RW354
Principles of Computer Networking

A.E. Krzesinski and B.A. Bagula
Department of Computer Science
University of Stellenbosch

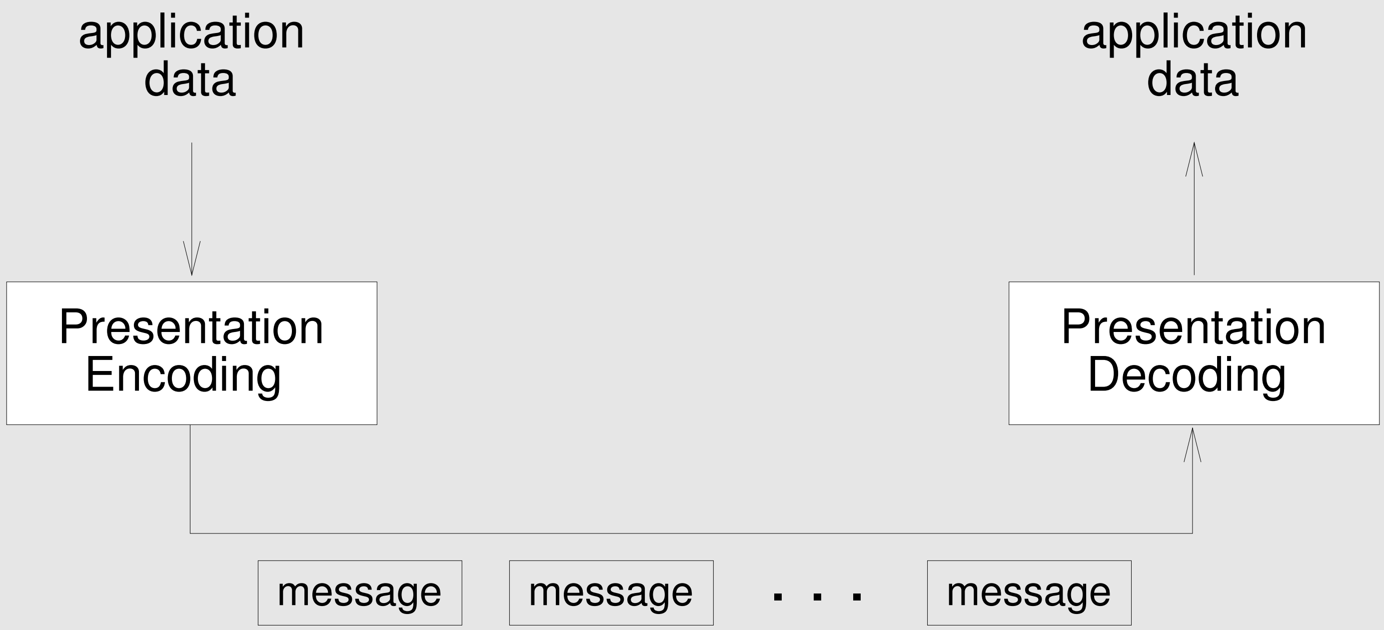
Last updated: 14 October 2004

The material presented in these slides is used with permission from

- *Larry L. Peterson and Bruce S. Davie. Computer Networks: A Systems Approach (Second Edition). Morgan Kaufmann Publishers. ISBN 1-55860-577-0.*
- *William Stallings. Data and Computer Communications (Sixth Edition). Prentice-Hall Inc. ISBN 0-13-571274-2.*
- *Andrew S. Tannenbaum. Computer Networks (Fourth Edition). Prentice Hall Inc. ISBN 0-13-349945-6.*

Presentation: Overview

- *encoding (marshalling) application data into messages*
- *decoding (unmarshalling) messages into application data.*



The application data are transformed into a form suitable for transmission over the network.

Presentation: Overview

- *Data types we consider*
  - *integers*
  - *floating point numbers*
  - *character strings*
  - *arrays*
  - *structures.*
- *Types of data we do not consider (now):*
  - *images*
  - *video*
  - *multimedia documents.*

Presentation: Difficulties

Representation of base types

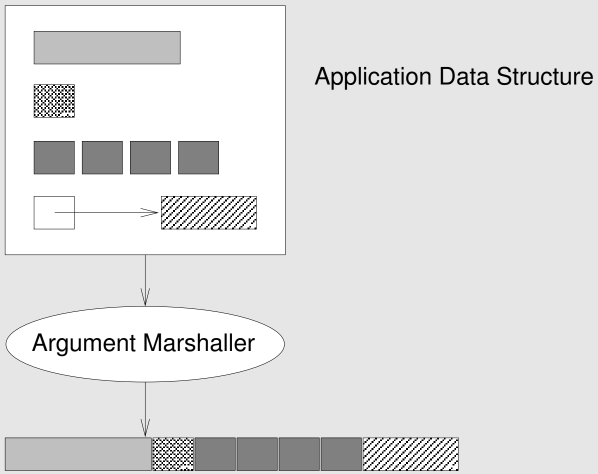
- *floating point: IEEE 754 versus non-standard*
- *integer: big-endian versus little-endian (e.g. 34,677,374)*

