to build their multicast paths first. This ensures more effective use of links with high capacities.

In both the concurrent and the sequential approaches, *broadcast storm* is a common problem which must be avoided in any packet exchange based distributed algorithm where broadcast is used to reach unknown but potentially useful nodes in the network. Its symptom is that a broadcast packet results in endless retransmissions of its duplicates in the network so that the normal operation of the network is badly affected or even stopped. A simple way to solve the problem is by including an *age* field in broadcast packets which is decreased by one each time it is received by a node. Thus, the maximum number of retransmissions of a packet is restricted.

Another concern for a distributed algorithm is the amount of its communication overhead. This is the amount of control information exchanged between network nodes and the efficiency of this process. To a large extent, it reflects the complexity of the algorithm. Since overload resolution in the concurrent approach usually incurs a high communication overhead, especially in bandwidth-limited situations, the sequential scheme is a much better choice in this respect.

The distributed algorithm proposed in the next section is based on the sequential approach because of the above discussion. Requirements for the operation of such algorithms are as follows:

- A network-wide unique identification or naming scheme is needed so that a sequencing mechanism can be built on it.
- Each node knows the bandwidth available on all of the links connected to it.
- For a version where link distance is a measure for the cost of a multicast path, each node also knows the distances of all links connected to it.
- All of the nodes should be aware of the scale of the network so that the age of a broadcast packet can be properly set, and path finding failures can be detected.

5.2 A Distributed Scheme

The distributed path finding algorithm proposed in this section uses a sequential approach. Based on its unique identifier in the network, each destination node takes turns to build its Steiner tree. When destination nodes have different bandwidth requirements, they are classified into different bandwidth categories. Those with higher bandwidth requirements start their tree construction processes before the others. The identification based sequencing mechanism operates inside each category.

Packet exchange is the basic communication method employed in the distributed algorithm. It takes place mainly between immediately neighbouring nodes. There are seven types of control packets used in the algorithm, which are:

- A *TRY* packet contains the current source node number, a list of destination nodes, the minimum bandwidth required for the tree, parameters of an available path from the source to this node (i.e P_{try} defined in Table 5.1), and the maximum transmission range (i.e. the age) of the packet. It is the major means used by a node trying to make the tree grow to reach all of the destination nodes.
- A *CONNECT* packet contains the source node number, P_{try}, and the destination node number from which the packet originates. It is the response sent by destination node to the source node.
- A *DISCONNECT* packet only contains the source node number. It is used by a non-destination node to remove itself from the tree (as an unnecessary leaf) and to inform the corresponding neighbour.

- A *CHECK* packet contains the source node number and the list of the destination nodes. It is sent out by the source node to stabilise the tree when it is found and prevent further changes to the path.
- A *CONFIRM* packet contains the same types of information as those in a *CONNECT* packet. It is a response from a destination node to the source node to confirm the path and the available bandwidth.
- A *DISMISSED* packet contains the source node number only. It is broadcast from the source node to announce the failure of a path finding process.
- A *COMPLETED* packet contains the source node number and the available capacity of the tree. The source node announces the successful completion of a tree search by broadcasting this packet.

Note that only *TRY* packets may cause a broadcast storm due to their widespread transmission, and thus need an age field to prevent it. Other types of packets are either unicast to a single destination (the source node or a neighbouring node) or multicast through a fixed route.

During a tree searching process, each node must maintain the following information:

- the incoming node, which is the parent node of this node on the tree,
- an outgoing table, which lists all the links connecting child nodes on the tree to this node, and
- a set of parameters, $P_{current}$ defined in Table 5.1, describing the available path from the source node to this node.

Similar integrating methods for link capacities as used by the centralised path finding schemes are also explored in the distributed algorithm. These are referred to as the *bandwidth only* and *bandwidth and distance* versions of the algorithm. They differ only in whether distance is included in the parameter set and a path selection function, which are both defined in Table 5.1.

The algorithm is initialised by a destination node (as the first source node) starting the *source node procedure*. It involves three stages: *tree building, confirming*, and *concluding*. Their relations are shown in Figure 5.1.

A tree building stage begins with the source node broadcasting[1] a *TRY* packet with BW_{try}

[1] In the description of the path finding algorithm, broadcast means to send a copy of the packet on each link connected to the node which has a capacity not less than the maximum value among the list of currently required bandwidth $B_i(i = 1, ..., n)$.

Version	Bandwidth only	Bandwidth and Distance
P_{try}	BW_{try}	$\{BW_{try},\ DIST_{try}\}$
$P_{current}$	$BW_{current}$	$\{BW_{current},\ DIST_{current}\}$
F_{try}	$min(BW_{try},\ BW_{incoming})$	$\frac{min(BW_{try},\ BW_{incoming})}{DIST_{try} + DIST_{incoming}}$
$F_{current}$	$BW_{current}$	$\frac{BW_{current}}{DIST_{current}}$

where,

BW_{try}	— bandwidth available from the source node to the sending node of the *TRY* packet which contains this parameter;
$BW_{incoming}$	— bandwidth available on the link from which the *TRY* packet is received;
$BW_{current}$	— currently recorded available bandwidth from the source node to this node, which is replaced by $min(BW_{try},\ BW_{incoming})$ when the new path is taken;
$DIST_{try}$	— distance from the source node to the sending node of the *TRY* packet which contains this parameter;
$DIST_{incoming}$	— distance of the link from which the *TRY* packet is received;
$DIST_{current}$	— currently recorded distance from the source node to this node, which is replaced by $DIST_{try}+DIST_{incoming}$ when the new path is taken.

Table 5.1: Parameter and Function Definitions

set to the total bandwidth requirement, and $DIST_{try}$ set to zero for the bandwidth and distance version. This is because initially the tree consists of the source node only and no more than $\sum_{i=0}^{n} B_i$ is required for the tree capacity.

The broadcast triggers other nodes to find a capable path back to the source node. When all of the destination nodes report the finding of such a path, the tree building stage ends successfully. However, it is possible that a capable Steiner tree does not exist. In this case, none of the nodes know the situation because after the propagation of *TRY* packets they either wait for replies or are not contacted at all. This can be detected by a timer at the source node.

Since the maximum number of links in a network is $\frac{1}{2}(|V|^2-|V|)$ as analysed in the previous

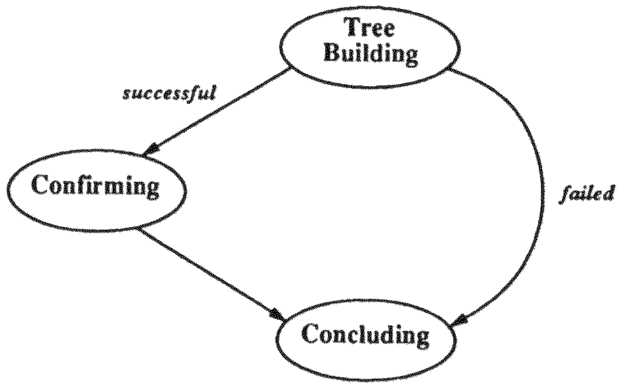


Figure 5.1: Operation Stages of the Distributed Algorithm

chapter, the maximum round trip time between any pair of nodes is this value times the average transmission time between neighbouring nodes plus the average processing and queueing delays at a node. This provides a safe upper limit for the timer. However, this may not be necessary in practice. The simulation, to be presented later in the chapter, shows that when a tree is found the search time is only around 2% of the above value. Also, in large scale networks, applications may choose a small age for *TRY* packets to restrict the tree searching area, and the timer should therefore be set accordingly.

In a distributed environment, it is impossible to have all nodes know immediately when a Steiner tree has been found. This means that they will keep trying to find a better path to the source node. The tree becomes stable only when *TRY* packets disappear from the network or the safe upper limit of the timer elapses. This is not a good choice for the above reasons. The alternative is to include a confirming stage to force the tree to be stable. This may result in a non-optimal solution, but can shorten the process significantly.

A confirming stage begins with the source node sending out *CHECK* packets, and ends by all of the destination nodes replying with *CONFIRM* packets. While a *CONFIRM* packet travels back to the source node, it stabilises the path.

The concluding stage is to inform the network of the result of the path finding process. In the event of a time-out from the timer, there is no confirming stage, and a failure report (i.e. a *DISMISSED* packet) is sent to conclude the algorithm.

The detailed procedure for a source node is given below in pseudo C code.

Source Node Procedure:

```
STEP 1:  Set $timer$ to a proper value determined by applications according
             to the scale of the network or the maximum $age$ of packets;

STEP 2:  Broadcast $TRY$ with $BW_{try} = \sum_{i=1}^{n} B_i$ and
             $DIST_{try}$ = 0 for the bandwidth-and-distance version;

STEP 3:  Perform the intermediate procedure (see below) until $CONNECT$
             received from all of the destination nodes or time-out;

STEP 4:  IF $timer$ is time-out
             THEN
                 Broadcast $DISMISSED$;
                 Exit;                             /* path finding failed */
             ELSE
                 Send $CHECK$ on outgoing links;        /* path found */

STEP 5:  Perform the intermediate procedure until $CONFIRM$ received from
             all of the destination nodes;

STEP 6:  Calculate the tree capacity $BW_{tree}$;
         Send $COMPLETED$ on outgoing links;
         Exit;                                 /* algorithm finished */ □
```

Source Node Intermediate Procedure:

```
SWITCH (type of the received packet)
    CASE $TRY$:
        IF the sending node is in the outgoing table
            THEN
                Delete the sending node from the outgoing table;

    CASE $CONNECT$:
        Record originated destination node number and
                bandwidth available to the node;
        Add the sending node to the outgoing table;
        IF $CHECK$ has been sent
            THEN
            Send CHECK to the sending node again;

    CASE DISCONNECT:
        Delete the sending node from the outgoing table; □
```

When a node receives a *TRY* packet whose destination node list includes this node, it starts the *destination node procedure*. Otherwise, it starts the *other node procedure*.

Both procedures have a similar packet driven loop structure of processing and modifying their recorded information based on the packet received. The major decision in selecting a link to be part of the tree is made every time a *TRY* packet is received. F_{try} (as defined in Table 5.1) is a measure of the quality of the new path through the node which sent the the *TRY* packet, while $F_{current}$ measures the currently recorded path. If $F_{try} > F_{current}$, the new path is taken in place of the old. As this process proceeds, the tree tends to be optimised for the specified requirements.

Although not specified explicitly in the following description of the procedures, the corresponding amount of bandwidth should temporarily be reserved on its incoming link[2] when a node broadcasts *TRY* packets. The reason for this is to guarantee the bandwidth stated on the *TRY* packets. Accordingly, it should be released when a *DISCONNECT* packet is either sent or forwarded. The final reservation is made for BW_{tree} when *COMPLETED* packets traverse the tree.

Due to the fact that a node does not know the exact time when a confirming stage starts, it has to prevent inconsistencies in a path which result from any changes since a *CHECK* packet was sent by the source node. This is done by the receiving node verifying whether the *CHECK* packet is received from its incoming node. The confirmation steps are performed only when the verification is true.

Since the procedures for destination and other nodes differ in their details due to the different node types, they are given separately below:

Destination Node Procedure:

```
SWITCH (type of the received packet)
    CASE TRY:
        IF $F_{try}$ > $F_{current}$ AND this node has not sent any CONFIRM yet
            THEN
                Record new incoming node and new $P_{current}$;
                Send CONNECT to the incoming node;
```

[2]This is the link between a node and its incoming node.

```
                Broadcast TRY;
          IF the sending node is in the outgoing table
              THEN
                  Delete the sending node from the outgoing table;

      CASE CONNECT:
          Forward the packet onto the incoming link;
          Add the sending node to the outgoing table;

      CASE DISCONNECT:
          Delete the sending node from the outgoing table;

      CASE DISMISSED:
          Forward the packet onto the outgoing links;
          Delete all records;
          Exit;                                  /* path finding failed */

      CASE CHECK:
          IF the sending node is the incoming node
              THEN
                  Send CONFIRM to the source node via the incoming node;
                  Forward CHECK onto outgoing links if there is any;
                      /* No further change of incoming node is allowed
                         after doing this. */

      CASE CONFIRM:
          Forward the packet onto the incoming link;

      CASE COMPLETED:
          Record the incoming node and the outgoing links;
          Reserve $BW_{tree}$ on related links;
          Forward COMPLETED onto outgoing links if there is any;
          Exit;                                       /* path found */ □
```

Other Node Procedure:

```
SWITCH (type of the received packet)
    CASE TRY:
        IF the sending node is in the outgoing table
            THEN
```

```
                Delete the sending node from the outgoing table;
                IF the outgoing table is empty
                    THEN
                         send DISCONNECT to the incoming node;
                              /* This node is currently removed from
                                 the path. */
         IF $F_{try} > F_{current}$ AND this node has not forwarded any CONFIRM
             THEN
                  Record new incoming node and new $P_{current}$;
                  Broadcast TRY;
                  IF the outgoing table is not empty
                      THEN
                           Send CONNECT to the incoming node;

    CASE CONNECT:
         Forward the packet onto the incoming link;
         Add the sending node to the outgoing table;

    CASE DISCONNECT:
         Delete the sending node from the outgoing table;
         IF the outgoing table is empty
             THEN
                  send DISCONNECT to the incoming node;
                     /* This node is currently removed from the path. */

    CASE DISMISSED:
         Forward the packet onto the outgoing links;
         Delete all records;
         Exit;                                  /* path finding failed */

    CASE CHECK:
         IF the outgoing table is not empty
             THEN
                  IF the sending node is the incoming node
                      THEN
                           Forward CHECK onto outgoing links;
                                      /* This node is a Steiner node. */
             ELSE
                  Send DISCONNECT to the sending node;
                           /* This node is not currently on the path. */
```

```
CASE CONFIRM:
     Forward the packet onto the incoming link;
       /* No change of incoming node is allowed after doing this. */

CASE COMPLETED:
     Record the incoming node and the outgoing links;
     Reserve $BW_{tree}$ on related links;
     Forward COMPLETED onto outgoing links;
     Exit;                                          /* path found */ □
```

At the end of a concluding stage, a similar step to that of both the modified KMB and RS algorithms is taken to allocate the tree capacity to the bandwidth requirements from various destination nodes. A new round of tree searching is then started with the unallocated destination node which has the smallest identification number as the new source node. The algorithm repeats until the total capacity of Steiner trees matches the total bandwidth requirement or a tree searching process fails.

Rather like the centralised path finding schemes, this algorithm does not guarantee to find a set of capable Steiner trees even when such a solution is available from the network. However, in each round of tree searching, it is able to find a Steiner tree if it is available. This is because if one or more Steiner trees are available in a network, the broadcast of *TRY* packets should cover at least one of them, and the solution is the best among those being covered.

5.3 Computational Complexity

The computational complexity of the distributed path finding algorithm can be analysed stage by stage. Since the processing at each node only involves simple calculations and selections based on its local information, the computational complexity is proportional to the maximum length of paths along which packets travel.

At the tree building stage, the longest path from which a destination node may be reached by a *TRY* packet consists of all of the links in the network. Since the maximum number of edges for a network of a given number of nodes is $\frac{1}{2}(|V|^2 - |V|)$, this is also the length of such a path. If the path happens to be a "good" one, a *CONNECT* packet is generated, and travels along the same route back to the source node as the last *CONNECT* packet received. Therefore, the worst case performance is $O(|V|^2)$.

The confirming stage has a similar request-response cycle as the tree building stage except it is in a non-broadcast style. Thus again its complexity is $O(|V|^2)$ in the worst case when the longest path identified above is included in the tree. The complexity of the concluding stage is also the same because it only consists of the first half cycle of the confirming stage.

Since in the worst case each destination node has to find its own Steiner tree, the computational complexity of the whole path finding scheme is $O(n|V|^2)$. This is better than both types of centralised path finding algorithms discussed in the previous chapter ($O(n^2|V|^3)$ for the modified KMB and $O(n|V|^3)$ for the modified RS), even though the centralised algorithms have the natural advantage of possessing global knowledge of a network.

5.4 Simulation Results

The simulation of the distributed path finding algorithm was performed on the same set of sample networks described in the previous chapter. Results on both the bandwidth-and-distance and bandwidth-only versions are presented in Figures 5.2 – 5.6. For comparison of the distributed and centralised schemes, the performance of both versions of the modified RS algorithms are also given in dotted curves.

To measure the accuracy of the simulation, the 95% confidence intervals for the results plotted in Figure 5.2 are given in Table 5.2. All of the curves in other figures are indicated with the ranges of 95% confidence intervals for the simulation results on them, although such tables are not repeated. The simulation is regarded as reliable since the 95% confidence intervals are smaller than 5% of the corresponding results.

	95% Confidence intervals				
Number of destination nodes	4	6	8	10	12
Bandwidth-and-Distance version	0.11	0.14	0.14	0.17	0.20
Bandwidth-only version	0.20	0.21	0.22	0.22	0.24

Table 5.2: 95% Confidence Intervals of Normalised Multicast Costs

Figure 5.2 shows that the normalised tree costs resulting from both versions of the distributed path finding algorithm are very similar to those from the centralised versions. The same effects of the different ways of integrating link capacities into centralised algorithms are also found in the distributed versions, i.e. the greater the shift of emphasis from link distance to its capacity, the higher the expected multicast cost.

