Chapter 7
End to End Data

Gandeva Bayu Satrya (GBS)

gbs@ittelkom.ac.id

gandeva.bayu.s@gmail.com

TELKOM ENGINEERING SCHOOL

Telkom University

Agenda

Chapter 7. End to End Data
[PETERSON & DAVIE]

7.1. Presentation Formatting

XDR, ASN.1, NDR, and XML

7.2. Data Compression

JPEG, and MPEG

Problem

- From the network's perspective, application programs send messages to each other.
- Each of these messages is just an uninterpreted string of bytes.
- From the application's perspective, however, these messages contain various kinds of *data—arrays of integers,* video frames, lines of text, digital images, and so on. In other words, these bytes have meaning.
- We now consider the problem of how best to encode the different kinds of data that application programs want to exchange into byte strings.

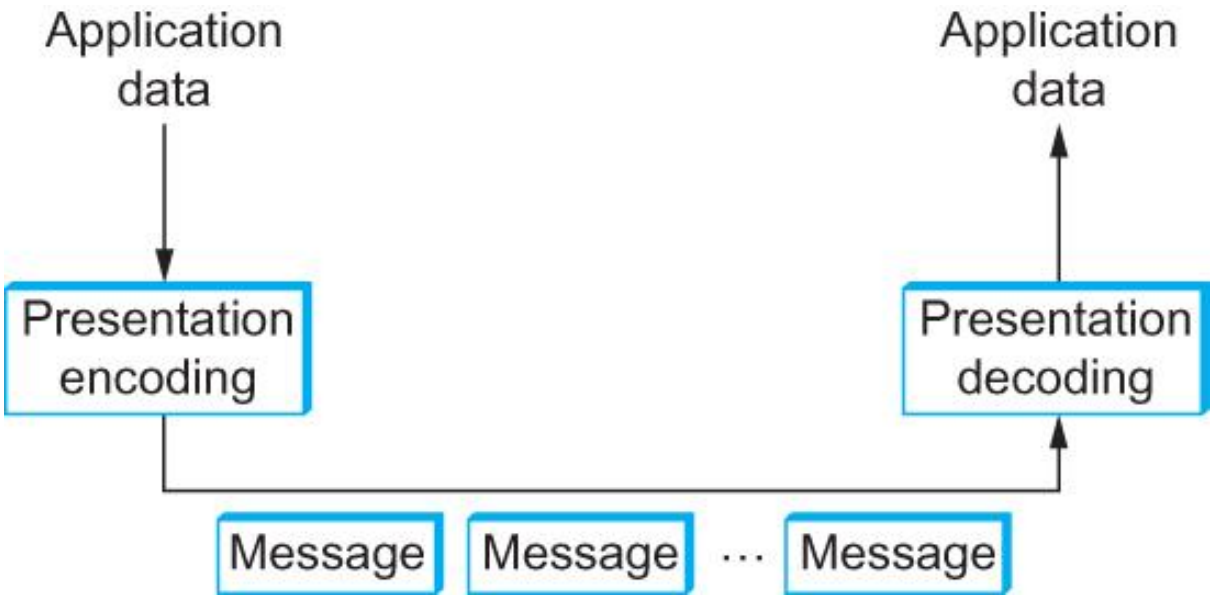
Presentation Formatting

- One of the most common transformations of network data is from the representation used by the application program into a form that is suitable for transmission over a network and vice versa.
- This transformation is typically called *presentation formatting.*

Presentation Formatting

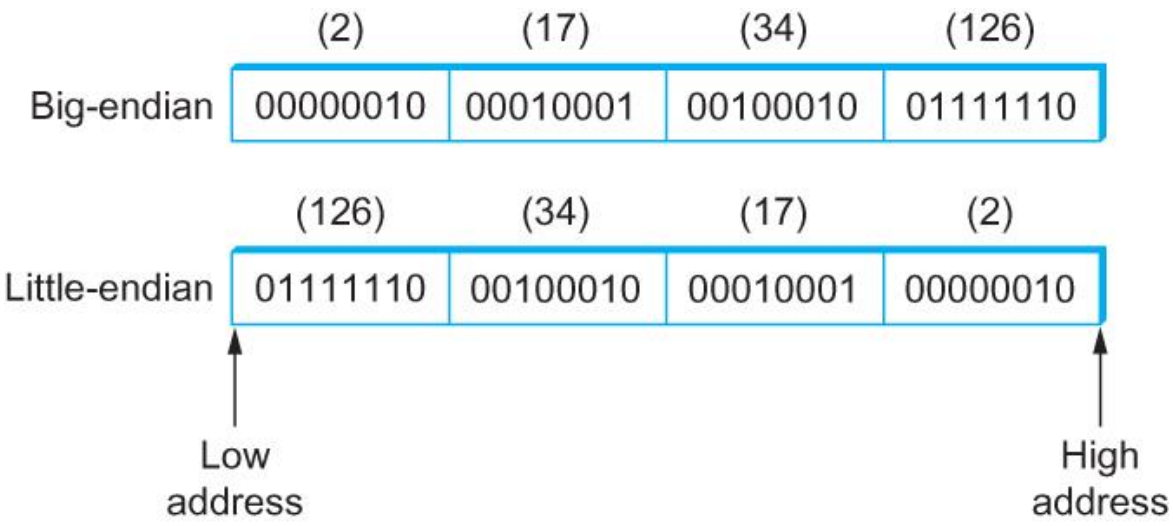
- The sending program translates the data it wants to transmit from the representation it uses internally into a message that can be transmitted over the network; that is, the data is *encoded in a message.*
- On the receiving side, the application translates this arriving message into a representation that it can then process; that is, the message is *decoded.* Encoding is sometimes called argument *marshalling, and* decoding is sometimes called *unmarshalling.* This terminology comes from the RPC world.

