and application program 305b require the communication of data to each other during normal operation of the software system 300.

[0022] In general, application program 305a and application program 305b reside on different platforms, as represented in FIGURE 3. The platform where the application programs reside and are executed can be a computer system integrated into a local exchange, such as the local exchange 21 of FIGURE 1. Because the application programs 305a, 305b reside on different platforms, a communications channel 315 is needed to transmit data between the application programs 305a, 305b. The communications channel 315 can be implemented using, for example, an ISDN line, a carrier, or an Ethernet link.

[0023] Although the platforms running the application programs 305a, 305b are so connected that a sequence of bytes can be sent from one platform to another platform, the system designer must still decide on the format for the data to be exchanged between the platforms. A common solution has been to write a customized implementation for the data exchange format between the platform. However, in many cases, the platforms exchanging data are not made by the same manufacturers. This is especially likely in a telecommunications system where two communicating platforms can be located in different cities, countries, or even different continents. If the platforms are not made by the same manufacturer the platforms are likely to use different internal representations of data structures, such as integers, characters, etc. Thus, for example, the platform on which application program 305a resides might store integers in byte order from left to right (known as "big endian" integers) while the platform on which application program 305b resides may store integer in byte order from right to left. Similarly, numeric data may be internally stored in either American Standard Code Information Interchange (ASCII) or Extended Binary-Coded Decimal Interchange Code (EBCDIC) representational formats.

[0024] In order for platforms using different internal representations of data structures to communicate with each other, ASN.1 is used to form at the data transmitted across the communication channel 315. As discussed, ASN.1 provides a standardized set of rules for representing instances of data structures which can then be encoded into a stream of bytes according to a predefined set of encoding rules. Although the most widely used encoding rules are the "Basic Encoding Rules" or BER, it should be understood that other encoding rules, e.g., Packed Encoding Rules (PER), may also be used. In any event, an encoded stream of bytes representing an instance of a data structure is referred to in the art as an ASN.1 message.

[0025] With reference again to FIGURE 3, an ASN.1 Decoder/Encoder Program module (ADEP) 310a, 310b is inserted between the respective application programs 305a, 305b and the communications channel 315. The ADEPs 310a, 310b encode data structures from the respective application programs 305a, 305b according to the ASN.1 specification into an ASN.1 message for transmission across the communication channel 315. The ADEPs 310a, 310b also decode an ASN.1 message received on the communications channel 315 into a format understood by the respective application programs 305a, 305b and the platform on which the application program resides.

[0026] Although in a preferred embodiment of the application programs 305a, 305b and the ADEPs 310a, 310b are implemented as separate and distinct layers, it should be understood that in an alternative embodiment the application programs 305a, 305b and the respective ADEPs 310a, 310b, can be integrated into a singe module. The application programs 305a, 305b and the ADEPs 310a, 310b in the presently preferred embodiment of the present invention are separated because the application program and the ADEP can then be written concurrently. Additionally, changes in the ASN.1 specification can be made requiring modifications to only the ADEPs 310a, 310b. With reference again to FIGURE 3, the combination of the application program 305 and the ADEP 310 together form what is known as an application stack 302.

[0027] In the context of describing the present invention, it is important to briefly explain the underlying and associated technology involved.

Abstract Syntax Notation One

[0028] ASN.1 provides a means of describing data type as well as the values of these types in an abstract manner. ASN.1 includes a number of simple and structured built-in types which allows a user to define more complex types and associated data values by combining one or more of the built-in types. In addition, ASN.1 also provides a set of subtype constructors (e.g. value range and size constraints) to define types whose values are only a subset of the values of a parent type. Illustrative examples of simple built-in types in ASN.1 include the BOOLEAN type, the INTEGER type, the ENUMERATED type and the OCTET STRING type, while illustrative examples of built-in structured types include the SEQUENCE type, the SET type and the CHOICE type.

[0029] ASN.1 provides for two different syntax - the ASN.1 type notation and the ASN.1 value notation. The ASN.1 type notation is directed to the structure of the data type while ASN.1 value notation is directed to the value of an instance of the data type. The ASN.1 type notation and the ASN.1 value notation are best understood by considering a hypothetical personnel record as an illustrative example.