An interesting point to be noted in the above figure is that the distributed bandwidth-and-distance version gives slightly higher normalised multicast costs than the centralised

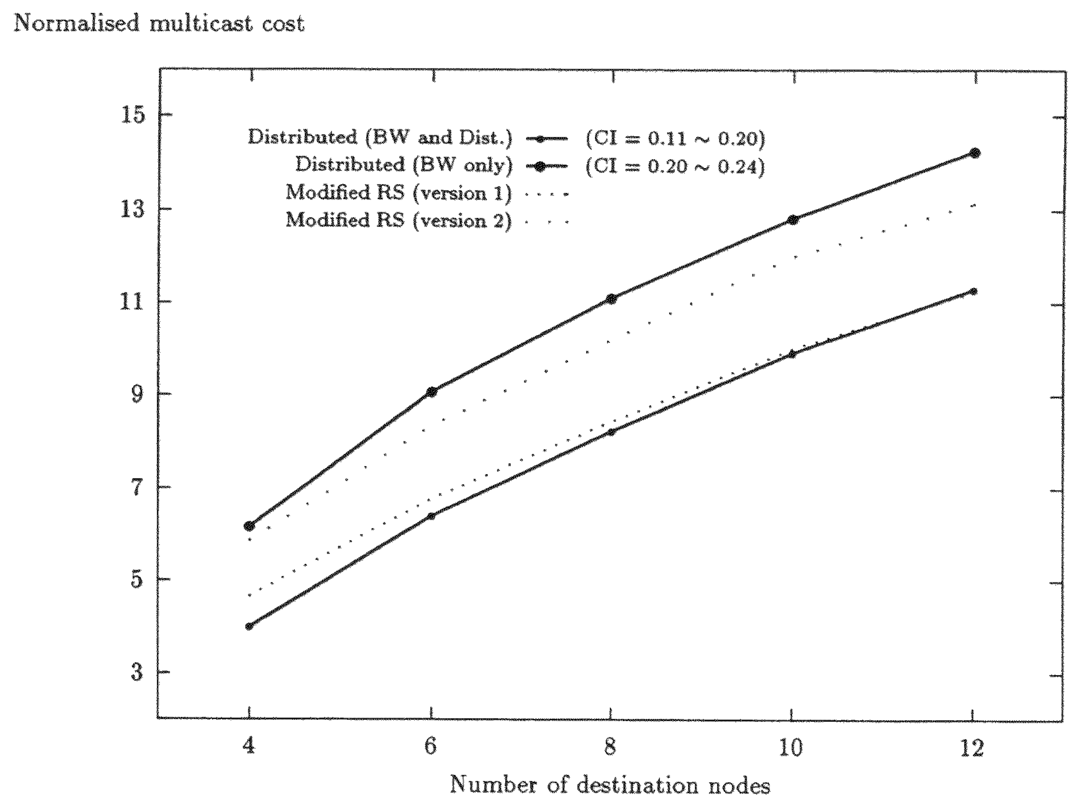


Figure 5.2: Normalised Cost of Multicast Trees

one. Both curves start from almost the same point, but move apart from each other when the number of destination nodes increases. The bandwidth-only versions, on the other hand, show the opposite trend.

This phenomenon can be explained by the difference between the cost functions used in the centralised algorithms and the path measure functions used in the distributed versions. The relationships between them are as follows:

	<u>Centralised</u>		<u>Distributed</u>
BW-and-Dist.:	$cost = \frac{distance}{[nc/\sum_{i=1}^{n} B_i]}$	$\Longleftrightarrow$	$\frac{1}{F_{try}} = \frac{DIST_{try} + DIST_{incoming}}{min(BW_{try},\ BW_{incoming})}$
BW-only:	$cost = \frac{1}{[nc/\sum_{i=1}^{n} B_i]}$	$\Longleftrightarrow$	$\frac{1}{F_{try}} = \frac{1}{min(BW_{try},\ BW_{incoming})}$

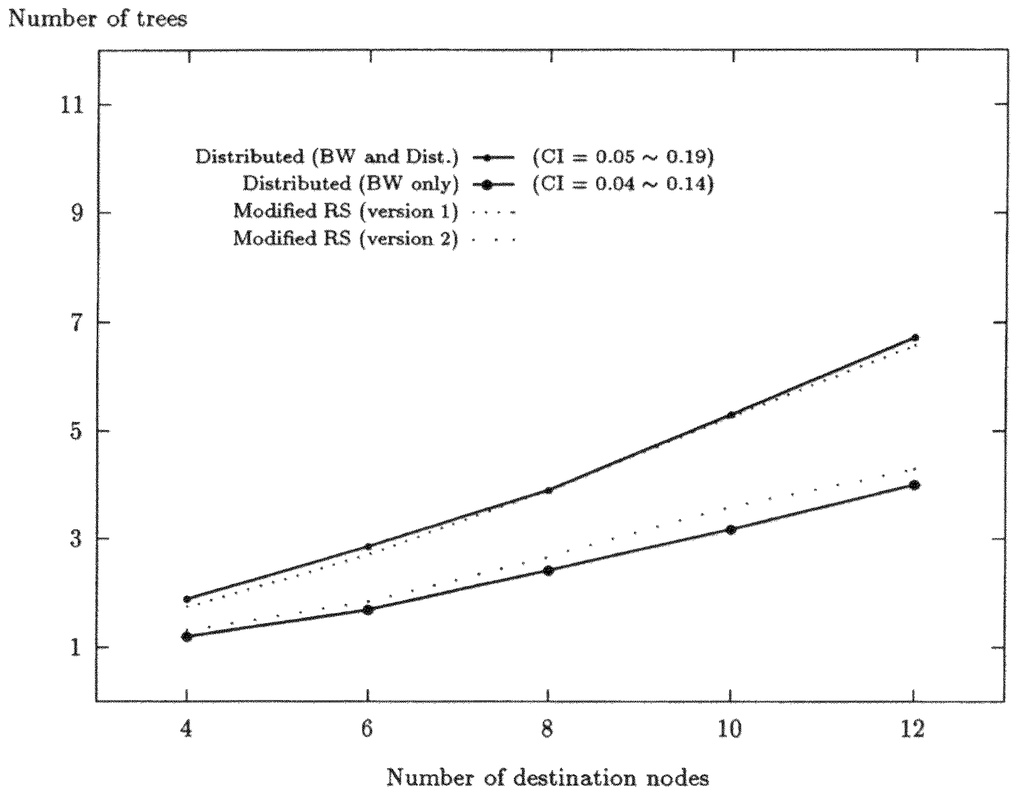


Figure 5.3: Average Number of Trees in Multicast Route Sets

In comparison to the *cost* of the centralised algorithms, two points can be observed from $1/F_{try}$ of the distributed versions:

(a) Unlike the minimum cost path used in the centralised algorithms to join a pair of sub-trees, the length of an attempted path from the source node to a destination node in the distributed algorithm is not necessarily minimised. Furthermore, the bandwidth factor is not maximised either. This means that while the cost functions can be optimally explored in the centralised schemes, F_{try} may result in a non-optimal path in the distributed versions. Therefore, the effect on the performance of the algorithms is that the distributed versions have a tendency to produce higher-cost paths.

(b) On the other hand, the distributed algorithm normally does tend to find short paths, because of the nature of the packet exchange mechanism, i.e. nodes which are closer to the source node should received *TRY* packets earlier. This gives the reverse effect

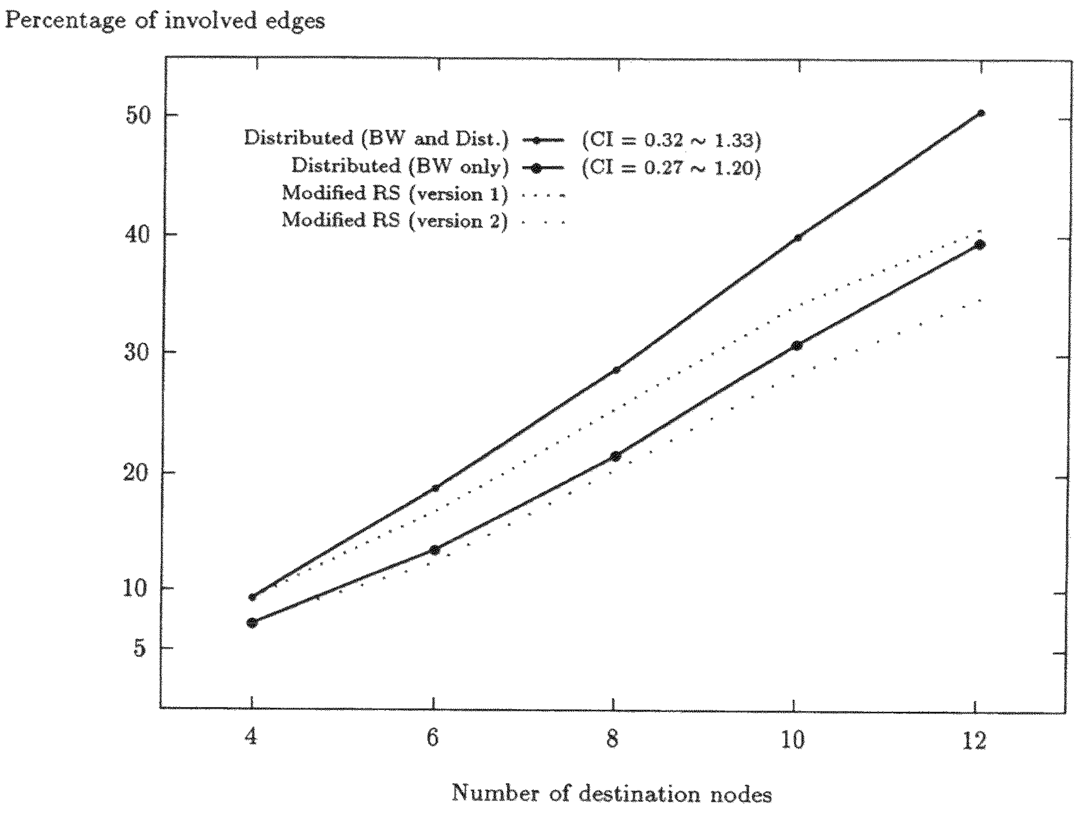


Figure 5.4: Percentage of Network Links Used by Multicast Trees

of that of the previous one.

The relative importance of (a) and (b) varies with different destination node densities in the network. As the number of destination nodes increases, longer paths are required to build more trees, and thus the effect of (a) becomes more significant because the difference between costs of non-optimal and optimal paths also increases. At the same time, the effect of (b) is reduced because the increased execution time of an algorithm gives a wider range of nodes to be reached by *TRY* packets.

The combined effects of (a) and (b) explain the curves in Figure 5.2. Since the bandwidth-and-distance versions are affected by both (a) and (b), and (a) becomes more important than (b) when the number of destination nodes increases, the two curves diverge. For the bandwidth-only versions, however, the effect of (a) is entirely eliminated because link distance is not a factor to be considered. Therefore, the distributed scheme produces slightly lower cost multicast paths when there are fewer destination nodes, but the result

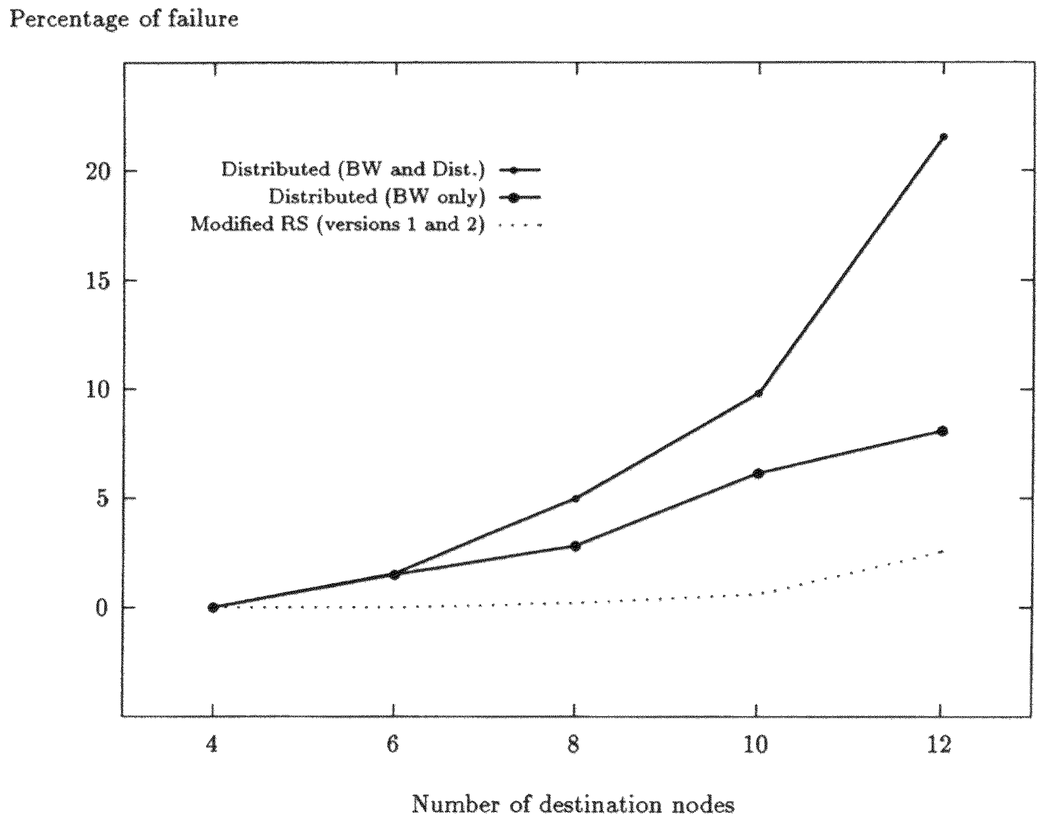


Figure 5.5: Failure Rates of the Routing Algorithms

moves closer to that of the centralised scheme as the number of destination nodes increases.

The maintenance cost of multicast paths and their effect on network traffic are reflected in the curves in Figures 5.3 and 5.4. The results again show similar trends to those found for the centralised path finding algorithms, i.e. increased link capacity integration results in decreasing number of Steiner trees and involved edges.

Both curves of the numbers of trees for the distributed path finding schemes are in fact very close to those for the centralised algorithms, while the percentages of involved edges are higher than those for the centralised versions. The latter effect is due to the fact that the distributed algorithms only try, but do not guarantee, to optimise the bandwidth factor in their path measurement functions. However, the increase is not significant, only reaching about 25% when half of the total network nodes become destination nodes, which is not a usual case in practice.

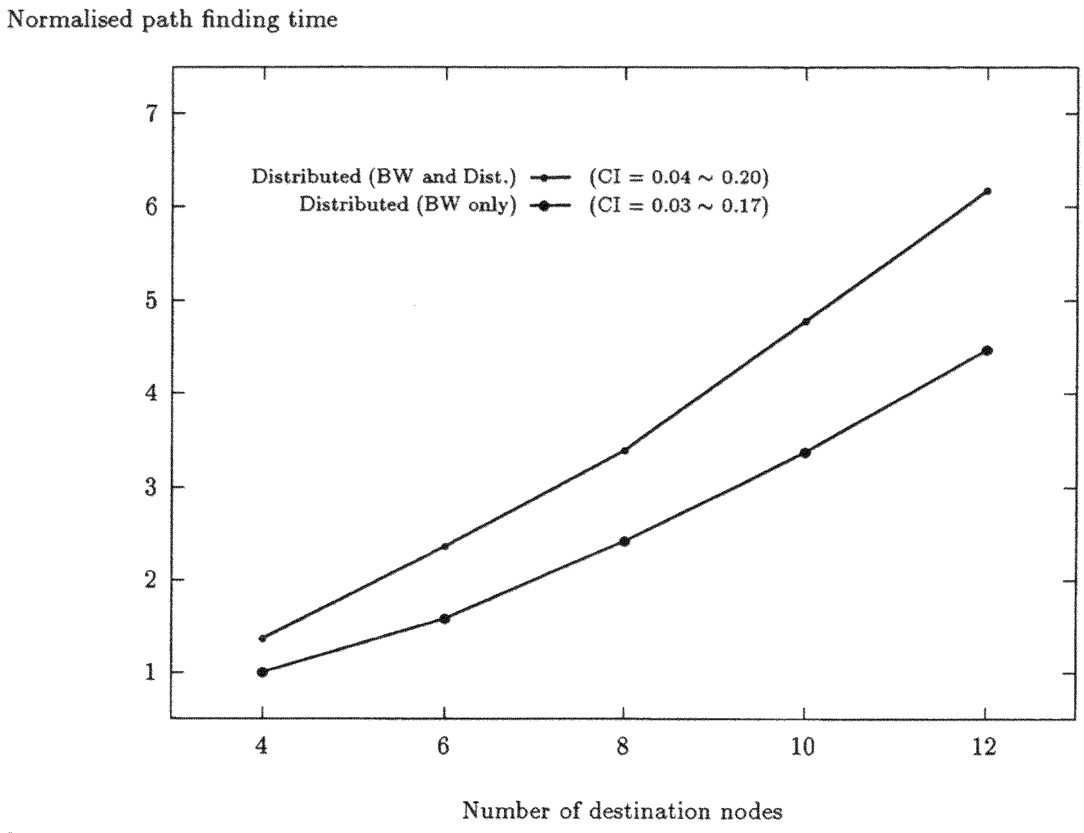


Figure 5.6: Normalised Path Finding Time

Figure 5.5 shows the failure rates of the distributed algorithm which are higher than their centralised counterparts. This is mainly because of the lack of global knowledge of a network and the non-optimal path selection mechanism used in the algorithm. Improvements may be expected by extending the searching range or time in the tree building stage of the algorithm, but this leads to an increase in the complexity of the algorithm.

In fact, if the complexity needs to be low such improvements are not necessary, because the performance is no worse than that of the repeated run versions of the centralised algorithms. While the bandwidth-and-distance scheme shows a comparable result to the centralised version, the bandwidth-only scheme gives a much better performance of 8% failure rate when half the total network nodes become destination nodes.

Figure 5.6 verifies the linear relationship between the computational complexity of the distributed algorithm and the number of destination nodes. The normalisation of the results is made with respect to the minimum value on the figure, i.e. the failure rate of the bandwidth-only version at the point of 4 destination nodes.

In conclusion, the simulation shows that the relative weight of link capacity and distance when considering a distributed stream multicast path finding algorithm is very similar to that in a centralised algorithm. Therefore, the same trade-offs as those discussed for the centralised schemes must be made for individual applications. Note that some aspects of the performance are slightly inferior to centralised versions due to the local-information-only nature of the distributed approach. This should generally be tolerated to reduce algorithm complexity.

5.5 Implementation

Implementation of distributed stream multicast path finding algorithms in a network protocol engine is much simpler than its centralised counterpart in terms of the amount of link state information which must be stored and the computation required. In a distributed environment, only local information and extremely simple computations (as seen from the algorithm presented in this chapter) are necessary. Normally, the communication overhead is also low and packet exchange happens mainly between neighbouring nodes, as observed from the simulation. These properties make distributed algorithms preferable to their centralised counterparts for implementation in a network protocol engine, especially when other aspects, such as multicast cost and failure rate, are not major concerns.