Presentation Formatting



Presentation formatting involves encoding and decoding application data

Presentation Formatting

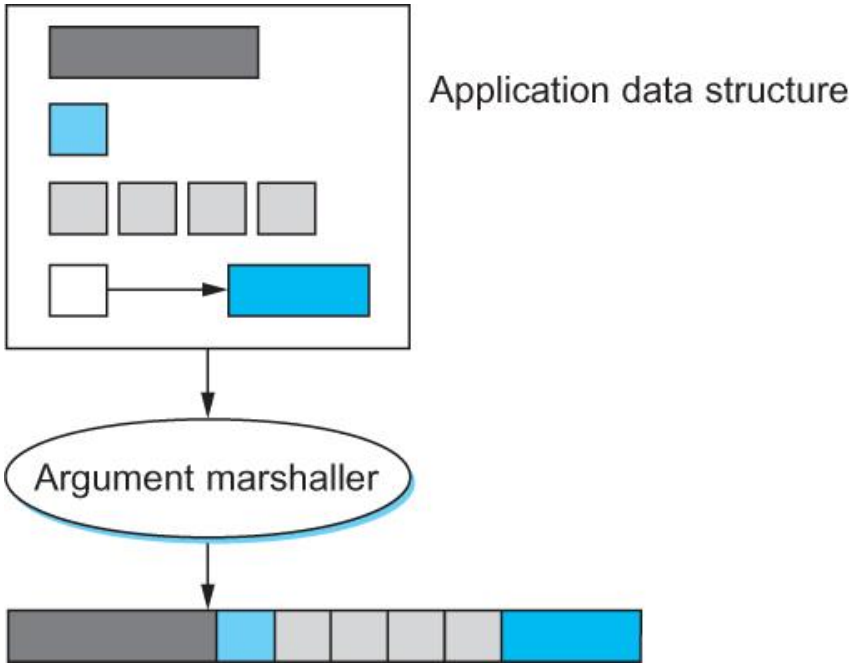


Big-endian and little-endian byte order for the integer 34,677,374.

Presentation Formatting

- Taxonomy
  - Data Types
    - The first question is what data types the system is going to support. In general, we can classify the types supported by an argument marshalling mechanism at three levels.
      - At the lowest level, a marshalling system operates on some set of *base types. Typically,* the base types include integers, floating-point numbers, and characters. The system might also support ordinal types and booleans.
      - At the next level are *flat types—structures and arrays.*
      - At the highest level, the marshalling system might have to deal with *complex types*—those types that are built using pointers
      - In summary, depending on how complicated the type system is, the task of argument marshalling usually involves converting the base types, packing the structures, and linearizing the complex data structures, all to form a contiguous message that can be transmitted over the network

Presentation Formatting



Argument marshalling: converting, packing, and linearizing

Presentation Formatting

- Conversion Strategy
  - Once the type system is established, the next issue is what conversion strategy the argument marshaller will use. There are two general options:
    - *canonical intermediate form* and *receiver-makes-right*

Presentation Formatting

Conversion Strategy

- The idea of canonical intermediate form is to settle on an external representation for each type; the sending host translates from its internal representation to this external representation before sending data, and the receiver translates from this external representation into its local representation when receiving data
- The alternative, which is sometimes called receiver-makes-right, has the sender transmit data in its own internal format; the sender does not convert the base types, but usually has to pack and flatten more complex data structures. The receiver is then responsible for translating the data from the sender's format into its own local format.

Presentation Formatting

Tags

- The third issue in argument marshalling is how the receiver knows what kind of data is contained in the message it receives. There are two common approaches: *tagged* and *untagged* data.
  - A tag is any additional information included in a message—beyond the concrete representation of the base types—that helps the receiver decode the message.
  - The alternative, of course, is *not to use tags*. How does the receiver know how to decode the data in this case? It knows because it was programmed to know.

