Chapter 4: The Presentation Layer (Layer 6)

The Presentation Layer is responsible for the encoding and decoding of information for representation on the network and eventual presentation at the destination. Think of it in terms of codecs (coders/decoders) that ensure data sent by an application on one system can be understood by the application on another system. This layer handles not just complex data types like audio and video, but also simpler data like plain text, ensuring compatibility between different systems and character sets.

Key Functions and Concepts:

1. Representing Text

- **Character Encoding:** Assigns a numerical value to each character. Common examples include:
    - **ASCII (American Standard Code for Information Interchange):** Represents 'A' as 65, 'B' as 66, etc.. It traditionally uses 7 bits but often occupies 8 bits in practice. Different versions of ASCII exist, which can lead to compatibility issues (e.g., the British Pound sign (£) is not in the American version).
    - **EBCDIC (Extended Binary Coded Decimal Interchange Code):** Another code, mainly used on mainframes, where 'A' is 193, 'B' is 194, etc..
    - **Unicode:** Aims to represent characters from almost all languages. 'A' is 65 in Unicode, similar to ASCII, but the number of bits used can vary.
        - **UTF-8:** A widely used Unicode encoding that is backward compatible with ASCII. ASCII characters use 1 octet (8 bits) starting with a '0'. Other characters can use up to six octets.
            - **UTF-8 Encoding Pattern (First Octet):**
                - 0xxxxxxx: Single octet character (128 characters).
                - 110xxxxx: Followed by one octet (10xxxxxx). Represents 25×26=211=2048 characters.
                - 1110xxxx: Followed by two octets (10xxxxxx 10xxxxxx). Represents 24×26×26=216=65536 characters.
                - 11110xxx: Followed by three octets.
                - 111110xx: Followed by four octets.
                - 1111110x: Followed by five octets.
            - Trailing octets always start with 10xxxxxx.
            - UTF-8 is self-synchronizing, meaning if part of the data stream is missed, a system can find the start of the next character by skipping octets that begin with 10.
        - **UTF-16:** Represents characters using either two or four octets.
- **Dealing with Character Set Differences:**
    - **Translation:** If communicating systems use different character codes (e.g., ASCII

and EBCDIC), the presentation layer translates characters to a common network representation (like Unicode) and then to the destination system's native code.
    - **Agreement/Negotiation:**
        - **Standardization:** Agreeing on a universal standard like Unicode.
        - **Minimal Shared Set:** Using only characters common to all relevant codes (e.g., plain ASCII). The ASCII Ribbon Campaign promotes plain text email for this reason.
        - **Negotiation:** The communicating parties can negotiate the character set to use. For example, a web browser's HTTP request might include an Accept-Charset header specifying preferred encodings (e.g., iso-8859-1, utf-8, utf-16) and their quality of preference (e.g., *;q=0.1). The server then responds with the chosen character set in its header (e.g., charset=Shift_JIS).
- **Diacritical Marks (Accents):**
    - Extended ASCII versions use 8 bits, adding 128 characters, some for accented characters (e.g., 'ê' at position 136). This can cause sorting issues and misrepresentation if sender and receiver use different extended sets.
    - **Solutions:**
        1. Agree on a character set (as above).
        2. Use markup to indicate accented characters.
- **Markup Languages:**
    - Indicate how text should be processed or displayed, rather than encoding each variation as a unique character.
    - **HTML (HyperText Markup Language):**
        - Special characters: ê for 'ê', & for '&'.
        - Formatting: <u>underline</u>, bold, italic.
        - Semantic markup (separating intent from presentation): emphasized (often rendered as italic, but could be spoken with intonation). <h1>Heading 1</h1>.
        - **CSS (Cascading Style Sheets):** Separates style from content in HTML, allowing changes to look and feel by changing the CSS file.
    - **XML (Extensible Markup Language):** Primarily for software-to-software communication, allowing structured data exchange with defined tags (e.g., <date><year>2005</year>...</date>).
- **Grammar-based Specification:**
    - Define the syntax of messages or data structures.
    - **BNF (Backus-Naur Form) / EBNF (Extended BNF) / ABNF (Augmented BNF):** Notations used to describe the syntax of languages, including network protocols like HTTP. For example, an HTTP GET request structure is defined using ABNF.
        - Request-Line = Method SP Request-URI SP HTTP-Version CRLF.
    - **ASN.1 (Abstract Syntax Notation One):** A formal notation for describing data structures to be exchanged between systems. It's like a record/structure declaration in programming.