	(2)	(17)	(34)	(126)
big endian	00000010	00010001	00100010	01111110
	(126)	(34)	(17)	(2)
little endian	01111110	00100010	00010001	00000010
	high address			low address

- *Compiler layout of structures*

Presentation: Taxonomy

- *Data types*
  - *base types (e.g. ints, floats); must convert*
  - *flat types (e.g. structures, arrays); must pack*
  - *complex types (e.g. pointers); must serialise*



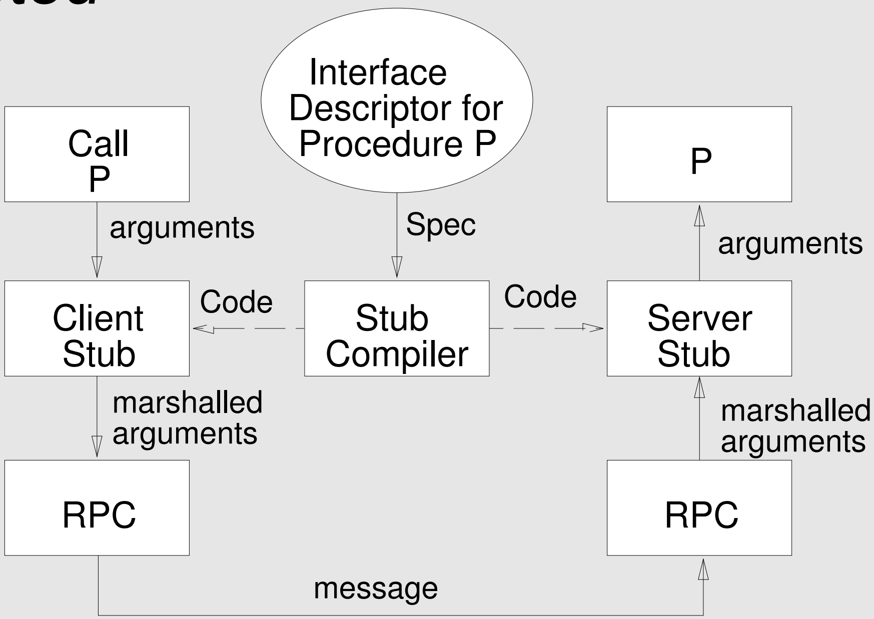
- *Conversion strategy*
  - *canonical intermediate form*
  - *receiver-makes-right: an $N \times N$ solution.*

Presentation: Taxonomy

- *Tagged versus untagged data*

INT	4	←	4–byte integer		→

- *Stubs*
  - *compiled*
  - *interpreted*



Presentation: Examples

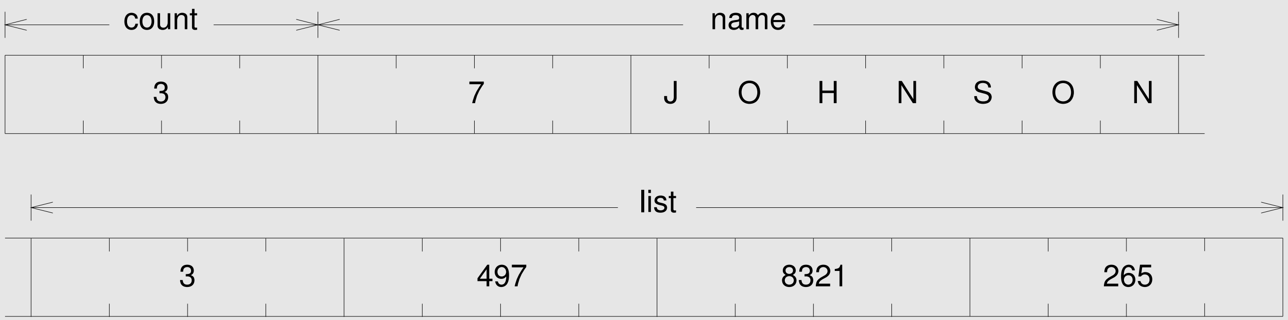
XDR: eXternal Data Representation

- *Defined by Sun for use with SunRPC*
- *C type system (without function pointers)*
- *Canonical intermediate form*
- *Untagged (except array length)*
- *Compiled stubs*