Presentation Formatting

- Tags

A 32-bit integer encoded in a tagged message

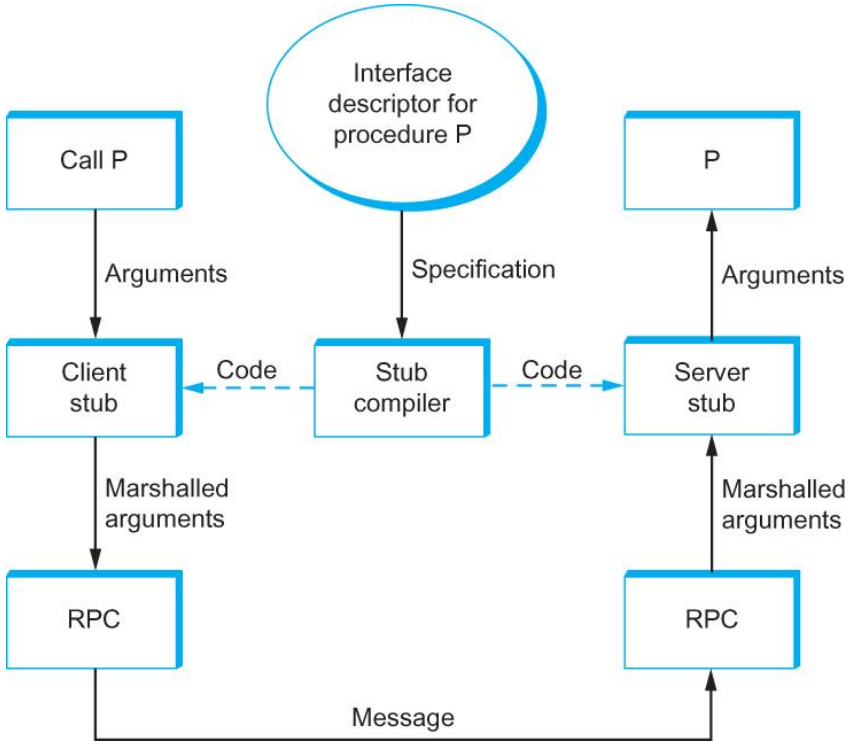
Presentation Formatting

Stubs

- A stub is the piece of code that implements argument marshalling.
- Stubs are typically used to support RPC. On the client side, the stub marshalls the procedure arguments into a message that can be transmitted by means of the RPC protocol.
- On the server side, the stub converts the message back into a set of variables that can be used as arguments to call the remote procedure.
- Stubs can either be interpreted or compiled

Presentation Formatting

- Stubs



Stub compiler takes interface description as input and outputs client and server stubs.

Presentation Formatting

- Examples
  - XDR [RFC 4506]
    - External Data Representation (XDR) is the network format used with SunRPC. In the taxonomy just introduced, XDR
      - supports the entire C type system with the exception of function pointers
      - defines a canonical intermediate form
      - does not use tags (except to indicate array lengths)
      - uses compiled stubs

Presentation Formatting

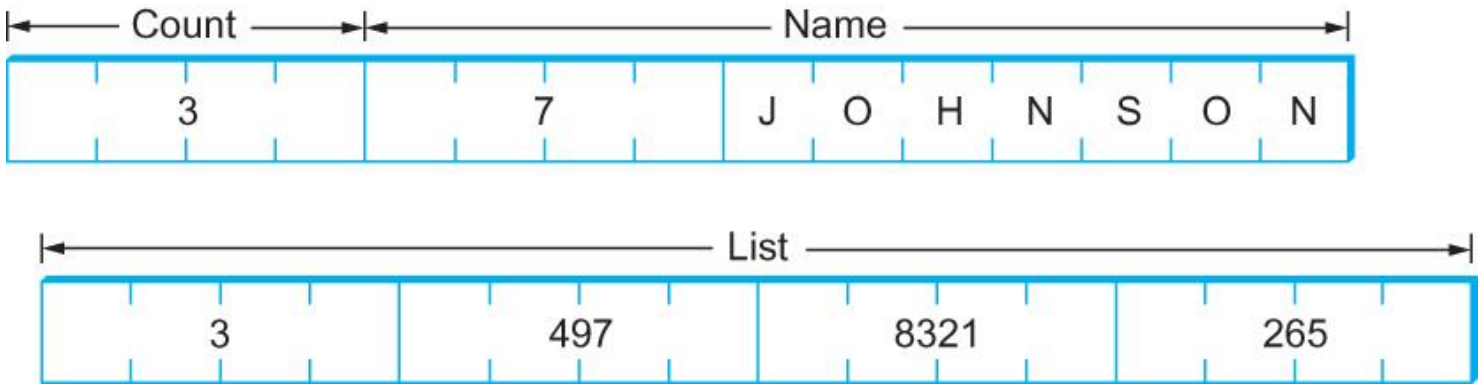
- Examples
  - XDR
    - An XDR integer is a 32-bit data item that encodes a C integer.
    - It is represented in twos complement notation, with the most significant byte of the C integer in the first byte of the XDR integer, and the least significant byte of the C integer in the fourth byte of the XDR integer.
      - That is, XDR uses big-endian format for integers.
    - XDR supports both signed and unsigned integers, just as C does.

Presentation Formatting

- Examples
  - XDR
    - XDR represents variable-length arrays by first specifying an unsigned integer (4 bytes) that gives the number of elements in the array, followed by that many elements of the appropriate type.
    - XDR encodes the components of a structure in the order of their declaration in the structure.
    - For both arrays and structures, the size of each element/ component is represented in a multiple of 4 bytes. Smaller data types are padded out to 4 bytes with 0s.
    - The exception to this “pad to 4 bytes” rule is made for characters, which are encoded one per byte.