        - Uses basic types (strings, integers) and constructors (SEQUENCE, SET, CHOICE).
        - Example: Courses ::= SET { code VisibleString, courseName VisibleString, mark INTEGER }.
        - ASN.1 provides an *abstract* syntax; actual encoding is done by rules like:
            - **BER (Basic Encoding Rules):** Uses a type-length-value (TLV) structure.
            - **CER (Canonical Encoding Rules):** Similar to BER but with fewer options.
            - **DER (Distinguished Encoding Rules):** Similar to CER but uses fixed lengths where possible, improving efficiency.
            - **XER (XML Encoding Rules):** Encodes ASN.1 data using XML, making it human-readable but larger.
        - SNMP uses ASN.1 with BER encoding. X.509 certificates are defined in ASN.1.
- **Application-Oriented Content Representation:**
    - Specific formats for particular application domains.
    - **SOAP (Simple Object Access Protocol):** XML-based protocol for exchanging structured information in web services.
    - **XBRL (eXtensible Business Reporting Language):** XML-based for business reporting.
    - **EDI (Electronic Data Interchange):** Older standard for exchanging business documents (orders, invoices).

2. Non-Textual Data

- The same principles of identifying data type and encoding apply to images (JPEG, GIF, PNG), audio, video, etc..
- **Metadata and "Markup" for Non-Textual Data:**
    - **EXIF (Exchangeable image file format):** Metadata embedded in image files (camera settings, geo-tags). This is more like embedding than traditional text markup.
    - HTML <img> tag's ALT attribute provides textual description for images.
    - Overlaying maps on aerial photos, or sequencing MRI scan images.
- **Data Type Negotiation Levels (Conceptual):**
    1. Encoding (e.g., UTF-8 for text, JPEG for image).
    2. Data type/Category (e.g., text, image).
    3. Markup/Additional Info (e.g., XML tags, EXIF data).
    4. Structured Message Syntax (e.g., specific format for a request).
    5. Semantic Unit (e.g., a complete cheque, invoice).

3. MIME (Multipurpose Internet Mail Extensions)

- Originally for email, now used in HTTP and other protocols to specify content type and encoding.

- MIME data has a header and a payload.
- **Key MIME Headers:**
  - MIME-Version: 1.0.
  - Content-Type: Indicates the media type and subtype.
    - Examples: text/html, text/xml, image/jpeg, application/msword, multipart/mixed.
    - **Main Types:** application, audio, example, image, message, model, multipart, text, video.
    - Parameters can provide more detail, e.g., charset=us-ascii or charset=ISO-8859-1 for text/html. For multipart types, a boundary parameter specifies the delimiter string.
  - Content-Transfer-Encoding: Specifies how the content has been further encoded for transport, especially if the underlying mechanism has limitations (e.g., SMTP designed for 7-bit ASCII).
    - 7bit, 8bit, binary: Indicates no further encoding (identity transformation) but provides info about the content's nature.
    - base64: Encodes arbitrary 8-bit data into a 7-bit ASCII character stream. Increases length (e.g., 'abc' (3 bytes) becomes 'YWJj' (4 bytes)).
      - **Calculation Example:** Base64 encoding takes 3 bytes of input (24 bits) and represents them as 4 ASCII characters (4 * 6 bits = 24 bits, chosen from a 64-character set). So, the output is 4/3 times the size of the input, plus padding if needed.
    - quoted-printable: Encodes arbitrary data into 7-bit values, attempting to keep readable characters as they are. Non-printable characters or characters with special meaning (like '=') are encoded.
      - 'abc' remains 'abc'.
      - Bytes 1, 2, 3 become =01=02=03.
      - 'a=b' becomes a=3Db (where 3D is the hex ASCII for '=').