5.6 Summary

Distributed path finding algorithms are useful for applications where global knowledge of a network, which is a pre-condition for centralised algorithms, is not available. Apart from some loose conditions, such as a network-wide unique node identification scheme, the only information required by a node in a distributed environment is the knowledge of the capacity (and distance if it is a factor to be considered in the algorithm) of each link connected to the node.

The distributed scheme presented in this chapter is based on packet exchanges between network nodes. A source node initialises a tree building process by broadcasting a *TRY* packet. A path from the source node is selected by a node if some measure of its quality is better than that of the old one. The sequential approach of searching one Steiner tree at a time is preferable to the concurrent approach because of its simplicity and scalable performance.

To investigate different ways of link bandwidth integration, two versions of the distributed algorithm, the bandwidth-and-distance and bandwidth-only, have been designed and simulated. The results show that their performance variation is very similar to that observed from the centralised schemes, and thus the same trade-offs discussed in the previous chapter also apply to the distributed case.

The most significant performance differences between the distributed and centralised algorithms are their failure rates and computational complexity. The failure rate of the bandwidth-only version of the distributed algorithm reaches 8% (compared to 3% for the modified RS algorithm) when half of the total network nodes are destination nodes. The value for the bandwidth-and-distance version at the same point is much higher, but still no worse than that of the repeated run versions of the centralised algorithms. On the other hand, the computational complexity of the distributed algorithm is one degree lower in terms of $|V|$ than for the centralised schemes (and another degree lower in n than the modified KMB algorithm).

When comparing the simulation results of the distributed schemes with those of the centralised versions, it is also noticed that the former have small performance degradations in certain respects. These result from the conflict in a distributed algorithm between the restriction of information available to a node and the complexity of the algorithm, which in one way or another need to be compromised.

It has been found that implementing a distributed path finding algorithm in a network protocol engine is much simpler than a centralised algorithm due to its limited storage requirements and low overheads in both computation and communication.

Chapter 6

Intermediate Stream Representation

Having discussed stream multicast path finding mechanisms, the rest of the thesis moves back to the signal representation issue at the presentation level. This, as pointed out in Chapter 2, is another important problem to be addressed in a system architecture which supports multipoint digital video communications.

This chapter proposes an *intermediate general representation scheme* for video streams based on the observed strengths and weaknesses of other digital video coding schemes and the proposed use in a multipoint communication environment. The scheme is presented with design considerations for stream architecture, frame structure, and coding descriptions, and is exemplified by a sample Pandora video stream. A possible decoder model is provided, and other related implementation issues are also discussed. A conformance evaluation of its features with the requirements given in Chapter 2 is presented at the end of the chapter, together with a discussion of its relative merits in comparison to the other schemes.

6.1 An Intermediate General Representation Scheme for Video Streams

The survey of basic signal coding types and digital video formats shows two important characteristics in the current situation of digital video representation. Firstly, a variety of digital video schemes have emerged in both telecommunications and computer worlds, but a limited set of basic coding techniques is shared among them. Secondly, there does not exist an easy way to interchange video streams between these schemes, although a few of them were designed with a certain degree of flexibility to accommodate a range of video streams. This situation makes it necessary to find a common intermediate representation method if video applications, especially among multiple parties, are required in heterogeneous networking environments.

The *Intermediate General Video Representation* (IGVR) is designed for such purposes. It is based upon the requirements of the presentation level support for multipoint digital video communications as discussed in Chapter 2. The key issues are interchangeability and scalability in which a hierarchical decoder can be built according to the demand at the receiver. These properties will be evaluated in a later section when this scheme is compared with other related schemes.

6.1.1 Stream Representation Structure

The scheme is similar to a hierarchical language for describing video streams. It has at the bottom level two types of packets: *control* and *data packets.* Control packets have a fixed length of four bytes, which is chosen as a convenient size to group relevant parameters while maintaining the efficiency to change any particular one of them[1]. They provide the structural information of a video stream and other parameters for a decoding process. Each video frame is broken into rectangular blocks which are transmitted in single data packets. The block position in a frame is included in the data packet, and the length of a data packet depends on the particular type of video stream.

A frame is a loose mid-level structure which consists of data blocks preceeded by motion vectors for the blocks if motion compensation is applied. A complete video stream at the top level has two components — a *prologue* followed by a *video* part. The prologue is composed of control packets giving a full description of the video stream ending with a *Silent Period* (SP) packet to be explained later. The video part contains the actual frames and is terminated by a *Stream End* (SE) control packet. It is also possible to have some control packets included in the video part should any parameters change, such as the frame rate or scene. This hierarchical structure is depicted in Figure 6.1.

[1]The fixed length of four bytes for both control packets and headers of data packets is also provided for programming efficiency when the scheme is implemented in software on a 32-bit computer.

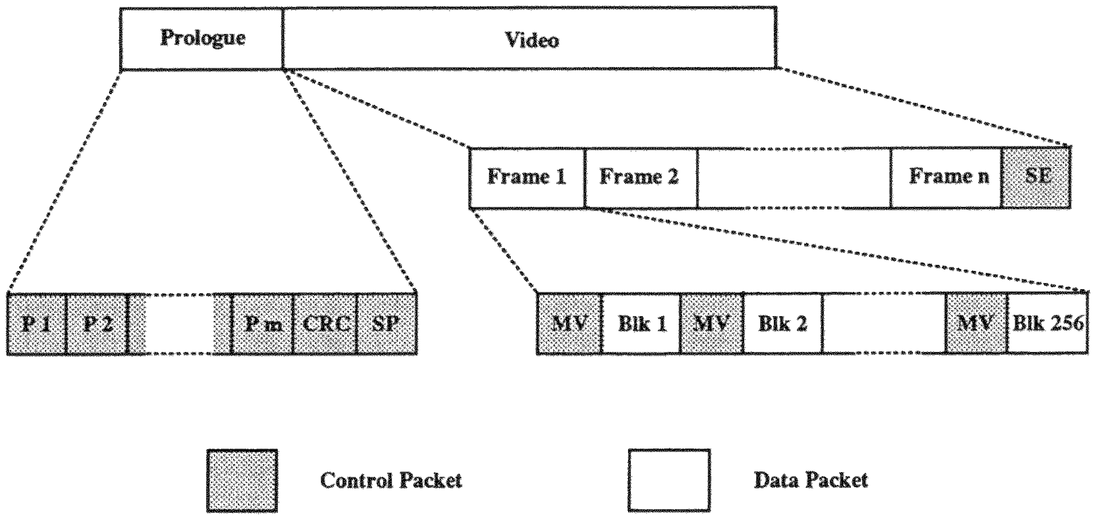


Figure 6.1: Video Stream Structure

6.1.2 Frame Segmentation and Data Packets

Each frame is divided by a factor of 16 in both directions resulting in 256 blocks. As frame size may vary from 128 to 720 excluding HDTV, block sizes are in the range between 8 and 45. This decomposition method has taken into account the requirements of both high- and low-end video streams. At the low-end, a block contains at least 8×8 luminance samples. If the ratio of chrominance to luminance samples is 1:4, the block still carries two chrominance samples per line which make conversions between different dimension or colour systems possible. At the high-end, the maximum block size is less than 7 KBytes (or 30 KBytes if HDTV is included) which sets a reasonable upper limit to a receiver/decoder's buffer size.

For obtaining smooth block boundaries after spatial conversions between mismatched frame sizes, a vertical line of samples to the right of the block and a horizontal line to the bottom can be optionally included. This may extend the above buffer size limit by 4 percent.

A commonly used frame segmentation scheme is to divide a frame into blocks of a fixed size, such as 8×8. The obvious advantage of this approach is that the same block structure can be directly shared by some block based coding techniques. But a vital drawback impedes its use in a general representation scheme: it becomes impossible to perform frame size conversions on the block basis, which is necessary for scalable stream reception, because there are different numbers of blocks per frame.

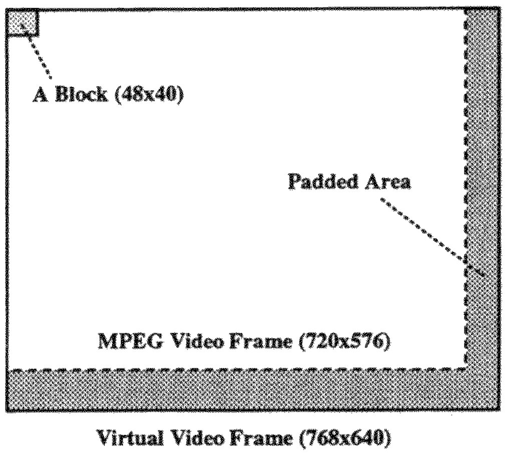


Figure 6.2: MPEG Frame and Virtual Frame

Considering most transform coding is based on blocks of 8×8, *virtual frames* are introduced to support their use on compressed streams. Sizes of such frames are restricted to multiples of 128, i.e. 128, 256, 384, 512, 640, or 768 so that the block sizes in these frames are multiples of 8. Frames of other sizes may be padded with samples of any values in either dimension to form the nearest larger virtual frames if transform or other coding techniques based on 8×8 blocks are used. The example of a MPEG frame and its virtual frame is illustrated in Figure 6.2.

When a receiver processes a virtual frame, it may assume another virtual frame of its own so that the following relationship exists:

$$\frac{D_v}{D_r} = \frac{S_v}{S_r}$$

where D_v and D_r are dimensions of the virtual and real frame sizes, respectively, that are required at the receiving end, and S_v and S_r those that are received. This makes the padded area invisible to end users.

The disadvantage of padding a real frame to a virtual one is the obvious decrease of the bandwidth utilisation[2], but in exchange it allows block-based coding to be performed locally without the presence of the rest of the frame outside of the block. This particular

[2]The worst case comes with a nearly 4 times bandwidth expansion when a 129×129 frame has to be padded to a 256×256 virtual frame. But fortunately, this represents a low-end video stream, which does not consume a large amount of communication bandwidth. Besides, there always exists the other option of not using a virtual frame.

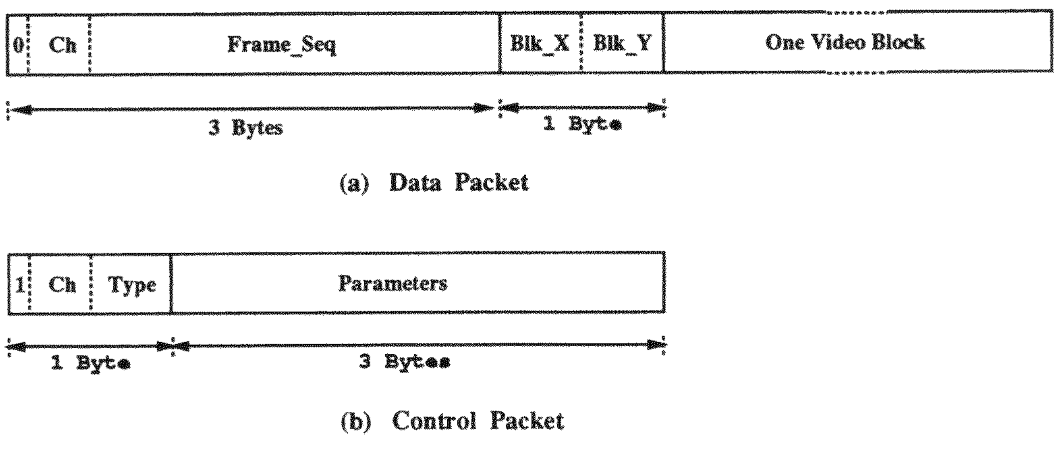


Figure 6.3: Packet Formats

feature enables the design of a scalable decoder which will be explained in more detail later.

The format of a data packet is shown in Figure 6.3 (a). Samples of a block are organised in a given scan order and on component by component basis. The packet starts with a zero bit which distinguishes it from control packets followed by a 3-bit *channel number (Ch)* field which identifies individual video streams for simultaneously receiving/mixing multiple streams by a decoder. The *Frame_Seq* is the sequence number for the frame to which the block belongs. The other fields, *Blk_X* and *Blk_Y* give the horizontal and vertical positions of the block in the frame, starting from #00 for the block at the top left corner.

6.1.3 Coding Description and Control Packets

Most of the basic coding techniques discussed earlier in Chapter 2 can be applied to streams in the general representation format. They are presented by a set of control packets which reside in the prologue part of a video stream. When a decoder receives a stream, it interprets the control packets to find the coding algorithm and other parameters of the stream.

Apart from the control packet identifier bit and the channel number field, which are similar to those in a data packet, a control packet has two other parts as shown in Figure 6.3 (b): the *type* field indicates the type of information provided in the packet followed by the *parameters* field which defines particular values. The complete set of control packets are presented in groups related to coding types.

In the following description, each packet is given with a name (or type) and its encoded parameters in braces. The number in parenthesis following a parameter indicates the number of bits it occupies in the parameters field. A special symbol *any* means that the field is not used and may be filled with any random bits.

6.1.3.1 General Frame Structure

STREAM GENERAL (SG) {
number of components (2);
aspect ratio (8);
frame rate (5);
compressed (1);
predictive coding used (2);
transform coding used (2);
variable length coding used (2);
reserved for extension to other coding methods and local use (2) }

The SG packet provides the basic characteristics of a stream. It is usually the first packet of a video stream, but like other control packets it can also be used anywhere in streams to indicate a change of related parameters. Its fields are self-explanatory except the last four which all have a two-bit space and imply four coding schemes can be defined. They are not fully filled but only the commonly used schemes are each allocated a code, i.e. DPCM and forward motion compensation[3] for predictive coding, DCT for transform coding and Huffman coding for VLC. The rest of the codes are left for future extension or local definitions. A special code, such as #00, in the last field may be reserved to indicate a follow-on packet which can include even more coding types.

FRAME SIZE (FS) {
real or virtual (1);
lines per frame (11);
samples per line (11);
extra lines in blocks (1) }

This packet is used for providing either real or virtual frame size depending on the setting of the first bit. Its last field indicates whether extra vertical and horizontal lines are included in data blocks.

[3]Backward and bidirectional motion compensation is only used in the MPEG system and has excessive requirements on decoders. It is therefore not included.

Picture Depth (PD) {
bits per sample of the first component (5);
bits per sample of the second component (5);
bits per sample of the third component (5);
bits per sample of the fourth component (5);
any (4) }

The packet indicates the number of bits per sample for each colour component. Most colour digital video signals in practice consist of three components. But for some special signal processing purposes, such as graphics superimposing and chroma-keying [Sandbank 90], an additional component may be used. The four-component-field structure of this packet facilitates such applications.

Stream Length (SL) {
number of frames (24) }

As the field is 24-bit long, the length of a video stream is up to 16.7×10^6, or 155 hours at 30 frames/sec. If the field is set to zero, it means that the video length is unknown for some reason. An obvious example is that a live video stream is being distributed.

Colour Conversion Matrix (CCM) {
x position in the matrix (2);
y position in the matrix (2);
precision (4);
element value (16) }

Each packet of this type gives the value of one element in the colour conversion matrix for transforming the component signal to the RGB form. A full matrix for a three-component signal requires nine such packets to convey the information. If a signal is already in the RGB format, only one packet with the precision field set to zero is required.

Stream End (SE) {
any (24) }

As mentioned in the previous section, this marks the end of a video stream, and should always be the last packet.

6.1.3.2 Predictive Coding and Motion Compensation

Block based prediction such as motion compensation may use smaller blocks for better performance. The only requirement is that either dimension of such blocks should exactly divide the corresponding size of the blocks obtained from frame segmentation.

PREDICTION PARAMETERS (PP) {
block size (12);
prediction error precision (5);
length of motion vectors (2);
any (5) }

The above packet has a 2-bit field indicating the motion vector length. This provides four choices of motion tracking range: [-2,1], [-4,3], [-8,7] and [-16,15] in both horizontal and vertical directions. Motion vectors of each type are therefore 4, 6, 8 and 10-bit long, and the following packet contains 6, 4, 3 and 2 vectors, respectively.

MOTION VECTORS (MV) {
vectors (20/24) }

For DPCM, there are no motion vectors, and only the *prediction error precision* field in the PP packet is meaningful.

The PP packet may also be used in the video part of a stream with the *prediction error precision* field set to zero when motion compensation is generally applied on the stream. It indicates the temporal disable of the prediction on the current frame. This is useful to break interframe prediction at a change of scene and to prevent its use over long sequences of frames.

6.1.3.3 Transform Coding

TRANSFORM PARAMETERS (TP) {
block size (8);
zigzag/line scan (1);
coefficient precision (5);
matrix parameters precision (4);
any (6) }

The 8-bit *block size* field allows blocks used for transform coding to be up to 16×16. Since all of the current transform based coding systems use 8×8 DCT, and to avoid unnecessary complexity of the decoder design, it is also considered preferable for streams under this general form to use only 8×8 blocks.

Zigzag scan is a common technique used for arranging the transform coefficients in frequency increasing order so that they can be further coded efficiently. A 1-bit field is therefore provided to identify such a situation.

The *matrix parameters precision* field indicates the actual number of bits occupied by each parameter of the transform matrix which is provided by the following packet.

TRANSFORM MATRIX (TM) {
x position in the matrix (4);
y position in the matrix (4);
element value (16) }

6.1.3.4 Variable Length Coding

VLC/FLC PARAMETERS (VFP) {
length of VLC look-up table (8);
length of escape symbol (3);
escape symbol (8);
FLC length (4);
any (1) }

Practically VLC is often applied in conjunction with FLC to avoid excessively long look-up tables [Liou 90]. An escape symbol is then required to distinguish the two types of codes.

Since 128 is a commonly used length for VLC look-up tables as adopted in the CCITT Recommendation H.261, the maximum length in this scheme is set to 256 to leave some space for flexibility. Longer lengths are seldom used and would lead to inefficient average usage of the table in a general decoder.

The actual entries of a look-up table are carried by the following packet type.

VLC LOOK-UP TABLE (VLUT) {
VLC symbol (8);
FLC equivalent (16) }

6.1.3.5 Subsampling

No specific type of packet is required to indicate that a stream has been subsampled. The process of subsampling has already been implied by the reduced frame size while the original size has no meaning to a receiver or decoder.