Presentation Formatting

- Examples
  - XDR



Example encoding of a structure in XDR

Presentation Formatting

- Examples
  - ASN.1 [RFC 6019]
    - Abstract Syntax Notation One (ASN.1) is an ISO standard that defines, among other things, a representation for data sent over a network.
    - The representation-specific part of ASN.1 is called the Basic Encoding Rules (BER).
    - ASN.1 supports the C type system without function pointers, defines a canonical intermediate form, and uses type tags.
    - Its stubs can be either interpreted or compiled.
    - One of the claims to fame of ASN.1 BER is that it is used by the Internet standard Simple Network Management Protocol (SNMP).

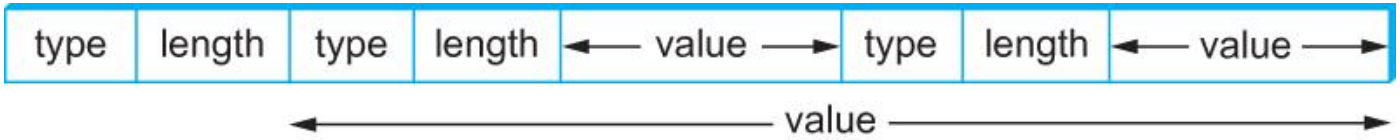
Presentation Formatting

- Examples
  - ASN.1
    - ASN.1 represents each data item with a triple of the form

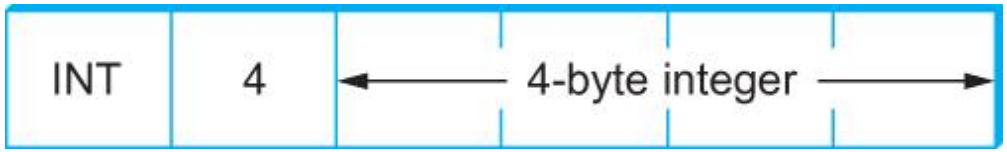
      < tag, length, value >
    - The tag is typically an 8-bit field, although ASN.1 allows for the definition of multi-byte tags.
    - The length field specifies how many bytes make up the value;
    - Compound data types, such as structures, can be constructed by nesting primitive types

Presentation Formatting

- Examples
  - ASN.1



Compound types created by means of nesting in ASN.1/BER



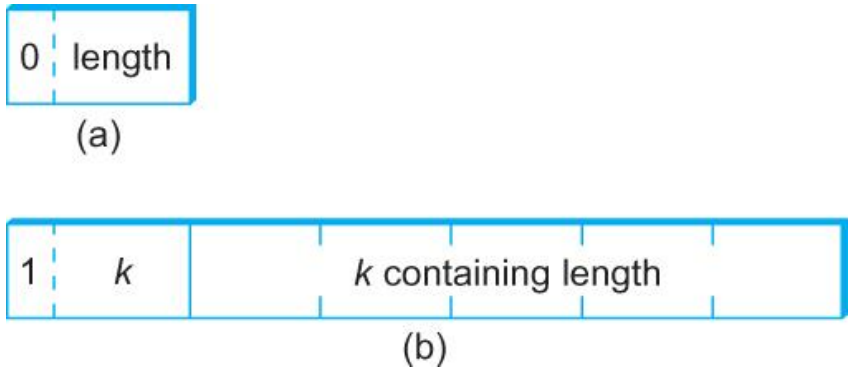
ASN.1/BER representation for a 4-byte integer

Presentation Formatting

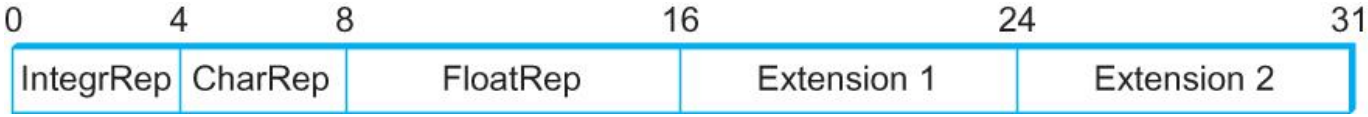
- Examples
  - NDR
    - Network Data Representation (NDR) is the data-encoding standard used in the Distributed Computing Environment
    - Unlike XDR and ASN.1, NDR uses receiver-makes-right. It does this by inserting an architecture tag at the front of each message; individual data items are untagged.
    - NDR uses a compiler to generate stubs.
      - This compiler takes a description of a program written in the Interface Definition Language (IDL) and generates the necessary stubs
      - IDL looks pretty much like C, and so essentially supports the C type system.

Presentation Formatting

- Examples
  - NDR



ASN.1/BER representation for length:
(a) 1 byte; (b) multibyte



NDR's architecture tag

Presentation Formatting

Markup Languages – XML [RFC 3023]

- Markup languages, of which HTML and XML are both examples, take the tagged data approach to the extreme.
- Data is represented as text, and text tags known as *markup* are intermingled with the data text to express information about the data.
- In the case of HTML, markup merely indicates how the text should be displayed; other markup languages like XML can express the type and structure of the data

Presentation Formatting

Markup Languages – XML

- XML syntax looks much like HTML.
- For example, an employee record in a hypothetical XML-based language might look like the XML document (next slide), which might be stored in a file named employee.xml.
- The first line indicates the version of XML being used, and the remaining lines represent four fields that make up the employee record, the last of which (hire date) contains three sub-fields.