Presentation: Examples

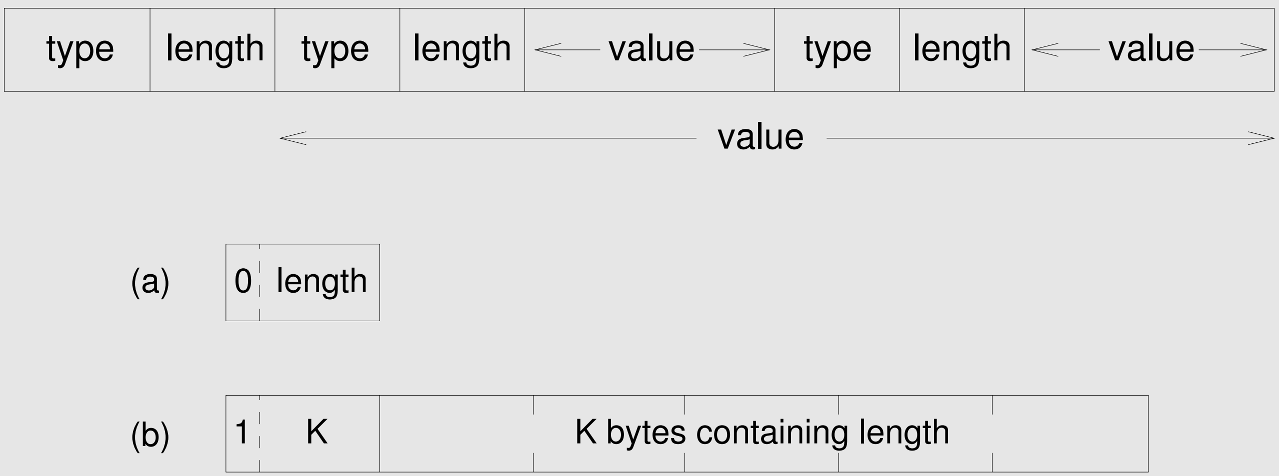
```
#define MAXNAME 256;
#define MAXLIST 100;

struct item {
   int      count;
   char     name[MAXNAME];
   int      list[MAXLIST];
};
bool_t
xdr_item(XDR *xdrs, struct item *ptr) {
    return(xdr_int(xdrs, &ptr->count) &&
       xdr_string(xdrs, &ptr->name, MAXNAME) &&
       xdr_array (xdrs, &ptr->list, &ptr->count,
                  MAXLIST, sizeof(int), xdr_int));
}
```



Presentation: ASN.1 Abstract Syntax Notation One

- *an ISO standard*
- *essentially the C type system*
- *canonical intermediate form*
- *tagged*
- *compiled or interpreted stubs*
- *BER: Basic Encoding Rules* $(\mathtt{tag}, \mathtt{length}, \mathtt{value})$



Presentation: NDR: Network Data Representation

- *defined by Distributed Computing Environment (DCE)*
- *essentially the C type system*
- *receiver-makes-right (architecture tag)*
- *individual data items untagged*
- *compiled stubs from IDL.*

Presentation: NDR: Network Data Representation

4-byte architecture definition tag

Integer Rep	Char Rep	FloatRep	Extension 1	Extension 2

- `IntegerRep`
  - *0 = big-endian* • *1 = little-endian*
- `CharRep`
  - *0 = ASCII* • *1 = EBCDIC*
- `FloatRep`
  - *0 = IEEE 754* • *1 = VAX* • *2 = Cray* • *3 = IBM*

Data Compression: Overview

Data must be encoded into a message. Compression is concerned with with removing redundancy from the encoding. There are two classes of compression

- *Loss-less: ensures that the data recovered from the compression/decompression process are exactly the same as the original data. Commonly used to compress executable code, text files & numeric data.*
- *Lossy: does not ensure that the data received are exactly the same as the data sent; removes information that it cannot later restore. Hopefully, no one will notice. Commonly used to compress digital imagery, including video.*

Data Compression: Overview

The compression/decompression process takes time. Whether or not you compress data (& how much you compress it) depends on whether you have more cycles (for compression) or bandwidth (for transmission).