6.1.3.6 Miscellaneous

The ability to decode a video stream relies entirely on successful receiving of its prologue part. For this reason, a receiver has to be sure that a received prologue is correct and complete. The CRC packet is designed for such a purpose. It must be the last except the SP packet (if present) in a prologue, and the checksum is run on all packets from the beginning of the stream.

Cyclic Redundancy Check (CRC) {
    checksum (24) }

Since any receiver requires a set-up time to interpret a prologue before it is ready to decode the video stream, the following packet is used to tell the receiver the amount of time, which depends on the complexity of the coding types applied on the stream, between the transmission of the CRC and the first packet of the video part. Transmitters may choose whether to be idle during this period or to send multiple SP packets to fill the gap.

Silent Period (SP) {
    length in milliseconds (20);
    any (4) }

The PM packet is provided for the convenience of applications when manipulating video streams. It may be placed anywhere in a stream for any particular meaning, such as the beginning of a scene, the time stamp on a frame, the start of an editing unit, or any other marking purposes. The information contained in such a packet is entirely up to an application's definition.

Position Mark (PM) {
    application specific information (24) }

6.2 An Example IGVR Stream of Pandora Video

To illustrate how the IGVR scheme may be applied to a real video stream, this section presents an example of its use for the Pandora video.

Pandora is one of the active research systems on integrating digital video communication into computers, which has been briefly introduced in Chapter 2. It employs both normal (256×240) and small (128×120) video formats, and in each case, a subsampling process is applied in both horizontal and vertical directions, resulting in frames for transmission of 128×120 and 64×60, respectively. The only compressive coding scheme employed is DPCM giving 4 bits per sample in a final Pandora stream.

The IGVR example is shown for a normal Pandora video stream. Since its frame has 120 lines after subsampling, a virtual frame of 128×128 may be used. The resulting bandwidth expansion, when such a virtual frame is required, is less than 7%.

The colour transform matrix for a greyscale signal to be converted into its RGB format is similar to that for an YC_rC_b to RGB except that the relevant coefficients for the colour-difference components are removed. In particular, it is given as follows:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.284 \\ 0.107 \\ 4.859 \end{bmatrix} \begin{bmatrix} Y \end{bmatrix} \tag{6.1}$$

where Y is the greyscale component of a video signal.

The prologue part of a sample Pandora video stream is given as follows to exemplify the use of the IGVR scheme. Irrelevant fields of control packets are omitted, which can be filled with any random bits.

STREAM GENERAL (SG) { `/* Normal Pandora video stream */`

number of components = 1;
aspect ratio = 1:1;
frame rate = 25;
compressed = yes;
predictive coding used = DPCM }

FRAME SIZE (FS) { `/* real frame parameters */`

real or virtual = real;
lines per frame = 120;
samples per line = 128;
extra lines in blocks = yes }

FRAME SIZE (FS) { `/* virtual frame parameters */`

real or virtual = virtual;
lines per frame = 128;
samples per line = 128;
extra lines in blocks = yes }

PICTURE DEPTH (PD) { `/* one component only */`

bits per sample of the first component = 4 }

Stream Length (SL) { `/* live video stream */`
number of frames = 0 }

Colour Conversion Matrix (CCM) { `/* the first element */`
x position in the matrix = 1;
y position in the matrix = 1;
precision = 13 (bits);
element value = 1.284 }

Colour Conversion Matrix (CCM) { `/* the second element */`
x position in the matrix = 1;
y position in the matrix = 2;
precision = 13 (bits);
element value = 0.107 }

Colour Conversion Matrix (CCM) { `/* the third element */`
x position in the matrix = 1;
y position in the matrix = 3;
precision = 13 (bits);
element value = 4.859 }

Prediction Parameters (PP) { `/* DPCM parameters */`
block size = 1×1;
prediction error precision = 4 (bits) }

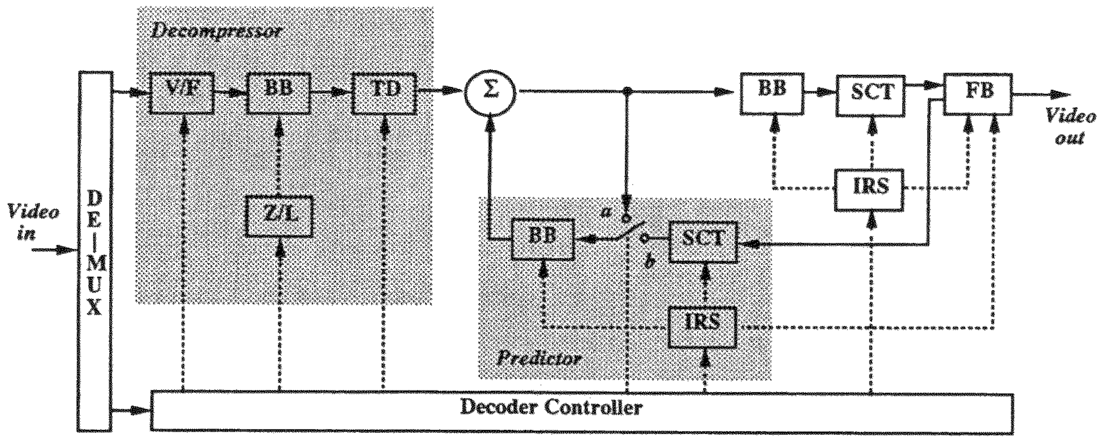
Cyclic Redundancy Check (CRC) { `/* checksum for the previous packets */`
checksum }

Silent Period (SP) { `/* 50 ms interval */`
length in milliseconds = 50 [4] }

6.3 IGVR Decoder Model

Although there are a number of derivatives and combinations of the basic coding types, they are normally applied in a fixed order, i.e. prediction, transformation and variable length coding. This characteristic simplifies the design of a universal decoder model for video streams under this general representation scheme. Furthermore, because of the decomposition method of dividing a frame into 16 blocks in each dimension, such a decoder

[4]This is the minimum value required by the experimental video processing system to be discussed in the next chapter.



BB:	Block Buffer
FB:	Frame Buffer
IRS:	Image Resampling Sequencer
SCT:	Space and Colour Transform
TD:	Transform Decoder
V/F:	Variable/Fixed Length Code Conversion
Z/L:	Zigzag/Line Scan Conversion
....	Control Signals

Figure 6.4: Structure of A Universal Decoder

is scalable to the requirements of the receiving end. The decoder structure is shown in Figure 6.4.

The decompressor consists of function modules for variable length and transform decoding, which can be bypassed if the corresponding scheme is not used to encode the stream. The technology for realising any of these modules with at most a few VLSI chips is already at hand. The implementations of DCT chips have been reported in the literature [Vetterli 86, Jutand 87, Sun 87, Liou 88, Sun 89a], while the zigzag/line scan converter and variable length decoder can be designed with simple logic circuits and *content addressable memory* (CAM) [Sun 89b], respectively.

The predictor performs either DPCM decoding or motion compensation by setting the switch to point *a* or *b* and adjusting the length of its block buffer. The functions of the space/colour transformer and image resampling sequencer are to select and process an area of samples from a source buffer and put the result samples in a second buffer. It is these function blocks embedded in the feedback loop that make the processing in receivers scalable: samples from the frame buffer are transformed back to their original space and colour formats as those of the reference and are therefore ready to be used for predictive decoding. Without such modules, the block buffer in the predictor must be replaced by a frame buffer which is not only much larger but, more importantly, display dependent. Detailed descriptions and implementation will be given in the next chapter.

A hierarchy of decoders can be derived from this general model by simplifying the decompressor and the predictor. To the extreme that both of them are omitted, no bit-rate reducing coding can be processed, but such a decoder is still able to receive video streams in various space and colour formats and has a large number of applications in a multipoint digital video communication environment. This is the subject of the next chapter's experiment.

It is also possible to design a variety of decoders between the two extremes of complexity with one or more function modules omitted. Even a special one for a particular coding system, such as an H.261 decoder, can be implemented for use in an environment where it is the dominating scheme. But the major difference of scalability between it and a normal one still remains.

6.4 Implementation Issues

Although the decoder model given in the previous section is hardware based, the general representation scheme may also be implemented in software. Both designs can be applied in different situations depending on an application's requirement for full or partial real time services.

A full real time service demands that each part of encoding, transmission, and decoding between a transmitter and a receiver should be processed at the speed of the video rate used, while a partial real time service does not. Both have many applications: the former includes videophone and video conferencing and the latter video mail, video archive, and video conference recording.

Software implementations usually have no guarantee for real time performance[5], but they have natural advantages of good maintainability and easy adoption to various computer platforms. Therefore, for the encoding process of video mail and playback of some archived video recoding where low frame rates are used, such schemes might be a cost effective choice.

[5]There has been active research in real time operating systems. But for services running at full video rates, it is still a subject for further study.

For applications requiring full real time services, another hardware approach is to design an *application specific integrated circuit* (ASIC) chip set for the codec. Since most parts of the previously discussed decoder model are based on general purpose VLSI chips which are not the most efficient for such coding and decoding, a set of customised chips can be expected to run much more efficiently and to result in a more compact design.

6.5 Evaluation of the IGVR and its Relation with Other Schemes

The requirements at the presentation level for supporting multipoint digital video communications proposed in Chapter 2 have been the design goals of the IGVR scheme. Such properties are examined in this section together with their availability in other related video stream representation methods.

For a general video representation scheme, scalability is regarded as the most important feature, which has also been pursued in the Open Architecture HDTV model. However, in the open architecture model, it is achieved by exploiting the scalable nature of the pyramid or subband coding while in IGVR it is accomplished by restricting coding operations to relatively fixed, and therefore independent, areas of frames. Both approaches have their advantages and disadvantages. The former requires special transmitters or encoders to have video streams strictly encoded in the particular form, but offers a selection of receivers or decoders with varying receiving qualities. The latter has no restriction on encoders to use one special coding technique so that its streams may take various forms, but its decoders have to be based on a general structure. It is in conjunction with other objectives, discussed in the following list, that the IGVR approach is considered preferable to be used in a multipoint digital video communication environment.

Genericity Most common coding methods can be applied in IGVR by using control packets as "descriptors". Future inclusion of more coding techniques is possible through the reserved extension part.

All the other schemes use predefined coding techniques although parameters may be selected on application basis.

Fidelity Since most common coding techniques are presentable in the IGVR scheme, the conversion of a stream from any other scheme requires few operations which result in minimum loss of quality.

	Quality loss is also small in other schemes due to shared use of DCT, motion compensation and VLC among them and the compatability consideration at their designing stages.
Interchangeability	Decoders based on the general structure model of the IGVR are able to handle any streams coded in a given range of coding types. This feature is irrelevant to other schemes because fixed coding methods are used.
Multiplexing Ability	It is provided by the *Channel Number* bits of each packet in the IGVR. No such facility exists in any other schemes.
Manipulability	This depends on coding types applied on streams in the IGVR. Most compressive coding makes random access impossible, and the granularity of editable units becomes coarse if predictive coding is applied continuously over a long sequence of frames. It is possible in this scheme to choose one or more proper coding types for desired manipulability by an application. At one extreme, uncompressed video streams may be used for gaining full editing ability. The MPEG scheme allows limited ability for manipulating streams. Random accessibility and editing granularity are both restricted to the interval between *I-pictures* in its streams; DVI systems also allow random access to its streams by adjusting coding parameters to maintain a fixed amount of data in each frame. But editing capabilities depend on the compression levels (i.e. edit or presentation level); there is no variable rate coding on Pandora streams so that full manipulability is available on such systems; none of the other schemes provide these editing features.
Efficiency	IGVR streams are less efficient (but only by a small percentage) due to the prologue and headers in data packets. Other schemes have no such overhead except Pandora streams which have segment headers.

Simplicity	IGVR codecs can be easily constructed in software as well as hardware because of the availability of the VLSI chips. More compact design can be achieved by using the ASIC technology. Codec design for other schemes is also simple as the coding algorithm is fixed in each scheme.
Robustness	CRC packets in IGVR streams provide required assurance of whether prologue parts are correctly received. Such a guarantee is not desirable for other schemes as there is no prologue part in those streams.

6.6 Summary

Upon observing the most commonly used coding techniques and digital video representation schemes, this chapter proposes a new scalable stream structure to support multipoint full motion video transmission in networks. It aims to provide an intermediate architecture, through which scalable receiving and easy interchanging of differently formated digital video signals are facilitated.

Various aspects of the IGVR scheme, including stream hierarchy, frame architecture, block organisation, and coding representation are discussed in detail. A Normal Pandora video stream is given to illustrate the use of such a scheme. Other implementation issues of the scheme are investigated, and a general decoder model is provided.

Features supported by the scheme are examined in comparison with other video representation approaches. It shows that all the requirements discussed in Chapter 2 for supporting multipoint digital video communications at the presentation level are well satisfied.

Chapter 7

A Stream Processing System

To experiment with real time conversion and processing of digital video streams and to verify the IGVR scheme proposed in the previous chapter, the *Network Video Stream Transceiver* (NVST) has been designed and built to implement a simple version of the scheme, i.e. with no compression coding applied on streams. This chapter presents the details of the system.

The design objectives and general structure of the system are discussed in the first two sections, followed by a detailed functional description and performance estimation of each of the three major modules: transmitter, receiver, and controller. Experimental results are then provided on various aspects of the system performance, including picture qualities after processing, delays of parameter processing and system setup, and some requirements for the low level network transport facilities.

Several examples of applications for the transceiver system are given at the end of the chapter, followed by the outline of a complete system setup supporting multipoint digital video communications.

7.1 Design Objectives

The main purposes of designing the Network Video Stream Transceiver are to verify the feasibility of the intermediate general representation schemes proposed in Chapter 6 and to experiment with real-time conversions between wide ranging digital video signals. The target parameters of the system are given in Table 7.1.

Maximum frame rate	25
Maximum lines per frame	576
Maximum samples per line	720
Bits per sample	8 — 24
Y:Cr:Cb	Variable
Compression	No
Multiplexing	Up to 4 streams

Table 7.1: NVST Design Objectives

When compared with Tables 2.1 and 2.2, it can be observed that these targets are selected so that the transceiver can handle digital video streams from both the computer and telecommunications worlds with the exception of HDTV since it consumes, at least for the time being, too much transmission bandwidth to be used in any network video applications. The above target range therefore covers any uncompressed video streams viable in a computer network environment.

For multi-point communication purposes, the receiver needs to be able to deal with up to four incoming streams simultaneously, each of which is displayed on a distinct area of the screen. For example, if four streams are being received, each of them will occupy one quadrant of the screen. The number of four is selected to simplify the implementation of the prototype system while the multiplexing ability is still maintained.

7.2 System Configuration

The organisation of the transceiver is shown in Figure 7.1. The transmitter, receiver, and controller are the three major parts of the system, interconnected by data, address, and control buses. These buses are used for command and control signal exchange between the controller and the transmitter or receiver. Video streams take separate paths, going directly through the receiver and transmitter.

The two interfaces shown in outline form are network and host dependent components and are therefore not fully implemented in the experimental system. At the host side, only an RS232 interface and a simple video stream emulator are provided for monitoring the received picture and feeding a test stream to the transmitter, respectively. At the network side, signals from the transmitter are looped back to the receiver bypassing the network interface.

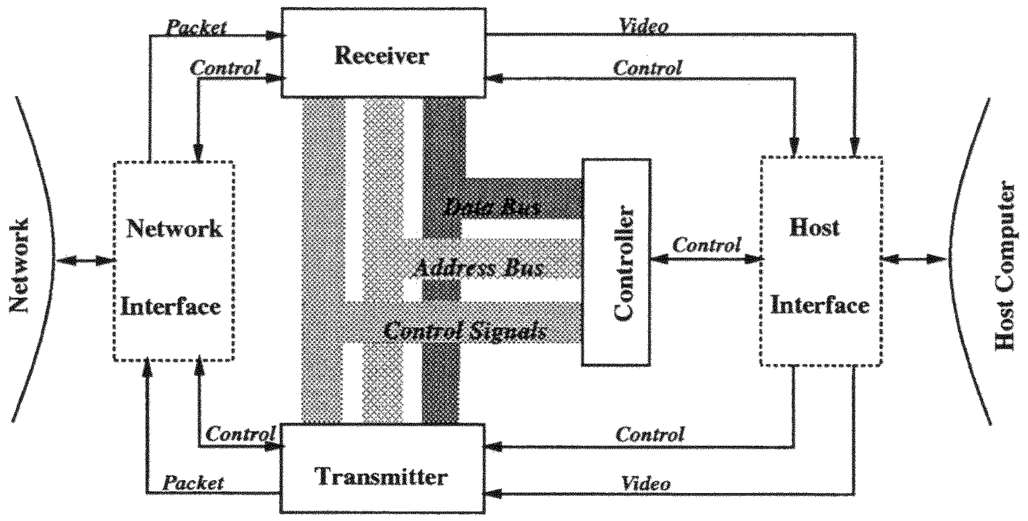


Figure 7.1: NVST System Configuration

7.3 Transmitter

7.3.1 Functional Description

Figure 7.2 shows the organisation of the transmitter part of the system in functional blocks.

Video-RAM (VRAM) is a special kind of *dynamic random access memory* (DRAM) which features an additional *static serial access memory* (SAM) and a separate input/output (I/O) port for accessing it. This structure is particularly suitable for use as a video frame store required by most image processing applications. A whole line of video can be fed into the SAM without interfering with the DRAM. The contents of the frame store, which is updated by a one-cycle data transfer operation from the SAM to the DRAM, are therefore available for processing most of the time.

Two registers are used in the transmitter. The Header Register is used for storing the header information of each video data packet. This four-byte information is prepended to each block of a video frame to form a complete video data packet which is then shifted to the data buffer for transmission. The Packet Size Register, indicates the number of samples in each data packet. Its output is reset to zero when a control packet is being transmitted or waiting for transmission, and therefore, also distinguishes data and command packets.