Presentation Formatting

- Markup Languages – XML
  - In other words, XML syntax provides for a nested structure of tag/value pairs, which is equivalent to a tree structure for the represented data (with employee as the root).
  - This is similar to XDR, ASN.1, and NDR's ability to represent compound types, but in a format that can be both processed by programs and read by humans.
  - More importantly, programs such as parsers can be used across different XML-based languages, because the definitions of those languages are themselves expressed as machine-readable data that can be input to the programs.

Presentation Formatting

- Markup Languages – XML

```
<?xml version="1.0"?>
<employee>
<name>John Doe</name>
<title>Head Bottle Washer</title>
<id>123456789</id>
<hiredate>
<day>5</day>
<month>June</month>
<year>1986</year>
</hiredate>
</employee>
```

Presentation Formatting

Markup Languages – XML

- The definition of a specific XML-based language is given by a *schema, which is* a database term for a specification of how to interpret a collection of data.
- There are a number of schema languages defined for XML.
- We will focus here on the leading standard, known by the none-too-surprising name *XML Schema.*
- An individual schema defined using XML Schema is known as an *XML Schema Document (XSD).*

Presentation Formatting

- Markup Languages – XML
  - The XSD for the employee.xml example is shown in the next slide
    - it defines the language to which the example document conforms. It might be stored in a file named employee.xsd.