4. Examples of Content Transfer

- **Email with Attachment (Figure 4.4):**
  - Outer Content-Type: multipart/mixed; boundary="----=_NextPart_000_..." indicates the email has multiple, different parts.
  - First part is multipart/alternative; boundary="----=_NextPart_001_...", offering the same content in different formats.
    - Alternative 1: Content-Type: text/plain; charset="us-ascii"; Content-Transfer-Encoding: 7bit (plain text version).
    - Alternative 2: Content-Type: text/html; charset="us-ascii"; Content-Transfer-Encoding: quoted-printable (HTML version).
  - Second main part (the attachment): Content-Type: application/msword; name="A Pax voucherless letter.doc"; Content-Transfer-Encoding: base64.
- **WAP Page (Figure 4.5 & 4.6):**
  - Response can be Content-Type: text/html; charset=utf-8.

    - Another example: Content-Type: application/xhtml+xml; charset=UTF-8 with Content-Encoding: gzip indicating the content is compressed. Gzip-compressed data is binary.

5. Remaining Layer 6 Issues

- **Data Exchange Formats:** Facilitate data sharing between different applications or systems.
    - **Container Formats:** Wrappers around different data parts (e.g., WMA files wrapping audio and metadata like DRM info, song title). Important for streaming capabilities.
    - **Data Serialization:** Translating a data structure into a linear sequence of values for storage or network transmission, which can then be reconstructed. XML and ASN.1 can be seen in this category.
    - **Interchange Formats:** Agreed-upon formats for exchanging specific types of data (e.g., LDIF for LDAP directory data, CSV for spreadsheets).
- **Compression:** Reduces data size for faster transmission and efficient storage. The presentation layer can handle compression/decompression.
    - **Lossless Compression:** Original data can be perfectly restored (e.g., for financial data).
        - **Run-Length Encoding (RLE):** Replaces sequences of identical bytes/bits (runs) with a count and the repeated item.
            - Character-based: "Ng" followed by 28 spaces (Ng<ESC><28>[). This uses 5 bytes instead of 30.
            - Bit-based: String 000011100000111000000 (23 bits) assuming starts with '0' run, encoded as run lengths 4,3,5,3,6. If run lengths encoded with 4 bits each: 0100 0011 0101 0011 0110 (20 bits needed for lengths, plus bit for initial char, not a saving here).
        - **Lempel-Ziv (LZ) algorithms:** Find repetitive sequences in data and replace subsequent occurrences with pointers to the first occurrence (e.g., <ESC><offset><length>).
        - **Huffman Coding:** Uses variable-length codes; shorter codes for frequently occurring characters, longer codes for infrequent ones.
            - **Example Calculation:** Characters A (20%), B (10%), C (20%), D (15%), E (35%).
                1. Initial nodes: A(20) B(10) C(20) D(15) E(35).
                2. Combine B(10) and D(15) -> BD(25). Branches: B(0), D(1). Nodes: A(20) C(20) E(35) BD(25).
                3. Combine A(20) and C(20) -> AC(40). Branches: A(0), C(1). Nodes: E(35) BD(25) AC(40).
                4. Combine BD(25) and E(35) -> BDE(60). Branches: BD(0), E(1). Nodes: AC(40) BDE(60). (Note: book combines BD(25) and E(35) to get 60%, labels BD as 0, E as 1. Figure 4.9 shows this combination with E(35) on one side and BD(25) on other). The

text then says the resulting trees are E (35%), the BD tree (25%) and the AC tree (40%). It then combines the two lowest: 25% (BD) and 35% (E) to form a tree of 60%.

5. Combine AC(40) and BDE(60) -> ABCDE(100). Branches: AC(0), BDE(1).

- Resulting codes:
  - A: 00
  - B: 100
  - C: 01
  - D: 101
  - E: 11

    - **Lossy Compression:** Some information is lost; acceptable for multimedia where imperceptible to humans (e.g., JPEG).
- **Encryption:** Can be handled by the presentation layer to provide end-to-end encryption of application data. Lower layer headers remain unencrypted, allowing traffic analysis.
    - **Link Encryption (Layer 2):** Encrypts everything, including headers from higher layers. Message decrypted at each hop.
    - In TCP/IP, encryption often occurs at/above Layer 4 (e.g., TLS/SSL) or at Layer 3 (IPsec).
- **Utilities:**
    - Tools exist for encoding/decoding (e.g., mimencode for base64, quoted-printable). Online utilities should be used with caution for sensitive data.

This chapter emphasizes that while the Presentation Layer's functions are crucial, they are often integrated into the application layer in practice, especially in the TCP/IP model which doesn't explicitly define a separate presentation layer.