Loss-Less Compression Algorithms

- *Run Length Encoding (RLE)*
  - *AAABBCDDDD is encoded as 3A2B1C4D*
  - *good for scanned text: 8-to-1 compression ratio*
  - *can increase the size for data with variation (e.g. some images).*
- *Differential Pulse Code Modulation (DPCM)*
  - *AAABBCDDDD is encoded as A0001123333*
  - *change the reference symbol if the delta becomes too large*
  - *works better than RLE for many digital images: 1.5-to-1 compression ratio.*

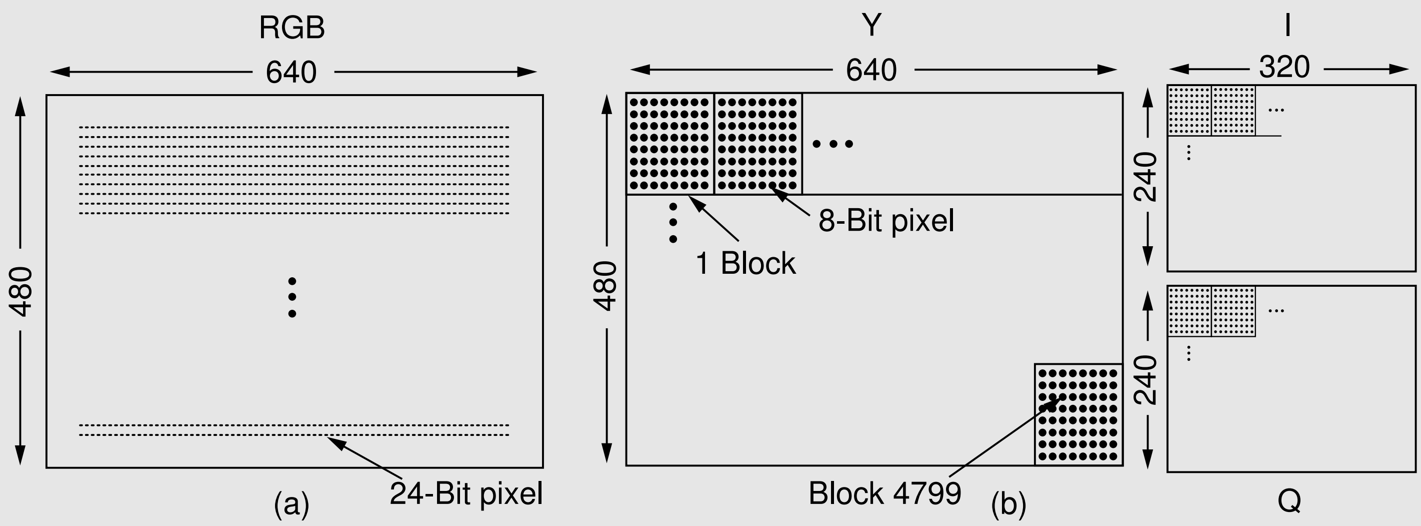
Loss-Less Compression Algorithms

Dictionary-Based Methods

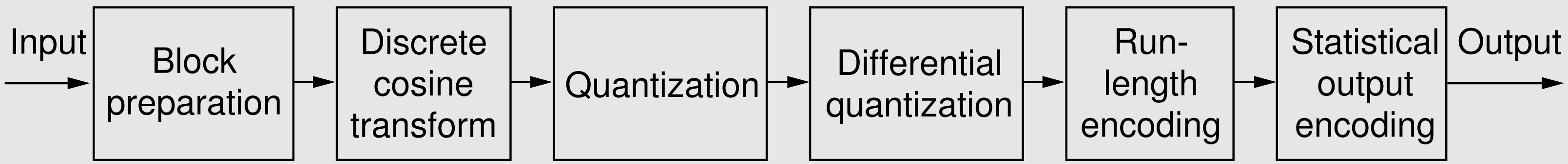
- *build a dictionary of common terms (variable length strings)*
- *translate each term into an index into the dictionary*
- *transmit the index into the dictionary for each term*
- *Lempel-Ziv (LZ) compression is the best-known example*
    - *commonly achieves a 2-to-1 compression ratio on text*
    - *a variation of LZ is used to compress GIF images*
        - *first reduce 24-bit color to 8-bit color*
        - *treat common sequences of pixels as terms in a dictionary*
        - *can achieve 10-to-1 compression ($\times$3).*

JPEG: Joint Photographic Expert Group

- *process the picture in 8×8 blocks of pixel values*
- *greyscale images: each pixel is an 8-bit value*
- *color images: each pixel is three values (Y,I,Q) for NTSC or (Y,U,V) for PAL*
  - *luminance* $Y = 0.30R + 0.59G + 0.11B$
  - *chrominance* $I = 0.60R - 0.28G - 0.32B$
  - *chrominance* $Q = 0.21R - 0.52G + 0.31B$



JPEG: Joint Photographic Expert Group



The idea is to change the picture into a vector of numbers that reveals the redundancies which can be removed by compression.