Presentation Formatting

- Markup Languages – XML

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
<element name="employee">
<complexType>
<sequence>
<element name="name" type="string"/>
<element name="title" type="string"/>
<element name="id" type="string"/>
<element name="hiredate">
<complexType>
<sequence>
<element name="day" type="integer"/>
<element name="month" type="string"/>
<element name="year" type="integer"/>
</sequence>
</complexType>
</element>
</sequence>
</complexType>
</element>
</schema>
```

Agenda

Chapter 7. End to End Data
[PETERSON & DAVIE]

7.1. Presentation Formatting

XDR, ASN.1, NDR, and XML

7.2. Data Compression

JPEG, and MPEG

Multimedia Data

- Multimedia data, comprising audio, video, and still images, now makes up the majority of traffic on the Internet by many estimates.
  - This is a relatively recent development—it may be hard to believe now, but there was no YouTube before 2005.
- Part of what has made the widespread transmission of multimedia across networks possible is advances in compression technology.
- Because multimedia data is consumed mostly by humans using their senses—vision and hearing—and processed by the human brain, there are unique challenges to compressing it.

Multimedia Data

- You want to try to keep the information that is most important to a human, while getting rid of anything that doesn't improve the human's perception of the visual or auditory experience.
- Hence, both computer science and the study of human perception come into play.
- In this section we'll look at some of the major efforts in representing and compressing multimedia data.

Multimedia Data

- To get a sense of how important compression has been to the spread of networked multimedia, consider the following example.
- A high-definition TV screen has something like 1080 × 1920 pixels, each of which has 24 bits of color information, so each frame is 1080 × 1920 × 24 = 50Mb and so if you want to send 24 frames per second, that would be over 1Gbps.
- That's a lot more than most Internet users can get access to, by a good margin.
- By contrast, modern compression techniques can get a reasonably high quality HDTV signal down to the range of 10 Mbps, a two order of magnitude reduction, and well within the reach of many broadband users.
- Similar compression gains apply to lower quality video such as YouTube clips—web video could never have reached its current popularity without compression to make all those entertaining videos fit within the bandwidth of today's networks.

Multimedia Data

Lossless Compression Techniques

- In many ways, compression is inseparable from data encoding.
  - That is, in thinking about how to encode a piece of data in a set of bits, we might just as well think about how to encode the data in the smallest set of bits possible.
  - For example, if you have a block of data that is made up of the 26 symbols A through Z, and if all of these symbols have an equal chance of occurring in the data block you are encoding, then encoding each symbol in 5 bits is the best you can do (since $2^5 = 32$ is the lowest power of 2 above 26).
  - If, however, the symbol R occurs 50% of the time, then it would be a good idea to use fewer bits to encode the R than any of the other symbols.
- In general, if you know the relative probability that each symbol will occur in the data, then you can assign a different number of bits to each possible symbol in a way that minimizes the number of bits it takes to encode a given block of data.
- This is the essential idea of *Huffman codes, one of the important early developments in data compression.*

Multimedia Data

Lossless Compression Techniques

Run length Encoding

- Run length encoding (RLE) is a compression technique with a brute-force simplicity.
- The idea is to replace consecutive occurrences of a given symbol with only one copy of the symbol, plus a count of how many times that symbol occurs—hence the name “run length.”
- For example, the string AAABBCDDDD would be encoded as 3A2B1C4D.

Multimedia Data

Lossless Compression Techniques

Differential Pulse Code Modulation

- Another simple lossless compression algorithm is Differential Pulse Code Modulation (DPCM).