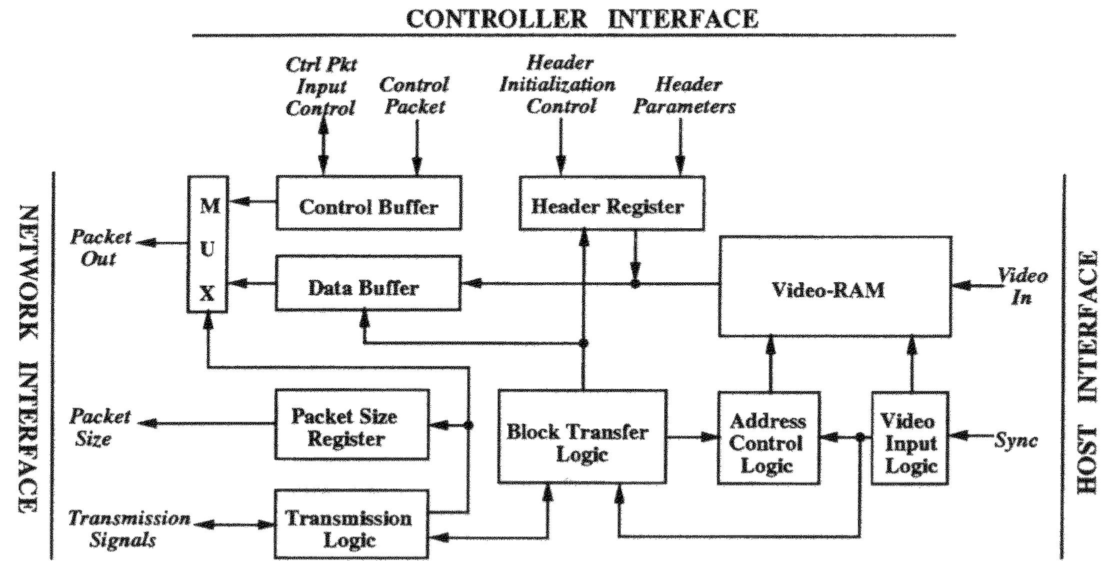


Figure 7.2: Block Diagram of NVST Transmitter

Both the Control and Data Buffers are fast *first-in first-out* (FIFO) dual-port memories with 3-state output buffers which simplify the design of the multiplexor. The Control Buffer can hold up to 128 control packets while the Data Buffer is designed to take a complete data packet from the frame store. It is a function of the Transmission Logic that when both the Control and Data Buffers are not empty the control packets have priority to be transmitted before the data packets.

Because reading and writing FIFOs are much faster than the operations on the VRAM, a new data block starts to move from the VRAM to the data buffer when either the buffer is empty or its contents are being transmitted. This data block move is also scheduled by a clock signal with a frequency of $25 \times 16 \times 16 = 6400$ (*blocks per second*) to ensure that only one block is transferred at one clock cycle so that the video is transmitted at a correct rate. All these functions are provided by the Block Transfer Logic, and the Transmission Logic coordinates the operations of the above function block and the network interface.

The input of a video line into the SAM area of the VRAM is controlled by the Video Input Logic which is also responsible for dumping a newly filled SAM to the DRAM. The DRAM addressing of both this dump operation and block transfer is coordinated by the Address Control Logic.

7.3.2 Performance Analysis

All the control logic and registers are implemented by *TTL (transistor-transistor logic)* devices, and buffers use fast *CMOS (complementary metal oxide semiconductor)* chips which provide a read/write cycle time of 45 ns. However, the speed of DRAM parts of VRAMs are normally much slower than the above devices, and represent the bottleneck of the transmitter.

One of the fastest VRAM chips available at present is used in the system. It provides a minimum normal read cycle of 190 ns and a faster page mode read cycle of only 90 ns. For the biggest target video format, i.e., 720×576 with 25 frames per second, the calculation is given as follows.

```
Block rate                        =  16×16×25
                                  =  6400 blocks/sec
Block cycle time                  =  156.25 μsec

Page mode DRAM read cycle time    =  0.09 μsec
Samples per block                 =  (720/16+1)×(576/16+1)
                                  =  1702
Time for transferring one block   =  1702×0.09
                                  =  153.18 μsec
                                  <  Block cycle time
```

The calculation shows the time requirements are met by the design. The video source emulator used in this experimental system for test purposes generates 256×256 pixels for each frame at the rate of 25 frames per second.

7.4 Controller

7.4.1 Functional Description

The tasks of the controller are the setting up of various control parameters, receiving and transmitting control packets, providing clock signals, and receiving control data from the host. All of these functions are straightforward and are illustrated in Figure 7.3. A dual-port RAM is used as a simple interface for the exchange of control data between the microprocessor and host computer.

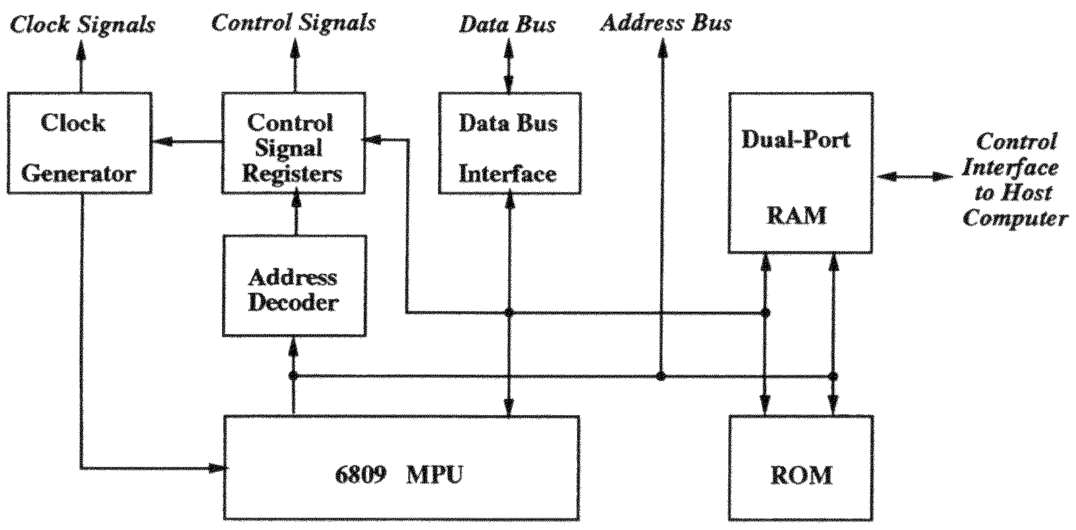


Figure 7.3: Block Diagram of NVST Controller

7.4.2 Performance Analysis

The speed of the controller does not affect the transceiver's performance significantly because it only operates when the status of the system or incoming video stream changes. However, at the beginning of a video stream it receives a large number of control packets and has to perform some calculations in order to derive the control parameters. The calculations usually require much more time than receiving those control packets, and therefore may cause the first few video packets to be lost. If the received video stream is only displayed on a screen, it may not matter because the loss of a few video blocks are generally not noticeable by human users. But if the stream has to be stored, it may be intolerable. To avoid this problem, it is the transmitter's responsibility to insert a small silent interval (of the order of 100 milliseconds) between the header and the data parts of its video stream.

7.5 Receiver

7.5.1 Functional Description

The block organisation of the receiver is shown as Figure 7.4.

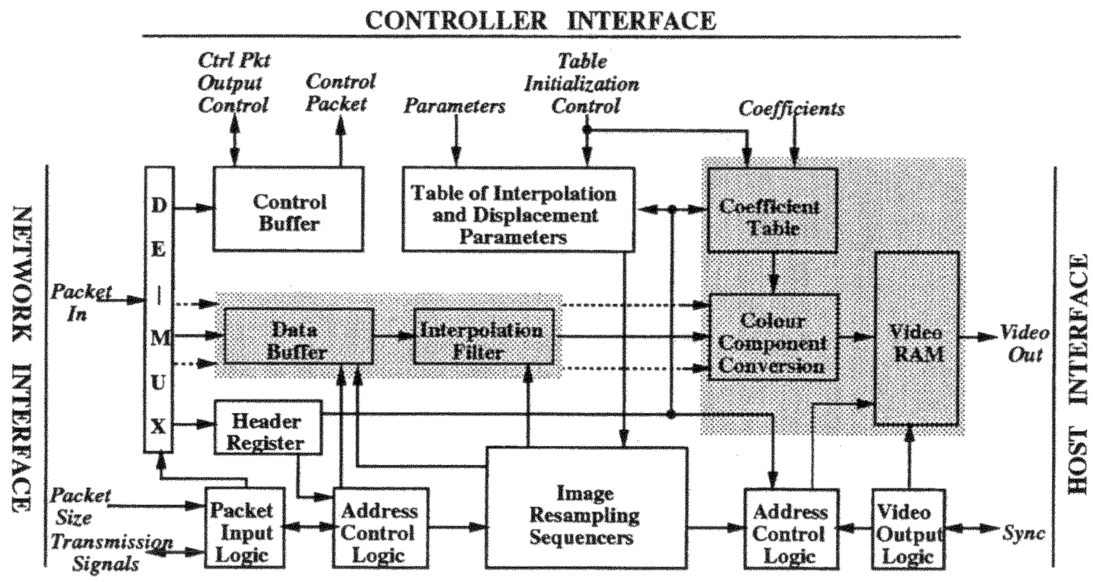


Figure 7.4: Block Diagram of NVST Receiver

According to the value of the *Packet Size* signal, the Packet Input Logic can easily control the demultiplexor module to direct data and control packets to their dedicated buffers. The Control Buffer is a similar FIFO memory to that used in the transmitter, while the data buffer is a dual-port RAM. The RAM capacity is twice as large as the size of the biggest video block so that data-receiving on one port and processing on the other by the Image Resampling Sequencers are performed simultaneously and independently. The receiving and processing parts of the RAM are reversed by flipping the highest address line. The header information of a data packet is needed by a number of the other functional blocks in the system to process the packet, and is stored in the Header Register.

There are two major tasks for uncompressed digital video stream conversions. Firstly, a *space transform* is required if the source and destination frames consist of different numbers of samples either horizontally or vertically. When a sparsely-sampled picture is converted into a more densely-sampled one interpolation is normally used, while for the reverse case, filtering or simply subsampling is a common operation. Secondly, if different kinds of colour formats are used by the two sides, a *colour transform* becomes necessary. A matrix operation is usually employed for this conversion.

The space transform is controlled by a pair of Image Resampling Sequencers (IRS), each of which is a type of *digital signal processing* (DSP) chip specially designed for interpolation and filtering of images. They are driven by an outside parameter table which defines the sizes and places of both the source and destination images or segments of images and how interpolation or filtering should be performed. For each destination sample, the sequencers generate a sequence of source sample addresses to drive the Data Buffer to output corresponding sample values, and clock ticks to drive the Interpolation Filter which consists of a multiplier-accumulator. The interpolation or filtering is therefore the operation of a weighted average of a source sample sequence.

The left-hand shaded part should have two additional duplicates, each of which operates on one colour component of the incoming video stream. This duplication is not implemented in the experimental system for simplicity.

The colour transform is performed by a second multiplier-accumulator which features multiple input ports for samples from different colour components. The coefficients of the transform matrix, stored in the Coefficient Table, are calculated by the Controller based on information given by control packets at the beginning of a video stream and the parameters of the local video display. This (right-hand shaded) part of the system also needs two more duplicates to produce one colour component each if the local video display is full-coloured. This is again not included for simplicity.

The two Address Control Logic blocks perform nearly the same function as that in the Transmitter. The Video Output Logic is also similar to the Video Input Logic except it is used for performing a read operation on the SAM in this case.

7.5.2 Performance Analysis

Akin to the situation in the Transmitter, the control logic, buffers, register, and parameters or coefficients tables are fast TTL devices with a bottleneck arising from the use of VRAM, multiplier-accumulators, and image resampling sequencers. The exact calculation for the Receiver VRAM is the same as that given for the Transmitter, implying that the VRAM part is fast enough for the design targets. The two multiplier-accumulators used as the Interpolation Filter and the Colour Component Conversion can be run at frequencies of 25 and 40 MHz, respectively, and therefore satisfy the time requirements being even faster than the VRAM. The only components which need attention are the Image Resampling Sequencers which have a maximum clock frequency of 18 MHz.

There are several modes of operation for the sequencers based on the number of adjacent source samples involved in an interpolation. This parameter can be chosen in the range from 1 (no interpolation), 4, 9, 16 up to 256. As a compromise between conversion quality and speed as discussed in the following section, a mode which uses one or four samples for each interpolation is employed. It should be noted that both the sizes and the number of incoming video streams are irrelevant to the processing speed of the sequencers. Only the size of the local video display or the frame store, i.e., the number of the destination samples needed to be generated in each frame cycle, is crucial to the processors. The calculation based on the mode using four samples per interpolation is as follows.

```
Maximum clock frequency for the IRS           =  18 MHz
Number of clock cycles for one interpolation  =  4
Time for producing one destination sample     =  4×1/18   =  0.22 μsec
Maximum number of destination samples
produced in a frame cycle                     =  1/0.22×10^6×1/25
                                              =  180000
Number of samples in each frame
for the largest target video stream           =  720×576  =  414720
```

The calculation shows that the speed of the DSP part is more than twice as slow when compared with the highest data rate in the target range of video streams. If the width to height ratio of a normal video screen is about 1.3, the largest video format applicable to the system is 480×375. It is possible in the context of some applications, such as video conferencing, to use a reduced frame rate. For the 720×576 picture format, the frame rate has to be reduced to 10 Hz. But to preserve the design targets given at the beginning of this chapter, more considerations are necessary for the DSP block. Alternative approaches are discussed in the following sub-section.

7.5.3 Alternative Schemes for Processing Large Pictures

The simplest way is to use a *no-interpolation* scheme, i.e. the one sample per interpolation mode. For each sample of the destination frame, its value is duplicated from the nearest neighbour of the corresponding position in the source frame, i.e., $S = S_1$ as shown in (a) of Figure 7.5. Since only one processing cycle is required for each destination sample, this scheme is four times faster than the four samples per interpolation mode, and can be used with a 720×576 picture format.

However, if the four samples per interpolation mode has to be preserved, an alternative approach is to have three DSP blocks running in parallel, each of which is responsible for processing one third of the samples in a destination frame. Figure 7.5(b) shows the process of this *averaging interpolation* which is in analytical form $S = (S_1 + S_2 + S_3 + S_4)/4$. This is an expensive design although it is consistent with the processing of pictures in small formats.

For comparison with the above two approaches, a *linear interpolation* scheme is introduced as a reference. The value of a destination sample is determined by the average of four neighbouring samples in the source frame weighted by the corresponding distances. The

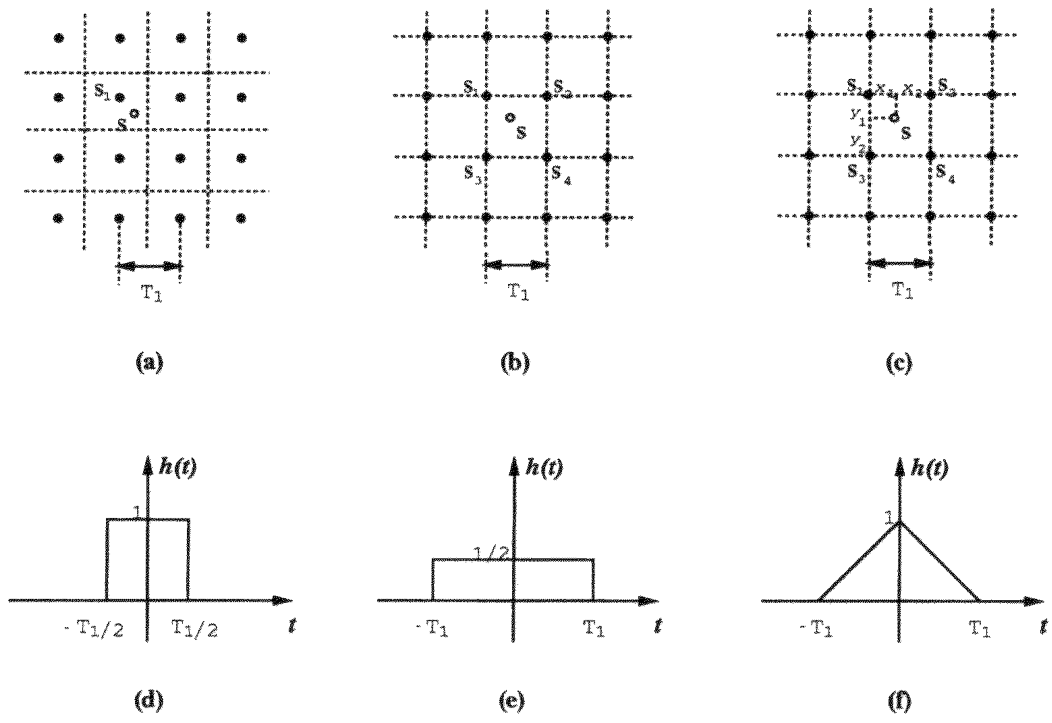


Figure 7.5: Relations between Source and Destination Samples

relation between input and output samples is

$$S = \frac{y_2}{y_1 + y_2}\left[\frac{x_2}{x_1 + x_2}S_1 + \frac{x_1}{x_1 + x_2}S_2\right] + \frac{y_1}{y_1 + y_2}\left[\frac{x_2}{x_1 + x_2}S_3 + \frac{x_1}{x_1 + x_2}S_4\right]$$

as shown in Figure 7.5(c). This scheme is the best of all proposed here but may not be implemented by the DSP chip, because incoming video streams may take any format so that the relative positions between source and destination samples usually depend on locations of samples.

It is noted that two-dimensional interpolation processing of an image can be decomposed as sub-processes in both directions. Because the two one-dimensional sub-processes can be analysed in exactly the same way, the following analysis of the interpolation schemes is given in one dimension only.