Lossy still image compression using a three phase process

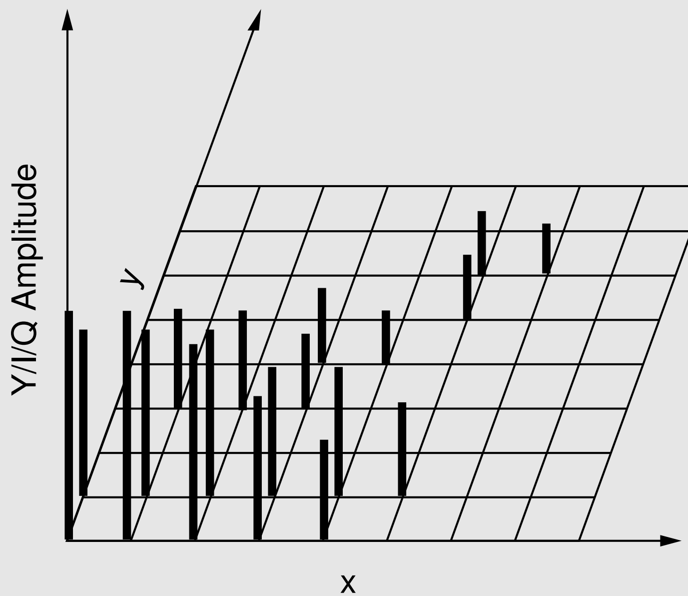
- *use a discrete cosine transformation (DCT) to transform the signal from the spatial domain to the frequency domain: loss-less*
- *apply a quantization to the results: lossy*
- *RLE-like encoding: loss-less.*

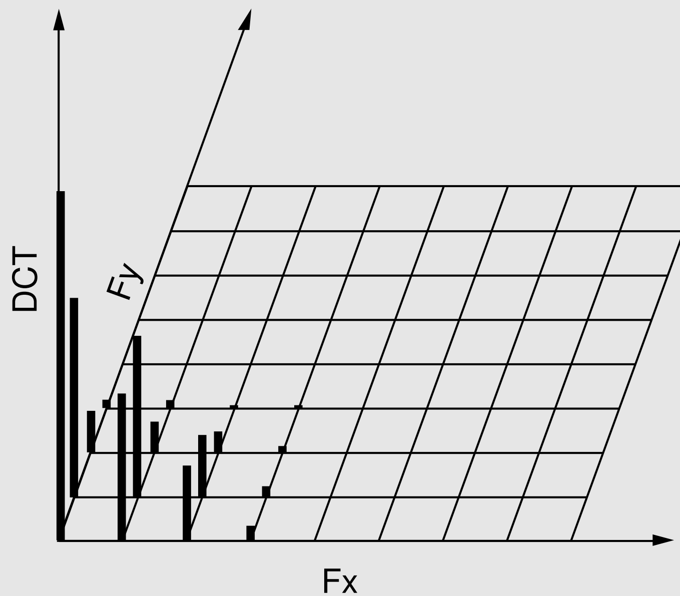
JPEG: DCT Phase

The DCT is applied to each 8x8 matrix of pixel values to output an 8x8 matrix of frequency coefficients.

DCT(0,0) is the DC coefficient representing the average value (large-scale features) of the 64 coefficients.

The other DCT coefficients are the AC coefficients which add higher frequency information (fine detail) to the DC coefficient.





JPEG: Quantization Phase

$$\texttt{QuantizedValue}(i,j) = \texttt{IntegerRound}(DCT(i,j)/\texttt{Quantum}(i,j))$$

$$\texttt{IntegerRound}(x) = \begin{cases} \lfloor x + 0.5 \rfloor & x \geq 0 \\ \lfloor x - 0.5 \rfloor & x < 0 \end{cases}$$

DCT Coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Quantized coefficients

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Decompression

$$DCT(i,j) = \texttt{QuantizedValue}(i,j) \times \texttt{Quantum}(i,j)$$

JPEG: Encoding Phase

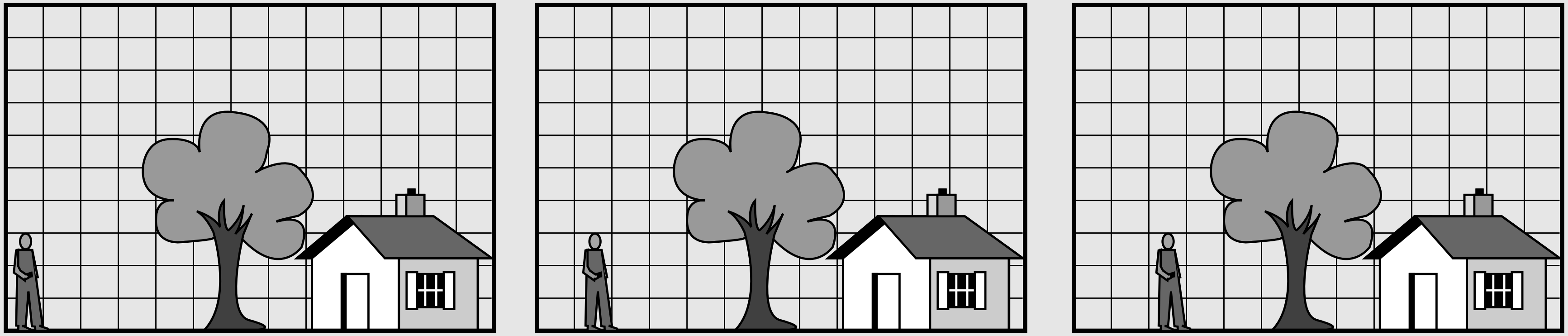
The quantized frequency coefficients are encoded using RLE & transmitted in this order

