RFC1832

[Index | Search | What's New | Comments | Help]

Network Working Group
Request for Comments: 1832
Category: Standards Track

R. Srinivasan
Sun Microsystems
August 1995

XDR: External Data Representation Standard

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

ABSTRACT

This document describes the External Data Representation Standard (XDR) protocol as it is currently deployed and accepted.

TABLE OF CONTENTS

1. INTRODUCTION

XDR is a standard for the description and encoding of data. It is useful for transferring data between different computer architectures, and has been used to communicate data between such diverse machines as the SUN WORKSTATION*, VAX*, IBM-PC*, and Cray*. XDR fits into the ISO presentation layer, and is roughly analogous in purpose to X.409, ISO Abstract Syntax Notation. The major difference between these two is that XDR uses implicit typing, while X.409 uses explicit typing.

XDR uses a language to describe data formats. The language can only be used only to describe data; it is not a programming language. This language allows one to describe intricate data formats in a concise manner. The alternative of using graphical representations (itself an informal language) quickly becomes incomprehensible when faced with complexity. The XDR language itself is similar to the C language [1], just as Courier [4] is similar to Mesa. Protocols such as ONC RPC (Remote Procedure Call) and the NFS* (Network File System) use XDR to describe the format of their data.

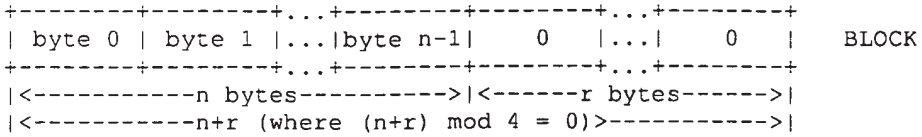
The XDR standard makes the following assumption: that bytes (or octets) are portable, where a byte is defined to be 8 bits of data. A given hardware device should encode the bytes onto the various media in such a way that other hardware devices may decode the bytes without loss of meaning. For example, the Ethernet* standard suggests that bytes be encoded in "little-endian" style [2], or least significant bit first.

2. BASIC BLOCK SIZE

The representation of all items requires a multiple of four bytes (or 32 bits) of data. The bytes are numbered 0 through n-1. The bytes are read or written to some byte stream such that byte m always precedes byte m+1. If the n bytes needed to contain the data are not a multiple of four, then the n bytes are followed by enough (0 to 3) residual zero bytes, r, to make the total byte count a multiple of 4.

We include the familiar graphic box notation for illustration and comparison. In most illustrations, each box (delimited by a plus sign at the 4 corners and vertical bars and dashes) depicts a byte.

Ellipses (...) between boxes show zero or more additional bytes where required.



3. XDR DATA TYPES

Each of the sections that follow describes a data type defined in the XDR standard, shows how it is declared in the language, and includes a graphic illustration of its encoding.

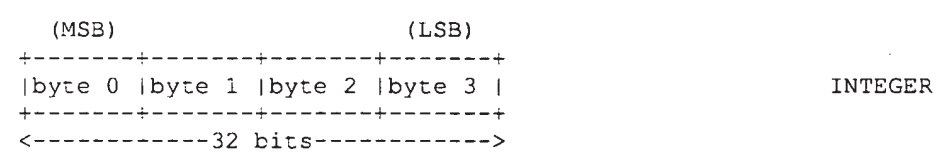
For each data type in the language we show a general paradigm declaration. Note that angle brackets (< and >) denote variablelength sequences of data and square brackets ([and]) denote fixed-length sequences of data. "n", "m" and "r" denote integers. For the full language specification and more formal definitions of terms such as "identifier" and "declaration", refer to section 5: "The XDR Language Specification".

For some data types, more specific examples are included. A more extensive example of a data description is in section 6: "An Example of an XDR Data Description".

3.1 Integer

An XDR signed integer is a 32-bit datum that encodes an integer in the range [-2147483648,2147483647]. The integer is represented in two's complement notation. The most and least significant bytes are 0 and 3, respectively. Integers are declared as follows:

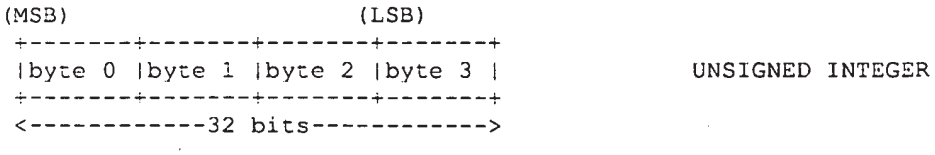
```
int identifier;
```



3.2. Unsigned Integer

An XDR unsigned integer is a 32-bit datum that encodes a nonnegative integer in the range [0,4294967295]. It is represented by an unsigned binary number whose most and least significant bytes are 0 and 3, respectively. An unsigned integer is declared as follows:

```
unsigned int identifier;
```



3.3 Enumeration

Enumerations have the same representation as signed integers. Enumerations are handy for describing subsets of the integers. Enumerated data is declared as follows:

```
enum { name-identifier = constant, ... } identifier;
```

For example, the three colors red, yellow, and blue could be described by an enumerated type:

```
enum { RED = 2, YELLOW = 3, BLUE = 5 } colors;
```

It is an error to encode as an enum any other integer than those that have been given assignments in the enum declaration.