An interpolation system is theoretically equivalent to a filter converting its input stream of samples to a continuous signal which is then resampled at a desired new frequency. The *impulse response* of the filter $h(t)$ is determined by the interpolation scheme. For the schemes shown in Figure 7.5(a–c), the functions are depicted in Figure 7.5(d–f), respectively, where $T_1 = 1/f_1$ is the sampling interval of an input stream $x(t)$. The output stream $y(t)$ of an interpolator, sampled at a frequency of $f_2 = 1/T_2$, can be obtained by applying an impulse sequence $\delta_{T_2}(t)$[1] to the output of the filter,

$$y(t) = (x(t) * h(t))\delta_{T_2}(t),$$ [2]

or in the frequency domain,

$$Y(f) = (X(f)H(f)) * \Delta_{T_2}(f).$$

$H(f)$ and $\Delta_{T_2}(f)$ can be found by performing Fourier transforms on $h(t)$ and $\delta_{T_2}(t)$, respectively.

$$\begin{aligned} \Delta_{T_2}(f) &= \int_{-\infty}^{\infty} \delta_{T_2}(t) exp(-2\pi jft)dt \\ &= \frac{1}{T_2}\sum_{n=-\infty}^{\infty} \delta(f - \frac{n}{T_2}) \end{aligned}$$

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t) exp(-2\pi jft)dt \\ &= \begin{cases} \frac{1}{\pi f}\sin(\pi T_1 f) & \textit{no-interpolation scheme} \quad \text{(Figure 7.5(a))} \\ \frac{1}{2\pi f}\sin(2\pi T_1 f) & \textit{interpolation scheme} \quad \text{(Figure 7.5(b))} \\ T_1\frac{\sin^2(\pi T_1 f)}{(\pi T_1 f)^2} & \textit{linear-interpolation scheme} \quad \text{(Figure 7.5(c))} \end{cases} \end{aligned}$$

Hence, the output spectra can be obtained as follows,

$$Y(f) = \int_{-\infty}^{\infty} X(u)H(u)\frac{1}{T_2}\sum_{n=-\infty}^{\infty} \delta\left[(f-u) - \frac{n}{T_2}\right] du$$

[1]An impulse sequence $\delta_{T_2}(t)$ is defined as

$$\delta_{T_2}(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_2),$$

where $\delta(t)$ is the *Unit Impulse Function*,

$$\delta(t) = \begin{cases} \infty & t = 0 \\ 0 & t \neq 0 \end{cases}, \quad \textit{and} \quad \int_{-\infty}^{\infty} \delta(t)dt = 1.$$

[2]The convolution of two functions $f(t)$ and $g(t)$ is defined as $f(t) * g(t) = \int_{-\infty}^{\infty} f(x)g(t-x)dx$.

$$= \begin{cases} \frac{T_1}{T_2}\sum_{n=-\infty}^{\infty} X(f-\frac{n}{T_2})\frac{\sin[\pi T_1(f-n/T_2)]}{\pi T_1(f-n/T_2)} & \textit{no-interpolation scheme} \\ \frac{T_1}{T_2}\sum_{n=-\infty}^{\infty} X(f-\frac{n}{T_2})\frac{\sin[2\pi T_1(f-n/T_2)]}{2\pi T_1(f-n/T_2)} & \textit{interpolation scheme} \\ \frac{T_1}{T_2}\sum_{n=-\infty}^{\infty} X(f-\frac{n}{T_2})\frac{\sin^2[\pi T_1(f-n/T_2)]}{[\pi T_1(f-n/T_2)]^2} & \textit{linear-interpolation scheme} \end{cases}$$

It is found that a picture format smaller than 128×128, where a person's face is hardly recognisable, is useless for most applications. Therefore, the ratio of output to input sampling frequencies f_2/f_1 will not exceed 720/128=5.625. Supposing the input stream has a trapezoidal spectrum, the output spectrum $Y(f)$ of each of the above schemes is shown in Figure 7.6 for $f_2/f_1 = 2, 3, 4$, and 5. Note that in Figure 7.6(a) the curves of the no-interpolation and linear-interpolation schemes coincide with each other. The figure shows that in each case the performance of the no-interpolation scheme, for the frequency space of interest $[-f_1, f_1]$, is closer to that of the linear-interpolation scheme, and when f_2/f_1 increases the schemes converge.

The conclusion of the above spectrum analysis is that if there are not enough redundant source samples to be averaged to produce each destination sample, the no-interpolation scheme is preferable to the averaging one. The receiver is therefore designed to generate destination samples by simply reproducing neighbouring samples of the source frame if its size is less than twice of the destination frame in either direction. This guarantees the DSP component has enough processing time for any conversion.

7.6 Experimental Results

7.6.1 Grey Shading Test Patterns

A grey shading test pattern generator has been built to feed test signals to the transmitter. The sample values of each scan line are `#0F` at the beginning, and decreased by 1 to the next position until `#00` for the 16th sample where the whole pattern is repeated for the rest of the line. Upon this basic 16-sample pattern, each pair of samples which are 8 samples apart can be set to zero to provide white stripes in a test signal. Three examples are given as follows.

`FEDCBA9876543210`	basic grey shading (Pattern 1)
`0EDC0A9006540210`	grey shading with white stripes (Pattern 2)
`F000BA0070003200`	grey shading with wider white stripes (Pattern 3)

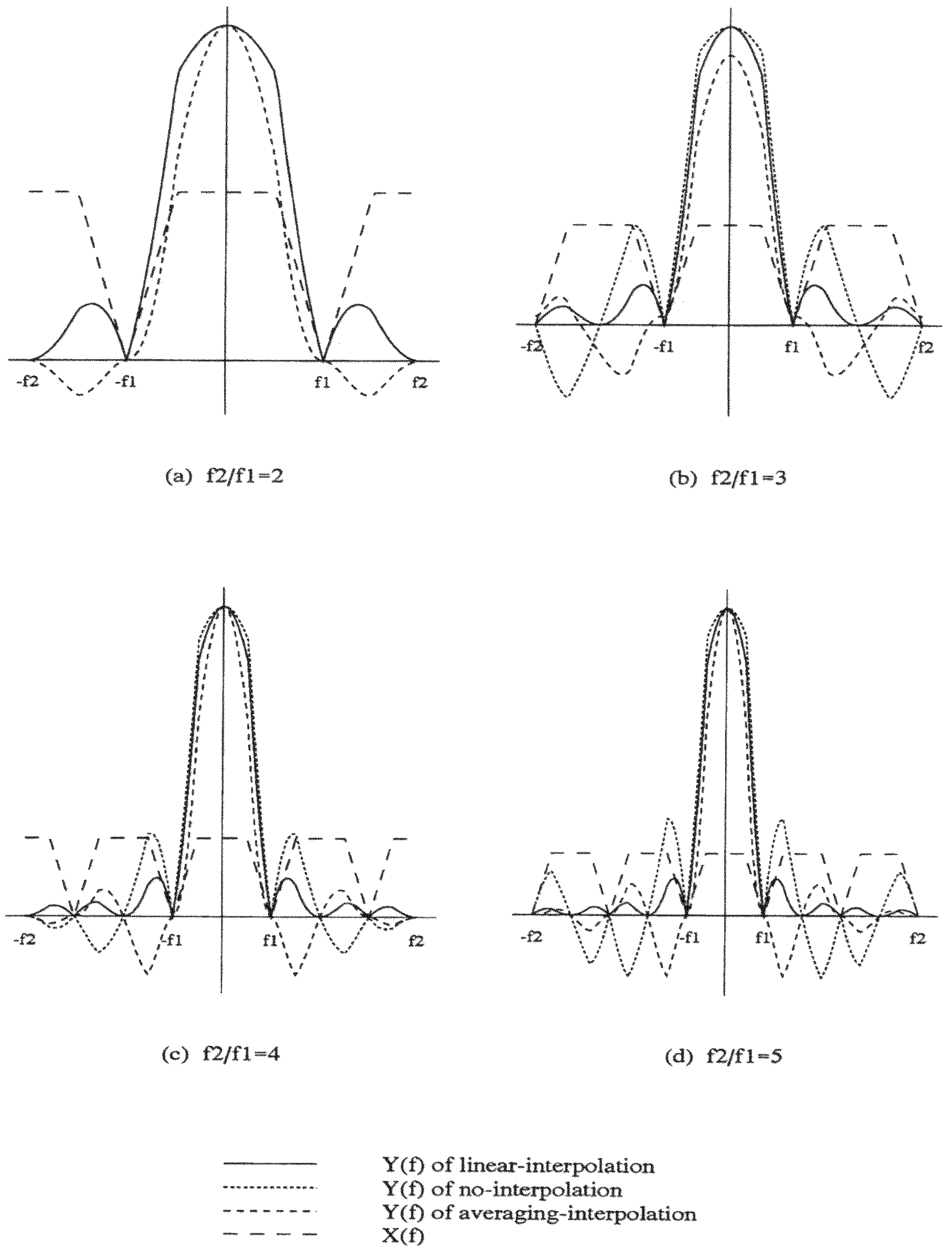


Figure 7.6: Frequency Spectra of Input and Output Streams

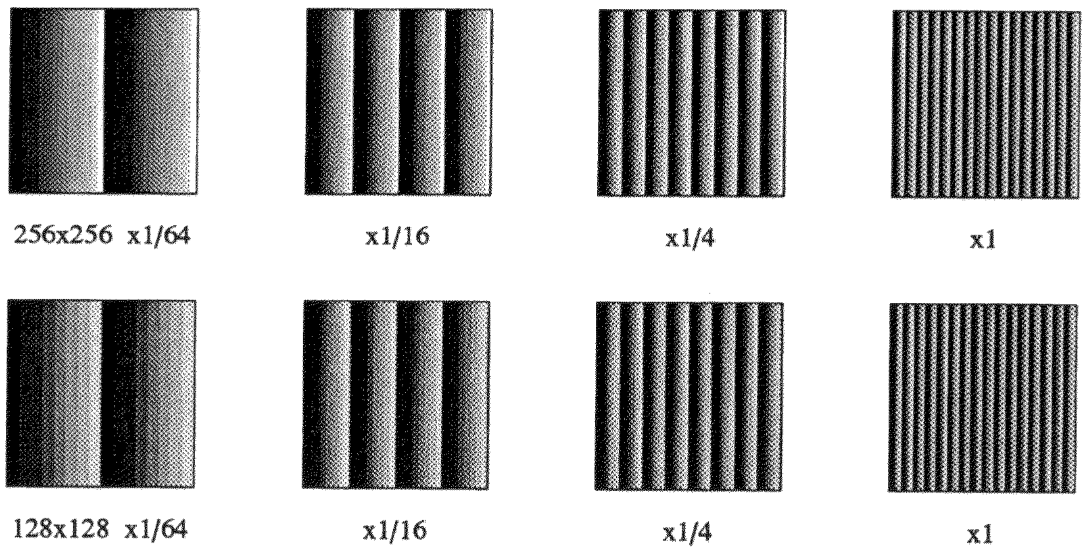


Figure 7.7: Test Pattern 1 — Basic Grey Shading

Pictures of these test patterns, generated at 25 frames per second with frame size of 256×256, are displayed on the top lines of Figures 7.7–7.9 with different scales of enlargement for quality comparison with the relevant received pictures. It should be noted that picture quality is degraded in proportion to the quantity of information which is lost in the manipulation of the original picture to satisfy the frame size at the receiver. For this reason, the results are presented on a destination screen of 128×128.

The bottom lines of pictures in Figures 7.7–7.9 illustrate the quality of received pictures in various enlargements in comparison to the corresponding original test patterns on the top lines.

It is found that for a gradually changing signal, i.e. containing a small amount of high frequencies as shown in Figure 7.7, the quality of the received signal is very high. The difference between the original signal and the converted one can only be seen when a picture is considerably enlarged to expose its details.

Test Pattern 2 in Figure 7.8 has some random white stripes, which introduce high frequencies at their edges. This is the worst situation for the system. The result shows that the received picture is blurred as the high frequencies are filtered out. But it can also be seen that the picture as a whole is still comparable to the original, and therefore, should be acceptable to most applications.

For signals containing even wider white stripes as in Test Pattern 3 in Figure 7.9, the result gets much better again. Although the filtering effect still exists in this case, the

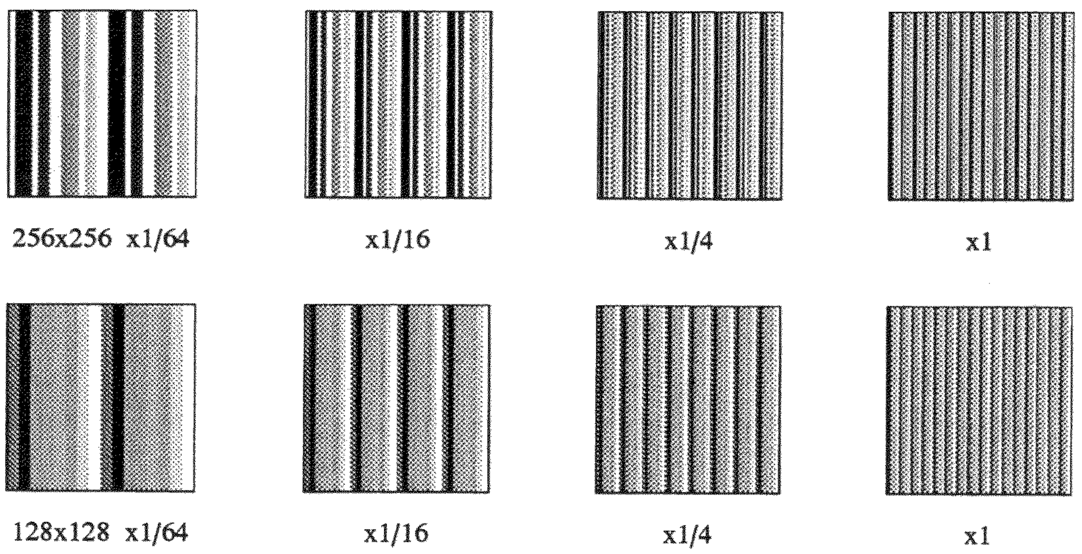


Figure 7.8: Test Pattern 2 — Grey Shading with White Stripes

same pattern is preserved which can be observed in the enlarged pictures.

In general the results from receiving the test pattern signals show that the video stream transceiver, using only simple interpolation schemes, is able to provide pictures of reasonable quality in the situations or applications targeted in its design objectives.

7.6.2 A Real Picture

Although the grey shading pattern tests demonstrate the performance of the system in technical details, they do not provide any practical visual presentation of real picture qualities actually perceived by the human eye. A 256×256 picture of a beach scene is therefore used as an additional test sample.

Since the test signal generator designed for the prototype system is only able to feed test patterns into the transmitter, this part of the experiment uses a computer to perform the interpolation algorithms on the picture.

The results are again presented for the case of converting the original 256×256 picture to 128×128 only, because this is the worst case for system performance as discussed above.

The original picture is given in Figure 7.10. The converted pictures, by averaging- and no-interpolation schemes, are presented in Figure 7.11 (a) and (b), respectively. Because the number of source samples is twice that of the converted ones in either horizontal or vertical

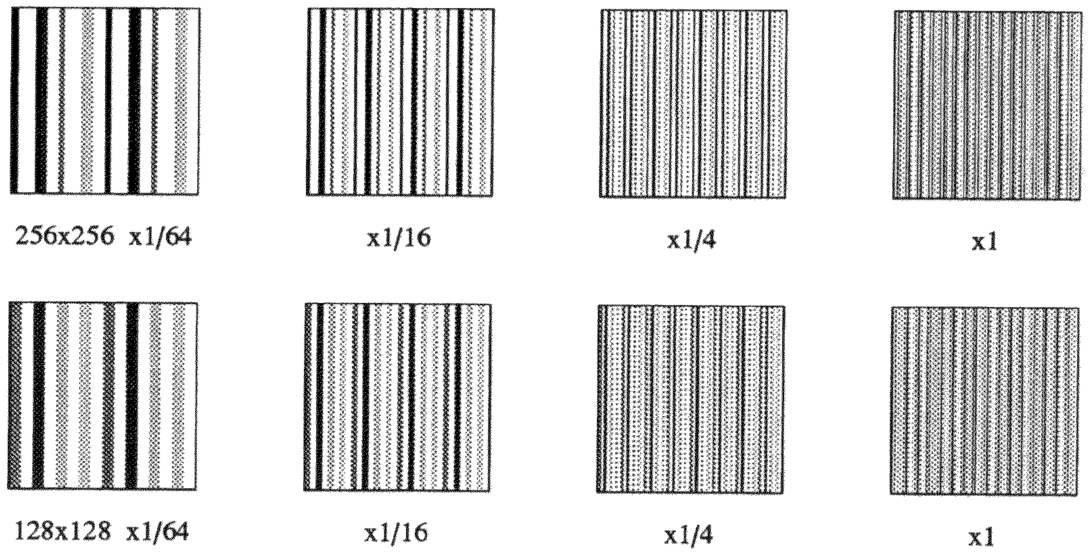


Figure 7.9: Test Pattern 3 — Grey Shading with Wider White Stripes

directions, the average-interpolation scheme is actually used in the system. The result for the no-interpolation scheme is only produced for comparison. It can be seen from the pictures of Figure 7.11 that (a) preserves more information from the original picture than (b) does, (notice that one leg of the person at the left-hand side has disappeared in (b)), while (b) obviously has a sharper, but to some viewers perhaps, unnatural appearance.

This test shows that even with a very small size of 128×128, a converted picture may still maintain a reasonable quality, and by choosing a proper but still very simple (averaging-interpolation) scheme the picture, after a large amount of reduction, is not significantly worse than the original and should be acceptable for the target video applications.

7.6.3 Delay and Jitter

There are two kinds of delays involved in the video stream transmission system. Firstly, processing delay is incurred by the controller in receiving control packets, calculating various control parameters, and setting up or adjusting the system for particular streams. This sort of delay mainly occurs at the beginning of a stream when the system has to calculate all of the parameters from the stream header. Secondly, a transmission delay exists in communications networks. This does not affect the system significantly if the delay remains relatively constant and small so that the interactivity required by some video applications can be maintained normally. However, if the delay is variable, also known as jitter, its effect needs to be considered on related parts of the system.