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MPEG: Motion Picture Expert Group

- *Lossy compression of video*
- *First approximation: JPEG on each frame*
- *Added compression to remove inter-frame redundancy: each frame is compressed into one of three frame types*
  - *intrapicture I frames are self-contained reference frames*
  - *P & B frames are not self-contained: they specify relative differences from some reference frame*
    - *predicted picture P frames specify the relative difference from the previous I frame*
    - *bidirectional predicted picture B frames interpolate previous & subsequent I or P frames.*
- *The audio signal is encoded with the video signal.*

MPEG: Inter Frame Redundancy



25 frames/second. Each frame is transmitted twice to avoid flicker.

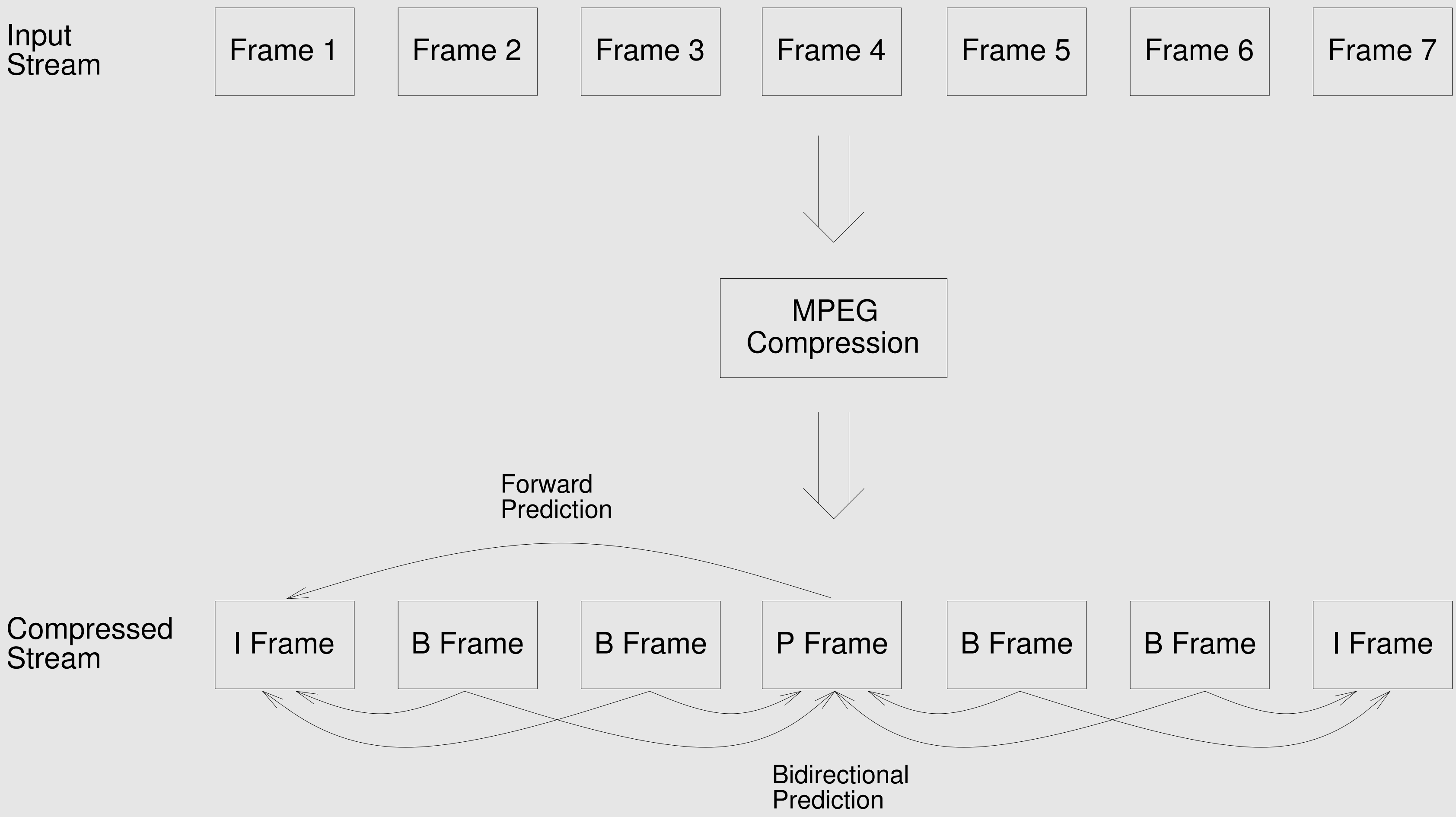
The lowest resolution colour frame is 1024 x 768 pixels.