- The idea here is to first output a reference symbol and then, for each symbol in the data, to output the difference between that symbol and the reference symbol.
- For example, using symbol A as the reference symbol, the string AAABBCDDDD would be encoded as A0001123333 since A is the same as the reference symbol, B has a difference of 1 from the reference symbol, and so on.

Multimedia Data

Lossless Compression Techniques

Dictionary based Methods

- The final lossless compression method we consider is the dictionary-based approach, of which the Lempel-Ziv (LZ) compression algorithm is the best known.
- The Unix compress and gzip commands use variants of the LZ algorithm.
- The idea of a dictionary-based compression algorithm is to build a dictionary (table) of variable-length strings (think of them as common phrases) that you expect to find in the data, and then to replace each of these strings when it appears in the data with the corresponding index to the dictionary.

Multimedia Data

Lossless Compression Techniques

- Dictionary based Methods
  - For example, instead of working with individual characters in text data, you could treat each word as a string and output the index in the dictionary for that word.
  - To further elaborate on this example, the word “compression” has the index 4978 in one particular dictionary; it is the 4978th word in /usr/share/dict/words.
  - To compress a body of text, each time the string “compression” appears, it would be replaced by 4978.

Multimedia Data

Image Representation and Compression

- Given the increase in the use of digital imagery in recent years—this use was spawned by the invention of graphical displays, not high-speed networks—the need for standard representation formats and compression algorithms for digital imagery data has grown more and more critical.
- In response to this need, the ISO defined a digital image format known as JPEG, named after the Joint Photographic Experts Group that designed it. (The “Joint” in JPEG stands for a joint ISO/ITU effort.)

Multimedia Data

Image Representation and Compression

- JPEG is the most widely used format for still images in use today.
- At the heart of the definition of the format is a compression algorithm, which we describe below.
- Many techniques used in JPEG also appear in MPEG, the set of standards for video compression and transmission created by the *Moving Picture Experts Group.*

Multimedia Data

Image Representation and Compression

- Digital images are made up of pixels (hence the megapixels quoted in digital camera advertisements).
- Each pixel represents one location in the two-dimensional grid that makes up the image, and for color images, each pixel has some numerical value representing a color.
- There are lots of ways to represent colors, referred to as *color spaces:* the one most people are familiar with is RGB (red, green, blue).

Multimedia Data

Image Representation and Compression

- You can think of color as being three dimensional quantity—you can make any color out of red, green and blue light in different amounts.
- In a three-dimensional space, there are lots of different, valid ways to describe a given point (consider Cartesian and polar co-ordinates, for example).
- Similarly, there are various ways to describe a color using three quantities, and the most common alternative to RGB is YUV. The Y is luminance, roughly the overall brightness of the pixel, and U and V contain chrominance, or color information.

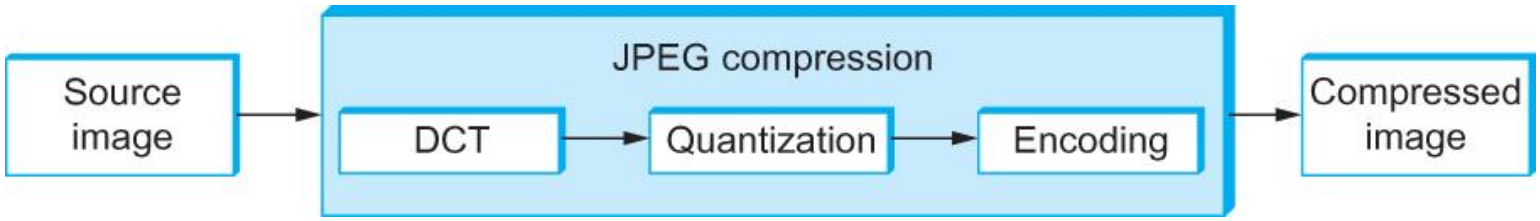
Multimedia Data

Image Representation and Compression

- Let's look at the example of the Graphical Interchange Format (GIF).
  - GIF uses the RGB color space, and starts out with 8 bits to represent each of the three dimensions of color for a total of 24 bits.
  - Rather than sending those 24 bits per pixel, however, GIF first reduces 24-bit color images to 8-bit color images.
  - This is done by identifying the colors used in the picture, of which there will typically be considerably fewer than 2^{24}, and then picking the 256 colors that most closely approximate the colors used in the picture.
  - There might be more than 256 colors, however, so the trick is to try not to distort the color too much by picking 256 colors such that no pixel has its color changed too much.