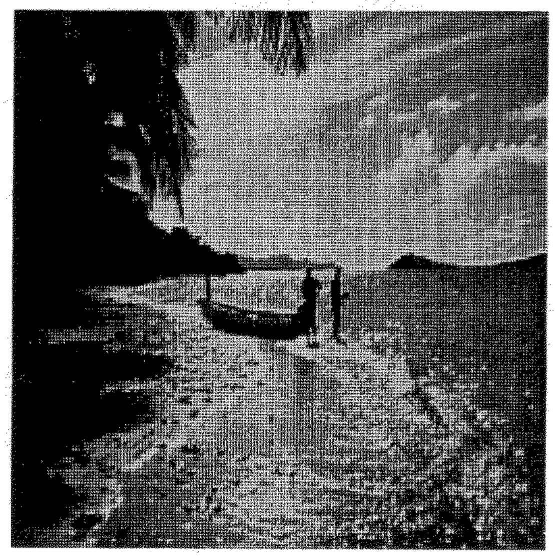


Figure 7.10: Original Picture of 256×256 samples

Table 7.2 gives delays of various operations performed by a receiver when receiving a new stream. The dominant factor is the time spent on initialising the space transform parameters table since there are sixteen pairs of 12-bit parameters for each of the 16×16 block locations in a video frame. The total delay of 50 msec is the reason for demanding a 100 msec silent interval between the header and data parts of a video stream, which also provides a safety margin for any possible transmission and queueing delays incurred by the header over networks.

Since the transceiver uses a low-end 8-bit microprocessor (MC6809), and the code for experimentation is not optimised, the above delays can be regarded as the upper limits for such network transmission systems. But even this 50 msec delay fits very well into the proposed requirements for real-time video services, i.e. a maximum setup time of 15 seconds [Ferrari 90].

Jitter due to network transmission has a number of consequences for the system. Firstly, if the header experiences a much longer delay than the first video block, so that the inserted silent interval is reduced to less than 50 msec, there may not be enough time for a receiver to be ready to receive the stream. Secondly, jitter on consecutive video blocks will not cause any congestion in a receiver because the incoming packets are double-buffered and the processing is performed at the same speed as packet-receiving. But the frame refreshing rate of the affected receiver may become abnormal in this case.

The jitter problem cannot be solved by the transceiver itself. It is one of the requirements for the low level transport facilities of networks carrying video services to provide a proper

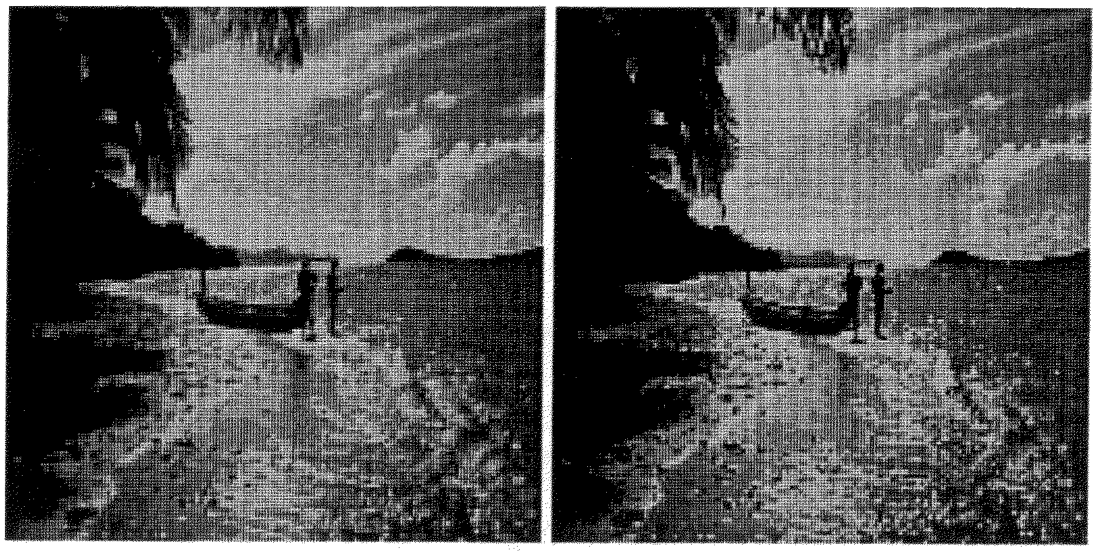

(a) --- by averaging-interpolation scheme
(b) --- by no-interpolation scheme

Figure 7.11: Received Pictures of 128×128 samples

	Operation	Delay (msec)
Basic stream header[a]	Reception	0.68
Space transform parameters	Calculation	0.60
	Setup	45.47
Colour transform parameters	Calculation	3.06
	Setup	0.07
Total		50

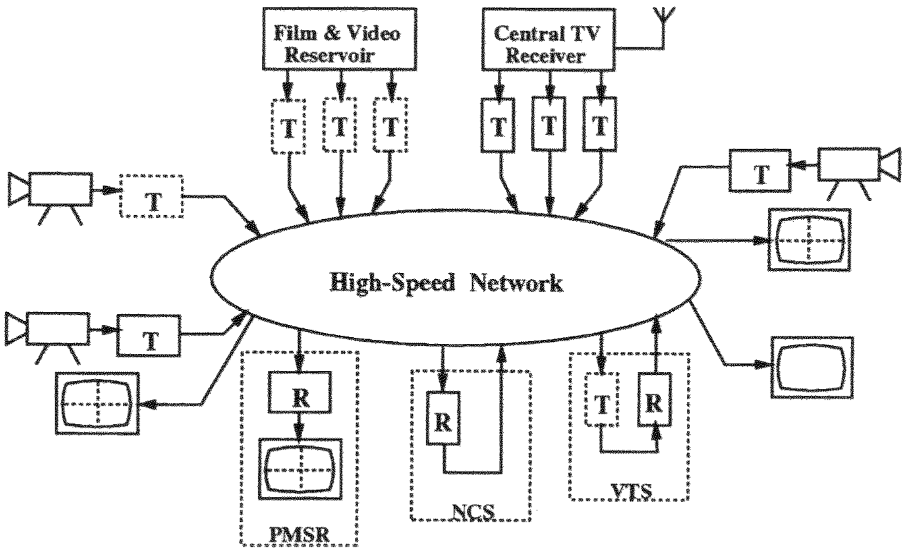
[a]A basic stream header has no more than 20 control packets including information on colour components, frame rate, scan lines per frame, samples per line, bits per sample, and a colour transform matrix for the conversion to RGB format.

Table 7.2: Delays for Setting-up of a Video Channel

flow control mechanism. If a maximum jitter of 5 msec [Ferrari 90] is guaranteed by such a mechanism, all of the above effects will be eliminated.

7.6.4 Sequencing and Duplication

The requirements of sequencing and absence of duplication depend on whether a stream is to be only displayed or to be stored. When displayed on a screen, the degradation of picture quality may hardly be noticed if the delay and jitter requirements are already met



PMSR ········· Personal Multi-Stream Receiver			
NCS ········· Network Conference Server		T ········· NVST Transmitter	
VTS ········· Video Translation Server		R ········· NVST Receiver	

Figure 7.12: Applications of the Network Video Stream Transceiver Systems

by transport networks. But in the case that a stream needs to be stored, neither out of order nor duplicated blocks are tolerable, and extra requirements of sequenced delivery and no-duplication should be enforced on low level transport facilities.

7.7 Applications

Due to the modular design of the transceiver system, it is possible to use the transmitter and receiver parts separately or in a different configuration. This may generate a number of interesting applications of the system in a networked environment in addition to its intended main use for desktop video conferencing. Some of them are depicted in the scenario of Figure 7.12, and will be briefly discussed in the following sections.

7.7.1 Personal Multi-Stream Receiver

A personal multi-stream receiver consists of an NVST receiver attached to a normal television or any video monitor. It is able to receive multiple video streams from a number of sources ranging from cameras to a central reservoir of movies and video or a central station for receiving television programs. All these sources can be equipped with one or more NVST transmitters so that the streams sent over networks are in a commonly understood form. Multicast facilities discussed in Chapters 4 and 5 may be provided by network transport mechanisms if there are multiple receivers of a single television or video program to save high-demand network bandwidth and ease congestion problems.

7.7.2 Network Conference Server

For some desktop video conferences where customised screen arrangement is not required and all of the displays are of the same type, a centralised network conference server may be used to eliminate the need of an NVST receiver at each participating site. This will both reduce the bulky conferencing equipment on user's desks and make it more cost effective.

The server uses an NVST receiver to collect all of the streams from the network and mix them in its local frame store. The mixed signal is then sent back to the network in its raw video format suitable for the displays. Again, a multicast scheme can save a lot of network bandwidth and avoid congestion.

7.7.3 Video Translation Server

When a stored video program needs to be viewed on an incompatible screen, a video translation server is required. It can be constructed by reversing the transmitter and receiver parts of an NVST system and looping back the output of the transmitter to the receiver. Both incoming and outgoing streams are in their raw, but different, video forms. If a source is equipped with an NVST transmitter, i.e. it sends its video stream in the common format, the transmitter part of the server is by-passed. This situation is indicated by the dotted transmitter boxes in Figure 7.12.

A problem currently associated with various experiments on real-time video applications is that specially designed hardware equipment is not widely available because of its high cost. Although an experimental video environment is usually attractive, it would be more convenient to the majority of users to have easy access to the new applications. As windowing systems, such as *X Windows* [Mansfield 90], have nearly become a standard feature on most modern workstations, it may be possible to have such a video translation server to convert a hardware specific video stream to a sequence of *X Pixel Maps*. In this way, the original picture quality may be degraded to some extent because there is no real-time guarantee in the standard X protocol, but it is still a helpful compromise to the above problem since it will at least provide normal workstations some simple but useful operations, such as "browsing" or "previewing" a video message. It can also help to gather more information on user reactions to the new communication medium.

7.8 An Organisation for Multipoint Video Communications

A complete configuration for multipoint digital video communications can be setup when stream multicast path finding algorithms, presented early in this dissertation, and intermediate stream representation schemes operate together. This is generally illustrated in Figure 7.13.

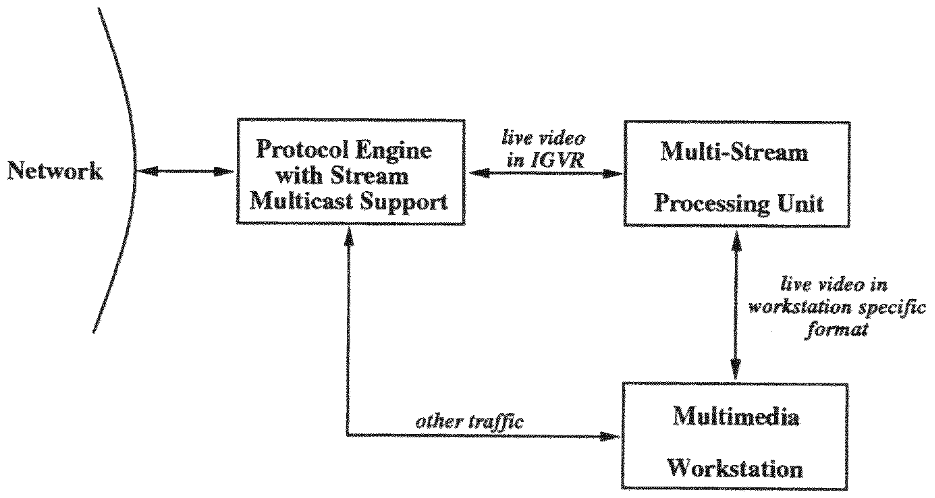

Network
Protocol Engine with Stream Multicast Support
live video in IGVR
Multi-Stream Processing Unit
live video in workstation specific format
other traffic
Multimedia Workstation

Figure 7.13: A Configuration for Multipoint Digital Video Communications

As discussed in Chapters 4 and 5, both distributed and centralised stream multicast path finding algorithms can be easily implemented in a network protocol engine. Such a network interface with stream multicast support could significantly relieve the congestion problem.

Video streams in IGVR can take a separate path from other traffic to go through a multi-stream processing unit which may be either an NVST system or a more complete version providing the full functionality of the IGVR. Trade-offs among software, hardware and ASIC implementations of the processing unit can be made on the application basis, as suggested in Section 6.4. Cost effective approaches can therefore be provided to individual application environments.

Streams received by the workstation have been converted to the machine specific format, and may be mixed according to the requirements of an end user. Through appropriate video facilities of the multimedia workstation, a network video conference and other multipoint digital video services can be provided.

7.9 Summary

The experimental network video stream transceiver system is described in detail in this chapter. Design choices on the interpolation schemes for processing different video streams are based on the frequency spectrum analysis. Performance is first evaluated by calculations of the time requirements on the slowest parts of the system, then further checked by tests on various kinds of signals.

The experimental results show that the system performance fits well to the design targets given in the beginning of the chapter. It is also seen that although the tests only reflect the low performance limit of such systems because of the simple processor used, it is satisfactory enough for the targeted application area.

A few examples of additional applications of the system are briefly given at the end of the chapter. It is not difficult to extend the application list by re-organising modules of the system in various ways.

A system configuration illustrated at the end of the chapter shows how stream multicast path finding algorithms and intermediate representation schemes can operate together to provide multipoint digital video communication services.

Chapter 8

Conclusion

8.1 Conclusion

This dissertation has presented a proposal for the architectural support of multipoint digital video communications at two different layers. Stream multicast path finding and intermediate general stream representation are regarded as two important issues in a multipoint video communication environment.

The survey of digital video signals shows that at the presentation level, a wide variety of digital video representation formats have emerged. This causes difficulties for multipoint digital video communications, such as easy interchange of differently formatted signals and scalable receiving of signals with respect to receiver specific requirements.

In addition, current multicast path finding algorithms at the network level are not satisfactory for use in a multipoint broadband communication environment. The major problem is that at present the bandwidth occupancy of a digital video signal is comparable to the total capacity of an average network link. The result is that a single path often lacks the required bandwidth for a multicast group which generates multiple digital video streams simultaneously, referred to as stream multicast.

Simple bandwidth checking mechanisms, either using a single-path or multi-path approach, may partially solve the bandwidth problem for some special cases, such as in a network where link capacity is not a scarce resource. But in other situations, these approaches lead to a high percentage of stream multicast path finding failures.

By proper integration of link capacities throughout a stream multicast path finding process, algorithm failure rates can be reduced to a significantly low level, while a comparable distance-oriented multicast path cost is still maintained. Such integration methods were discussed in detail for both centralised and distributed stream multicast path finding algorithms in the dissertation.

The fact that conventional multicast path finding algorithms have already been well researched makes it possible that new algorithms dealing with stream multicast can be based on modifications of the traditional ones with link capacity integration. Different ways for this basic approach were investigated in the dissertation, and it has been found that the relative weights in consideration of link distance and capacity factors affect the performance of a stream multicast path finding algorithm with respect to multicast path cost and the effect of such a path on the general traffic in a network.

The situation of distributed multicast path finding algorithms is rather different, as only a few of them have been proposed and studied. However, similar link capacity integration methods to those used for the centralised algorithms are still applicable to the distributed ones. They were again investigated with a new distributed stream multicast path finding algorithm proposed on the basis of a packet exchanging mechanism. The relation of link distance and capacity considerations with the resulting multicast path cost and its effect on network traffic is similar to that in the centralised case. Therefore, applications, whether on a distributed or centralised basis, can choose proper link capacity integration mechanisms for their stream multicast path finding processes based on such results.

Although costs of network links and paths in conventional multicast path finding algorithms are presumed to be based on link distances, it is clear that such a measurement can be viewed as an abstract cost in both conventional and stream multicast path finding algorithms. In this sense, the methods discussed for the integration of link capacity are of general interest to the path finding issue in multipoint digital video communication applications.

There is a related problem to conventional multicast path finding algorithms of altering a multicast tree due to the joining or leaving of a group member. In stream multicast cases, this is considered as a separate issue, and therefore, was not discussed in the dissertation. The main reason for this is that a joining or leaving operation involves a change of bandwidth requirement from a multicast group.

While a leaving operation may be performed simply by releasing the related bandwidth and paths on multicast trees, a joining operation is significantly different from its conventional multicast case due to a bandwidth increment. To handle the extra bandwidth required, either a new tree or an existing tree providing such bandwidth has to be found, and a branch connecting the new member to each of the existing trees is also needed.

Alternatively, the old connection may be completely torn down, and a fresh path finding process for the updated bandwidth requirements restarted from scratch.

Neither of the above approaches represents significantly new issues with respect to stream multicast path finding. In the case of the former approach, a new tree can be searched by a stream multicast path finding algorithm while new branches of existing trees may be built with any conventional join mechanism used in usual multicast cases. The latter case simply means a repetition of the stream multicast path finding process with new bandwidth requirements.

The signal interchanging and scalable receiving issues at the presentation level have not been thoroughly studied in the past, but are regarded as necessary for supporting multipoint digital video communications over a heterogeneous networking environment. The intermediate general video representation scheme proposed in the dissertation is among the early attempts to solve such problems.

The approach is rather simple, but effective. The discussions on its various design decisions, such as the stream hierarchy, frame structure, block organisation, coding description, and general decoder architecture, show that the scheme can be effectively used by multipoint digital video communication applications.

The experimental video stream processing system demonstrates the feasibility of such a video representation scheme. With higher speed VLSI signal processors becoming available and field programmable logic cell arrays widely applied, it is possible to design a more powerful yet compact version of the system. The new hardware devices will also facilitate a more elaborate video stream representation scheme on a more refined basis. One such example is that a special purpose description language can be used to present the decoding process for a digital video stream in detail, and can be either interpreted or complied into the hardware of a receiver.

The system outline given at the end of the previous chapter has illustrated how stream multicast path finding algorithms and intermediate signal representation schemes can work together with a general purpose protocol engine. The observation can be made that the architecture can support a new range of multipoint broadband communication services, and is flexible to suit individual application environments since trade-offs can be made among various implementations in software, hardware and ASIC technology.

In general, when applications of multipoint digital video communications become increasingly desirable with higher network bandwidth availability and faster signal processing technology, the architectural support issues discussed in this dissertation are among those having an imperative necessity to be addressed. The approach taken by this research provides reasonable results to show how such problems may be tackled.