For colour TV, 1 pixel = 24 bits (8 bits for each primary coplour).

The required data transmission rate is
2 x 25 x 1024 x 768 x 24 = 944 Mbps.

Compression is needed to transmit video over the Internet.

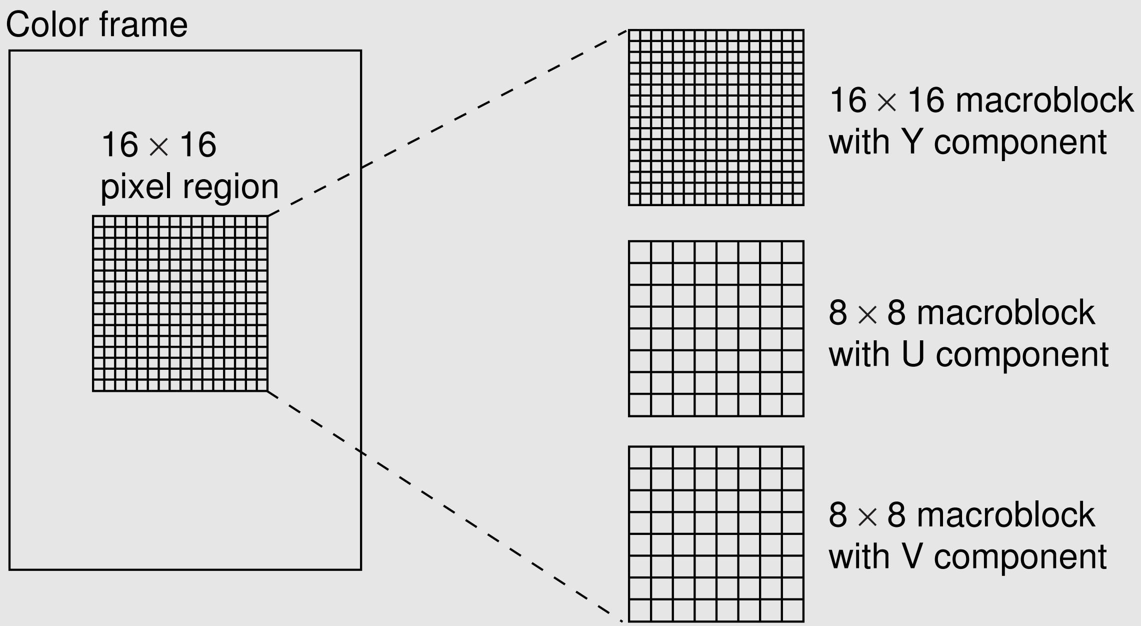
MPEG: Video Compression



The example sequence is transmitted as I P B B I B B

MPEG: I Frame Compression

Similar to JPEG except that MPEG works with 16x16 macroblocks.



The human eye is more sensitive to brightness (Y) than to colour (U,V) therefore the U,V components are transmitted less accurately than the Y component.

MPEG: P & B Frame Compression

The P & B frames are also processed in 16x16 macroblocks.

P & B frames are represented by a 4-tuple

- *a coordinate for the macroblock in the frame*
- *a motion vector* (x_p, y_p) *relative to the previous reference frame*
- *a motion vector* (x_f, y_f) *relative to the next reference frame (B only)*
- *a delta* $\delta(x, y)$ *for each pixel* (x, y) *in the macroblock.*