Multimedia Data

- Image Representation and Compression



Block diagram of JPEG compression

Multimedia Data

JPEG Compression

DCT Phase

- DCT is a transformation closely related to the fast Fourier transform (FFT). It takes an
- 8 × 8 matrix of pixel values as input and outputs an 8 × 8 matrix of frequency coefficients.
- You can think of the input matrix as a 64-point signal that is defined in two
- spatial dimensions (x and y); DCT breaks this signal into 64 spatial frequencies.

JPEG Compression

DCT Phase

- DCT, along with its inverse, which is performed during decompression, is defined by the following formulas:

$$DCT(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} pixel(x,y) \times \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]$$

$$pixel(x,y) = \frac{1}{\sqrt{2N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C(i)C(j)DCT(i,j) \times \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]$$

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

- where pixel(x, y) is the grayscale value of the pixel at position (x, y) in the 8×8 block being compressed; N = 8 in this case

Multimedia Data

JPEG Compression

Quantization Phase

- The second phase of JPEG is where the compression becomes lossy.
- DCT does not itself lose information; it just transforms the image into a form that makes it easier to know what information to remove.
- Quantization is easy to understand—it's simply a matter of dropping the insignificant bits of the frequency coefficients.

Multimedia Data

JPEG Compression

Quantization Phase

- The basic quantization equation is

  QuantizedValue(i, j) = IntegerRound(DCT(i, j)/Quantum(i, j))

  Where

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

- Decompression is then simply defined as

  DCT(i, j) = QuantizedValue(i, j) × Quantum(i, j)

Multimedia Data

JPEG Compression

Encoding Phase

- The final phase of JPEG encodes the quantized frequency coefficients in a compact form.
- This results in additional compression, but this compression is lossless.
- Starting with the DC coefficient in position (0,0), the coefficients are processed in the zigzag sequence.
- Along this zigzag, a form of run length encoding is used—RLE is applied to only the 0 coefficients, which is significant because many of the later coefficients are 0.
- The individual coefficient values are then encoded using a Huffman code.

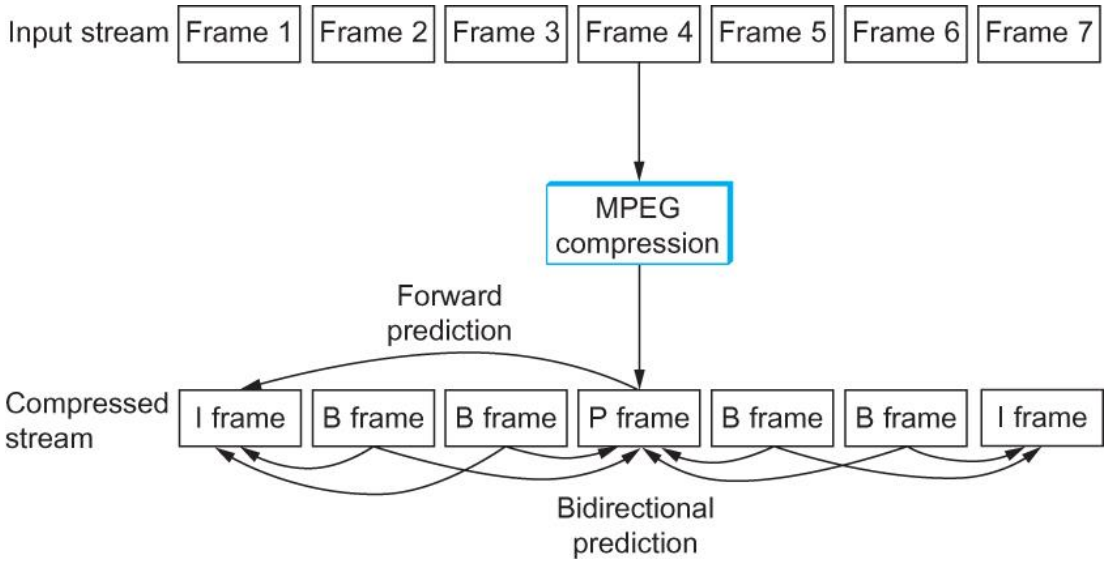
Multimedia Data

Video Compression (MPEG)

- We now turn our attention to the MPEG format, named after the Moving Picture Experts Group that defined it.
- To a first approximation, a moving picture (i.e., video) is simply a succession of still images—also called *frames or pictures*—*displayed at some* video rate.
- Each of these frames can be compressed using the same DCT-based technique used in JPEG.

Multimedia Data

- Video Compression (MPEG)



Sequence of I, P, and B frames generated by MPEG.

Multimedia Data

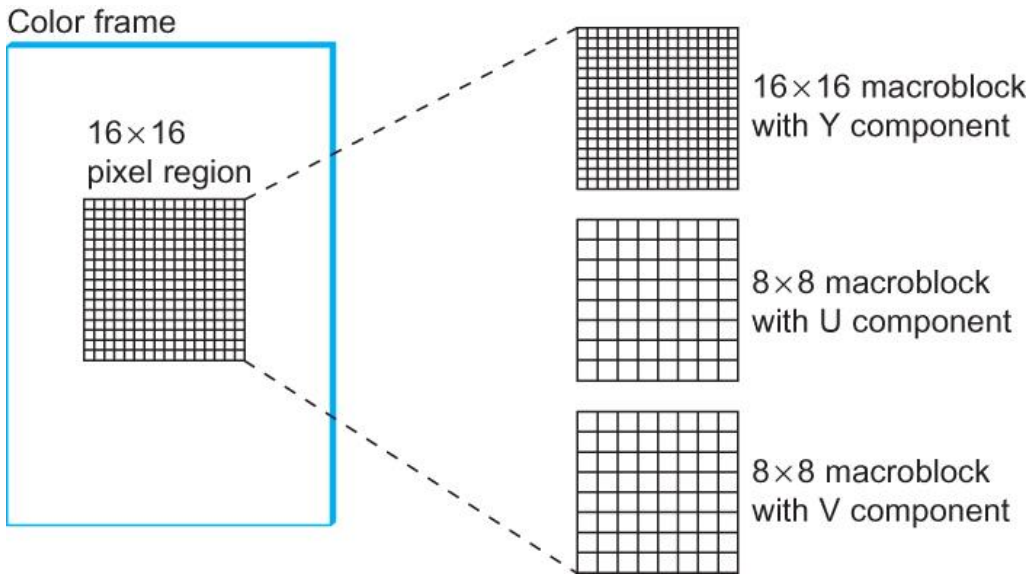
Video Compression (MPEG)

Frame Types

- MPEG takes a sequence of video frames as input and compresses them into three types of frames, called *I frames (intrapicture), P frames (predicted picture), and B frames* (bidirectional predicted picture).
- Each frame of input is compressed into one of these three frame types. I frames can be thought of as reference frames; they are self-contained, depending on neither earlier frames nor later frames.

Multimedia Data

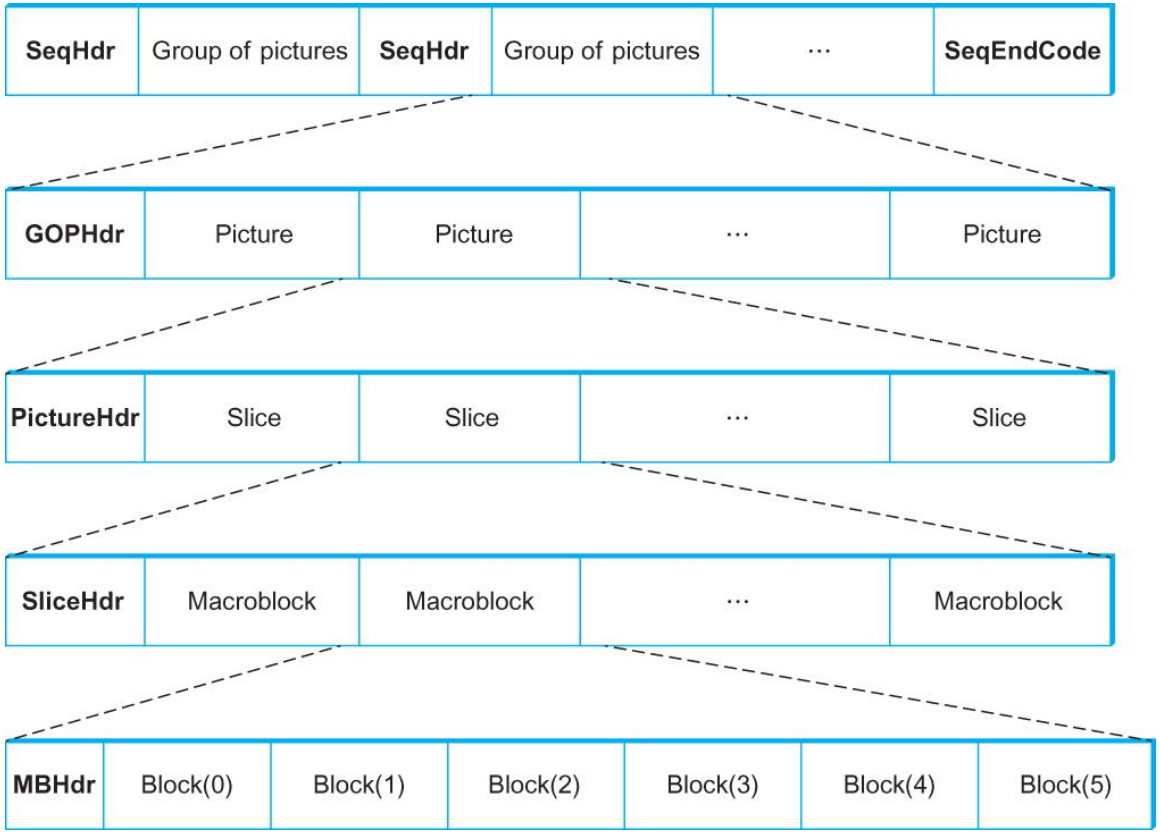
- Video Compression (MPEG)



Each frame as a collection of macroblocks

Multimedia Data

- Video Compression (MPEG)



Format of an MPEG-compressed video stream

Summary

- We have discussed how to represent how to represent data in the network
- We have discussed different compression techniques for handling multimedia data in the network

References

- Peterson, L.L., and Davie, B.S., Computer Networks: A Systems Approach Fifth Edition, Morgan Kaufmann, Burlington USA, 2012.
- David Salomon, Data Compression, The Complete Reference 4th Edition, 2007, SPRINGER.
- www.ietf.org
  - ✓ RFC 4506
  - ✓ RFC 6019
  - ✓ RFC 3023

THANK YOU

Gandeva Bayu Satrya (GBS)

gbs@ittelkom.ac.id

gandeva.bayu.s@gmail.com

TELKOM ENGINEERING SCHOOL

Telkom University