8.2 Future Work

Further work may be pursued in the following directions:

- From the results on the performance of different link capacity integration methods for both centralised and distributed stream multicast path finding algorithms, it is possible to design a versatile path finding algorithm which suits a wider range of application environments. In such a scheme, the new costs which determine the selection of a link as a branch of a tree may be a function of the original (or "real") link cost, link capacity, the average network capacity, the deviation of network capacities, and the required stream multicast bandwidth. This may maintain a proper compromise among path finding failure rate, multicast cost, and other impacts of a multicast path on the network.
- A link overload avoidance or resolution mechanism may be designed to investigate a concurrent algorithm for distributed stream multicast path finding. Comparison of its performance can then be made with that of the sequential algorithms given in the dissertation on a quantitative basis.
- As mentioned earlier, joining or leaving operation for stream multicast cases is a separate issue, and is worth a thorough investigation. A dynamic tree growing and optimisation procedure is another good research topic to be looked at.
- Field programmable logic cell arrays have a fast dynamically reconfigurable structure which can be easily organised into a special piece of hardware within tens of milliseconds. This feature makes them particularly suitable for the implementation of a general video stream decoder. The most serious drawback at present is that a compilation process from a high level description down to a cell interconnection pattern is time consuming. Therefore, before the hardware compilation problem is solved, a temporary compromise may be to provide a library of basic building blocks for various coding techniques, from which a real time hardware linker can be made.
- Another related subject to the above issue is the design of a proper hardware description language specially for presenting video coding. It may not necessarily be a sophisticated one, but should be tuned to facilitate the efficiency of its compilers or linkers.
- Transcoding between compressive coding schemes, such as between various international standards, based on the intermediate representation scheme is also an interesting issue. The follow-up of the experimental system discussed in the dissertation can be a full implementation of the IGVR scheme.

- Although software implementations tend to be much slower than their hardware versions in most applications, it will be interesting to find out the performance of IGVR decoders implemented in software over various computing platforms, and their usefulness for both time critical and non-critical applications.
- Finally, a thorough investigation into an integrated system architecture for supporting multipoint video communications, as discussed in the previous chapter, may be considered. The design of a general purpose protocol engine incorporating stream multicast mechanisms and its interaction with stream processing facilities are interesting research subjects.

In addition to the above list, each of the application examples given through various discussions in the dissertation is also worth further investigation or experimentation.

References

[Aho 83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, Massachusetts, 1983.

[Ang 91a] C. S. Ang, *Continuous Media in Fast Networks*, PhD thesis, University of Cambridge Computer Laboratory, Cambridge, UK, September 1991.

[Ang 91b] P. H. Ang, P. A. Ruetz, and D. Auld, "Video compression makes big gains", *IEEE Spectrum*, 28(10):16–19, October 1991.

[Attar 81] R. Attar, "A distributed adaptive multi-path routing — consistent and conflicting decision making", In *Proceedings of the 5th Berkeley Workshop on Distributed Data Management and Computer Networks*, pages 217–236, February 1981.

[Bellman 57] R. E. Bellman, *Dynamic Progamming*, Princeton University Press, Princeton, New Jersey, 1957.

[Benson 86] K. B. Benson (editor), *Television Engineering Handbook*, McGraw-Hill, New York, 1986.

[Boggs 80] D. R. Boggs, J. F. Shoch, E. A. Taft, and R. M. Metcalfe, "Pup: an internetwork architecture", *IEEE Transactions on Communications*, COM-28(4):612–624, April 1980.

[Bove 91] V. M. Bove and A. Lippman, "Open architecture television", In *Proceedings of the 25th Conference on Television*, New York, February 1991.

[Breslau 91] L. Breslau, D. Estrin, and L. Zhang, "Adaptive, multipath source routing for large-scale networks", Technical Report TR 91-06, Dept. of Computer Science, Univ. of Southern California, Los Angeles, California, February 1991.

[CCITT 90] CCITT, "Video codec for audiovisual services at $p \times 64$ kbit/s", CCITT Recommendation H.261, Geneva, 1990.

[Chesson 91] G. Chesson, "The evolution of XTP", In O. Spaniol and A. Danthine (editors), *High Speed Networking III*, pages 15–24. North-Holland, Amsterdam, 1991.

[CIP 90] CIP Working Group, "Experimental Internet Stream Protocol, Version 2 (ST-II)", RFC 1190, C. Topolcic (editor), October 1990.

[Clark 90] W. J. Clark, B. Lee, D. E. Lewis, and T. I. Mason, "Multipoint audiovisual telecommunications", *British Telecom Technology Journal*, 8(3):36–42, July 1990.

[CMTT 89] CMTT, "Draft new recommendation: transmission of component-coded digital video signals for contribution-quality applications at the third hierarchical level of CCITT Recommendation G.702", CCIR Document CMTT/303: period '86–'90, Geneva, October 1989.

[Cominetti 90] M. Cominetti and F. Molo, "A codec for HDTV signal transmission through terrestrial and satellite digital links", In *NAB'90*, Atlanta, April 1990.

[Dalal 78] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets", *Communications of the ACM*, 21(12):1040–1048, December 1978.

[De Prycker88] M. De Prycker, "Data communication in an ATM network", *Electrical Communication*, 62(3/4):333–337, 1988.

[Deering 90] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended LANs", *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[Ferrari 90] D. Ferrari, "Client requirements for real-time communication services", RFC 1193, November 1990.

[Floyd 62] R. W. Floyd, "Algorithm 97: shortest path", *Communications of the ACM*, 5(6):345, June 1962.

[Ford 62] L. R. Ford Jr and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, New Jersey, 1962.

[Gardner 87] M. L. Gardner, I. S. Loobeek, and S. N. Cohn, "Type-of-service routing with loadsharing", In *Globecom'87*, Tokyo, Japan, November 1987.

[Garey 79] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[Greaves 90] D. J. Greaves, D. Lioupis, and A. Hopper, "The Cambridge Backbone Network", In *IEEE INFOCOM'90*, San Francisco, June 1990.

[Greaves 91] D. J. Greaves and K. Zielinski, "Preliminary performance results for CBN half-duplex VME stations", In *Second International Conference on Local Communication Systems*, Palma, Spain, June 1991.

[Green 91] P. E. Green, "Exploiting photonic technology for gigabit computer networks", In O. Spaniol and A. Danthine (editors), *High Speed Networking III*, pages 3–14. North-Holland, Amsterdam, 1991.

[Guglielmo 91] M. Guglielmo, G. Modena, and R. Montagna, "Speech and image coding for digital communications", *European Transactions on Telecommunications and Related Technologies*, 2(1):21–44, January/February 1991.

[Handel 89] R. Handel, "Evolution of ISDN towards broadband ISDN", *IEEE Network*, 3(1):7–13, January 1989.

[Harney 91] K. Harney, M. Keith, G. Lavelle, L. D. Ryan, and D. J. Stark, "The i750 video processor: a total multimedia solution", *Communications of the ACM*, 34(4):64–78, April 1991.

[Hayter 91] M. Hayter and D. McAuley, "The desk area network", Technical Report 228, University of Cambridge Computer Laboratory, England, UK, May 1991.

[Hedrick 88] C. L. Hedrick, "Routing Information Protocol", RFC 1058, June 1988.

[Hinden 82] R. Hinden and A. Sheltzer, "The DARPA internet gateway", RFC 823, SRI Network Information Center, September 1982.

[Honig 84] M. L. Honig and D. G. Messerschmitt, *Adaptive Filters: Structures, Algorithms and Applications*, Kluwer Academic Publishers, Boston, MA, 1984.

[Hopper 86] A. Hopper and R. M. Needham, "The Cambridge Fast Ring Networking System (CFR)", Technical Report 90, Cambridge University Computer Laboratory, June 1986.

[Hopper 88] A. Hopper and R. M. Needham, "The Cambridge Fast Ring networking system", *IEEE Transactions on Computers*, 37(10):1214–1223, October 1988.

[Hopper 90] A. Hopper, "Pandora – an experimental system for multimedia applications", *ACM Operating Systems Review*, 24(2):19–34, April 1990.

[Hopper 91] A. Hopper, "Design and use of high-speed networks in multimedia applications", In O. Spaniol and A. Danthine (editors), *High Speed Networking III*, pages 25–38. North-Holland, Amsterdam, 1991.

[ISO90] "Information Technology, Telecommunications and Information Exchange Between Systems, Intermediate System-to-Intermediate System Routing Information Exchange Protocol for Use in Conjunction with ISO 8473", ISO 10589, 1990.

[Jayant 84] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Signal Processing Series. Prentice Hall, Englewood Cliffs, 1984.

[Jiang 91] X. Jiang, "Path finding algorithms for broadband multicast", In O. Spaniol and A. Danthine (editors), *High Speed Networking III*, pages 153–164. North-Holland, Amsterdam, 1991.

[Jiang 92] X. Jiang, "Routing broadband multicast streams", *Computer Communications*, 15(1):45–51, January/February 1992.

[Jutand 87] F. Jutand, et al., "A 13.5 MHz single chip multiformat Discrete Cosine Transform", In *Proceedings of SPIE Symp. on Visual Communications*, Vol. 845, pages 6–12, 1987.

[Karp 72] R. M. Karp, "Reducibility among combinatorial problems", In R. E. Miller and J. W. Thatcher (editors), *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.

[Keith 88] J. M. Keith, S. J. Golin, A. H. Simon, and B. Astle, "Digital video decompression system", US Patent No. 4785349, November 1988.

[Kou 81] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees", *Acta Informatica*, 15:141–145, 1981.

[Kruskal 56] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proceedings of the American Mathematics Society*, 7:48–50, 1956.

[Lai 85] W. S. Lai, "Bifurcated routing in computer networks", *ACM Computer Communications Review*, 15(3):28–49, August 1985.

[Lawler 76] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, New York, 1976.

[Lazar 87] A. A. Lazar and J. S. White, "Packetized video on MAGNET", *Optical Engineering*, 26(7):596–602, July 1987.

[Le Gall 91] D. Le Gall, "MPEG: a video compression standard for multimedia applications", *Communications of the ACM*, 34(4):46–58, April 1991.

[Liebhold 91] M. Liebhold and E. M. Hoffert, "Toward an open environment for digital video", *Communications of the ACM*, 34(4):103–112, April 1991.

[Liou 88] M. L. Liou, M. T. Sun, and L. Wu, "Two-dimensional Discrete Cosine Transform processor", US Patent 4791598, December 1988.

[Liou 90] M. L. Liou, "Visual telephony as an ISDN application", *IEEE Communications Magazine*, 28(2):30–38, February 1990.

[Liou 91] M. Liou, "Overview of the p×64 kbit/s video coding standard", *Communications of the ACM*, 34(4):59–63, April 1991.

[Lippman 89] A. Lippman and W. Butera, "Coding image sequences for interactive retrieval", *Communications of the ACM*, 32(7):852–860, July 1989.

[Lippman 91] A. Lippman, "Feature sets for interactive images", *Communications of the ACM*, 34(4):92–102, April 1991.

[Luther 91] A. C. Luther, *Digital Video in the PC Environment*, Intertext and McGraw-Hill, New York, 2nd edition, 1991.

[Mansfield 90] N. Mansfield, *The X Window System: A User's Guide*, Addison-Wesley, Amsterdam, 1990.

[McAuley 90] D. R. McAuley, "Protocol design for high speed networks", Technical Report 186, University of Cambridge Computer Laboratory, Cambridge, England, January 1990.

[McQuillan 80] J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET", *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[Mehlhorn 88] K. Mehlhorn, "A faster approximation algorithm for the Steiner problem in graphs", *Information Processing Letters*, 27(3):125–128, March 1988.

[Moy 91] J. Moy, "OSPF Version 2", RFC 1247, July 1991.

[Musmann 85] H. G. Musmann, P. Pirsch, and H. J. Grallert, "Advances in picture coding", *Proceedings of the IEEE*, 73(4):523–548, April 1985.

[Netravali 80] A. N. Netravali and J. O. Limb, "Picture coding: a review", *Proceedings of the IEEE*, 68(3):366–406, March 1980.

[Netravali 88] A. N. Netravali and B. G. Haskell, *Digital Pictures — Representation and Compression*, Plenum Press, New York, 1988.

[Newman 88] R. M. Newman, Z. L. Budrikis, and J. L. Hullett, "The QPSX MAN", *IEEE Communications Magazine*, 26(4):20–28, April 1988.

[OHRS 90] Open High Resolution Systems Working Group, "On harmonization of broadcast and non-broadcast uses of HRS/HDTV", Reference document submission to CCIR Study Group 11, Interim Working Party 11/9, September 1990.

[Perlman 91] R. Perlman, "A comparison between two routing protocols: OSPF and IS-IS", *IEEE Network*, 5(5):18–24, September 1991.

[Plesnik 81] J. Plesnik, "A bound for the Steiner tree problem in graphs", *Math. Slovaca*, 31:155–163, 1981.

[Postel 81] J. Postel, "Internet Protocol", RFC 791, USC/Information Sciences Institute, September 1981.

[Rayward 83] V. J. Rayward-Smith, "The computation of nearly minimal Steiner trees in graphs", *Int. J. Math. Educ. Sci. Technol.*, 14(1):15–23, 1983.

[Rayward 84] V. J. Rayward-Smith and A. Clare, "On finding Steiner vertices", Technical report, School of Computer Studies and Accountancy, University of East Anglia, England, UK, 1984.

[Rayward 86] V. J. Rayward-Smith and A. Clare, "On finding Steiner vertices", *Networks*, 16:283–294, 1986.

[Ross 89] F. E. Ross, "An overview of FDDI: the Fiber Distributed Data Interface", *IEEE Journal on Selected Areas in Communications*, 7(7):1043–1051, September 1989.

[Sage 71] A. P. Sage and J. J. Melsa, *Estimation Theory with Applications to Estimation and Control*, McGraw-Hill, New York, 1971.

[Sandbank 90] C. P. Sandbank (editor), *Digital Television*, John Wiley & Sons, West Sussex, England, 1990.

[Sijsterman91a] F. Sijstermans and J. van der Meer, "CD-I full-motion video decoder", *Communications of the ACM*, 34(4):90, April 1991.

[Sijsterman91b] F. Sijstermans and J. van der Meer, "CD-I full motion video encoding on a parallel computer", *Communications of the ACM*, 34(4):81–91, April 1991.

[Singhal 90] S. Singhal, D. Le Gall, and C. T. Chen, "Source coding of speech and video signals", *Proceedings of the IEEE*, 78(7):1233–1249, July 1990.

[Slater 91] J. Slater, *Modern Television Systems to HDTV and Beyond*, Pitman, 1991.

[Sullivan 82] G. F. Sullivan, "Approximation algorithms for Steiner tree problems", Technical Report 249, Department of Computer Science, Yale University, 1982.

[Sun 87] M. T. Sun, L. Wu, and M. L. Liou, "A concurrent architecture for VLSI implementation of Discrete Cosine Transform", *IEEE Transactions on Circuits and Systems*, CAS-34(8):992–994, August 1987.

[Sun 89a] M. T. Sun, T. C. Chen, and A. M. Gottleib, "VLSI implementation of a 16×16 Discrete Cosine Transform", *IEEE Transactions on Circuits and Systems*, CAS-36(4):610–617, April 1989.

[Sun 89b] M. T. Sun, K. M. Yang, and K. H. Tzou, "High-speed programmable ICs for decoding of variable length codes", In A. G. Tescher (editor), *Proceedings of SPIE Applications of Digital Image Processing XII*, Vol. 1153, pages 28–39, San Diego, California, August 1989.

[Takahashi 80] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs", *Math. Japonica*, 24:573–577, 1980.

[Tsai 89] W. T. Tsai, C. V. Ramamoorthy, W. K. Tsai, and O. Nishiguchi, "An adaptive hierarchical routing protocol", *IEEE Transactions on Computers*, 38(8):1059–1075, August 1989.

[Verbiest 88] W. Verbiest, L. Pinnoo, and B. Voeten, "The impact of the ATM concept on video coding", *IEEE Journal on Selected Areas in Communications*, 6(9):1623–1632, December 1988.

[Vetterli 86] M. Vetterli and A. Ligtenberg, "A Discrete Fourier-Cosine Transform chip", *IEEE Journal on Selected Areas in Communications*, SAC-4(1):49–61, January 1986.

[Waitzman 88] D. Waitzman, C. Partridge, and S. Deering, "Distance vector multicast routing protocol", RFC 1075, USC/Information Sciences Institute, November 1988.

[Wall 80] D. W. Wall, "Mechanisms for broadcast and selective broadcast", Technical Report STAN-CS-82-919, Department of Computer Science, Stanford University, Stanford, CA 94305, June 1980.

[Wall 82] D. W. Wall, "Selective broadcast in packet-switched networks", In *Proceedings of the sixth Berkeley workshop on distributed data management and computer networks*, pages 239–258, CA, USA, February 1982, also available from *Multicast Communication in Distributed Systems*, edited by M. Ahamad, IEEE Computer Society Press, Los Alamitos, California, 1990.

[Wallace 91] G. K. Wallace, "The JPEG still picture compression standard", *Communications of the ACM*, 34(4):30–44, April 1991.

[Wang 90] Z. Wang and J. Crowcroft, "Shortest path first with emergency exits", In *SIGCOMM'90*, pages 166–176, Philadelphia, Pennsylvania, September 1990.

[Waxman 88a] B. M. Waxman, "Routing of multipoint connections", *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.

[Waxman 88b] B. M. Waxman and M. Imase, "Worst case performance of Rayward-Smith's Steiner tree heuristic", Technical Report WUCS-88-13, Washington Univ., Dept. Comput. Sci., St. Louis, MO, 1988.

[Widmayer 86] P. Widmayer, "A fast approximation algorithm for Steiner's problems in graphs", In G. Goos and J. Hartmaris (editors), *Graph-Theoretic Concepts in Computer Science, International Workshop WG'86*, Vol. 246 of *Lecture Notes in Computer Science*, pages 17–28. Springer-Verlag, Berlin, Germany, June 1986.

[Winter 87] P. Winter, "Steiner problem in networks: a survey", *Networks*, 17(2):129–167, 1987.

[Wu 86] Y. F. Wu, P. Widmayer, and C. K. Wong, "A faster approximation algorithm for the Steiner problem in graphs", *Acta Informatica*, 23(2):223–229, 1986.

[Yang 88] K. M. Yang, L. Wu, H. Chong, and M. T. Sun, "VLSI implementation of motion compensation full-search block matching algorithm", In *Proceedings of SPIE Visual Communications and Image Processing*, Vol. 1001, Cambridge, MA, November 1988.