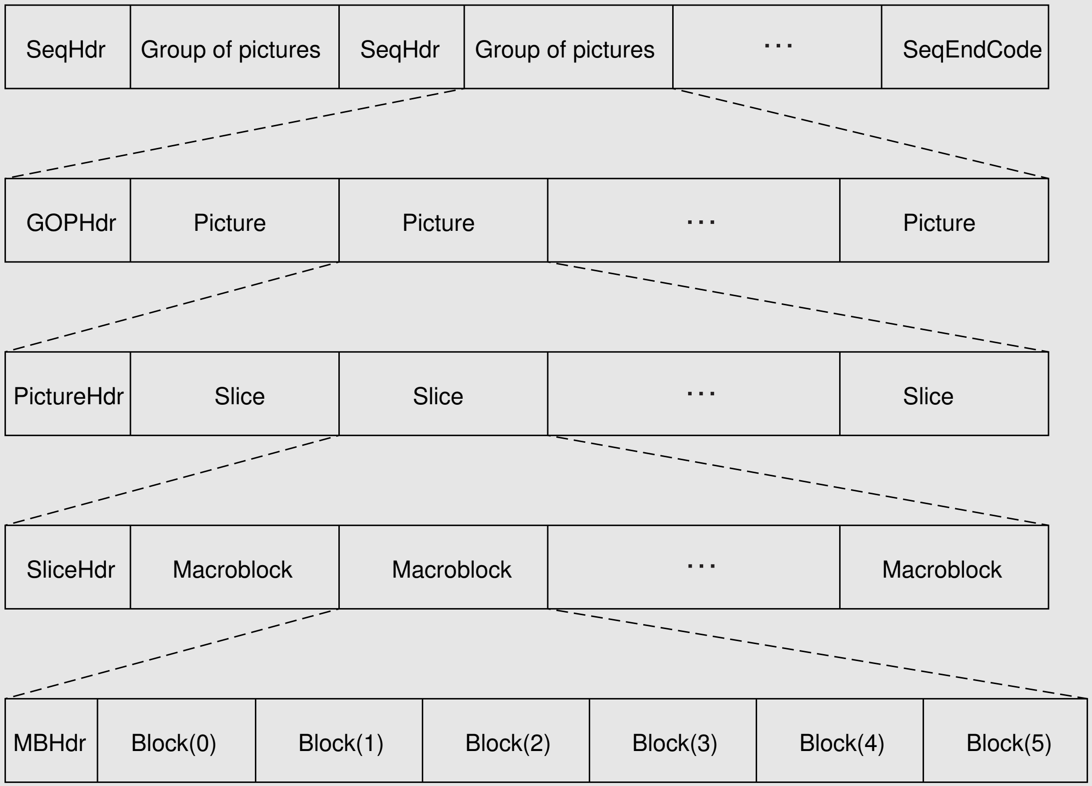
The pixel at coordinate (x, y) *in the current frame is*

$$F_c(x,y) = (F_p(x + x_p, y + y_p) + F_f(x + x_f, y + y_f))/2 + \delta(x,y)$$

MPEG: Effectiveness

- *the MPEG compression ratio is typically 90-to-1*
- *as high as 150-to-1*
- *30-to-1 for I frames*
- *P& B frames get another 3 to 5 times*
- *MPEG encoding is computationally expensive and is done off-line*
- *MPEG decoding is computationally less expensive and can be done on-line: a 400Mhz processor can decompress MPEG fast enough for a 640x480 video stream at 20 frames/second.*

Transmitting MPEG over a network



Transmitting MPEG over a network

Adaptive video coding: MPEG allows a trade-off between transmission bandwidth & picture quality.

`SeqHdr`

- *size of each picture*
- *interpicture period*
- *two quantization matrices: I frames, P & B frames*
- *the quantization table & frame rate can change at GOP boundaries.*

`GOPHdr`

- *number of pictures in the GOP*
- *where the GOP plays relative to the start of the movie*

Transmitting MPEG over a network

`PictureHdr`

- *the type of picture (I,P or B)*
- *picture-specific quantization table*

`SliceHdr`

- *vertical position of the slice*
- *scaling factor for the quantization table*

`MBHdr`

- *macroblock address within the picture*

Transmitting MPEG over a network

Video is not sent using TCP since TCP's re-transmission of lost segments would lead to unacceptable delays.

If video is sent using UDP then the packets end at macroblock boundaries.

If a B frame is lost then the previous B frame can be re-played. If an I frame is lost then none of the subsequent P & B frames can be used. A solution is to use Diffserv to mark I frames with a low drop probability.

The transmission of B frames must be delayed until their I & P frames have been transmitted. This can introduce unacceptable delays into interactive video streams. Motion-JPEG is used to encode video conferences: only I and P frames are used.

MP3 (MPEG Audio Layer 3) Audio Compression

CD-quality audio is sampled at 44.1KHz for a bandwidth of 0 to 22KHz. Each sample is quantized into 16 bits.

A 2-channel stereo audio stream is thus 1.41Mbps. Synchronization & error correction codes require 49 bits to encode each 16 bit sample. The actual bit rate is thus 4.32Mbps.

MP3 splits the audio stream into sub-bands. Each sub-band is split into variable length blocks of 64 to 1024 samples. A DCT is applied to each block which is then quantized & encoded.

Psychoacoustic models determine how many sub-bands are used & how many bits are allocated to each sub-band. The quantization tables for each sub-band may be changed.