3.4 Boolean

Booleans are important enough and occur frequently enough to warrant their own explicit type in the standard. Booleans are declared as follows:

```
bool identifier;
```

This is equivalent to:

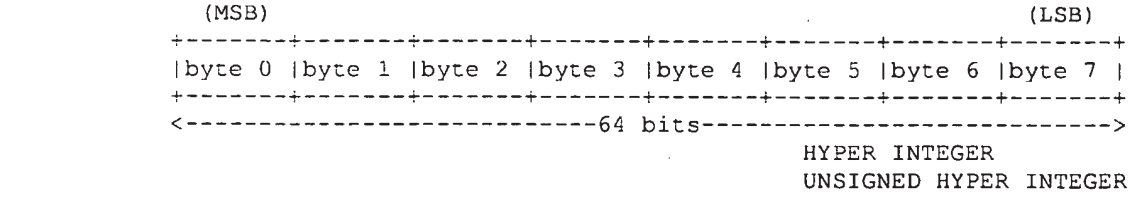
```
enum { FALSE = 0, TRUE = 1 } identifier;
```

3.5 Hyper Integer and Unsigned Hyper Integer

The standard also defines 64-bit (8-byte) numbers called hyper integer and unsigned hyper integer. Their representations are the obvious extensions of integer and unsigned integer defined above.

They are represented in two's complement notation. The most and least significant bytes are 0 and 7, respectively. Their declarations:

```
hyper identifier; unsigned hyper identifier;
```



3.6 Floating-point

The standard defines the floating-point data type "float" (32 bits or 4 bytes). The encoding used is the IEEE standard for normalized single-precision floating-point numbers [3]. The following three fields describe the single-precision floating-point number:

S: The sign of the number. Values 0 and 1 represent positive and negative, respectively. One bit.

E: The exponent of the number, base 2. 8 bits are devoted to this field. The exponent is biased by 127.

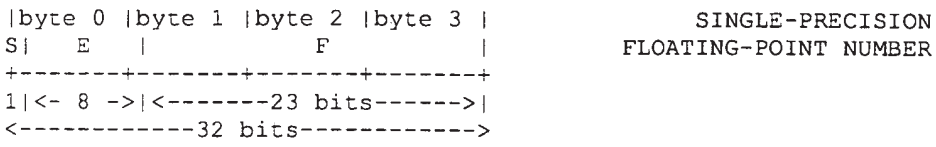
F: The fractional part of the number's mantissa, base 2. 23 bits are devoted to this field.

Therefore, the floating-point number is described by:

```
(-1)**S * 2**(E-Bias) * 1.F
```

It is declared as follows:

```
float identifier;
```



SINGLE-PRECISION FLOATING-POINT NUMBER

Just as the most and least significant bytes of a number are 0 and 3, the most and least significant bits of a single-precision floating-point number are 0 and 31. The beginning bit (and most significant bit) offsets of S, E, and F are 0, 1, and 9, respectively. Note that these numbers refer to the mathematical positions of the bits, and NOT to their actual physical locations (which vary from medium to medium).

The IEEE specifications should be consulted concerning the encoding for signed zero, signed infinity (overflow), and denormalized numbers (underflow) [3]. According to IEEE specifications, the "NaN" (not a number) is system dependent and should not be interpreted within XDR as anything other than "NaN".

3.7 Double-precision Floating-point

The standard defines the encoding for the double-precision floating-point data type "double" (64 bits or 8 bytes). The encoding used is the IEEE standard for normalized double-precision floating-point numbers [3]. The standard encodes the following three fields, which describe the double-precision floating-point number:

S: The sign of the number. Values 0 and 1 represent positive and negative, respectively. One bit.

E: The exponent of the number, base 2. 11 bits are devoted to this field. The exponent is biased by 1023.

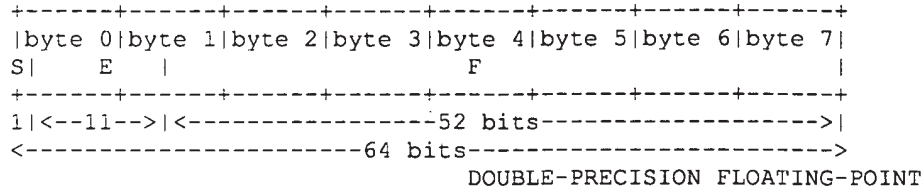
F: The fractional part of the number's mantissa, base 2. 52 bits are devoted to this field.

Therefore, the floating-point number is described by:

```
(-1)**S * 2**(E-Bias) * 1.F
```

It is declared as follows:

```
double identifier;
```



Just as the most and least significant bytes of a number are 0 and 3, the most and least significant bits of a double-precision floating-point number are 0 and 63. The beginning bit (and most significant bit) offsets of S, E , and F are 0, 1, and 12, respectively. Note

that these numbers refer to the mathematical positions of the bits, and NOT to their actual physical locations (which vary from medium to medium).

The IEEE specifications should be consulted concerning the encoding for signed zero, signed infinity (overflow), and denormalized numbers (underflow) [3]. According to IEEE specifications, the "NaN" (not a number) is system dependent and should not be interpreted within XDR as anything other than "NaN".

3.8 Quadruple-precision Floating-point

The standard defines the encoding for the quadruple-precision floating-point data type "quadruple" (128 bits or 16 bytes). The encoding used is designed to be a simple analog of of the encoding used for single and double-precision floating-point numbers using one form of IEEE double extended precision. The standard encodes the following three fields, which describe the quadruple-precision floating-point number:

S: The sign of the number. Values 0 and 1 represent positive and negative, respectively. One bit.

E: The exponent of the number, base 2. 15 bits are devoted to this field. The exponent is biased by 16383.

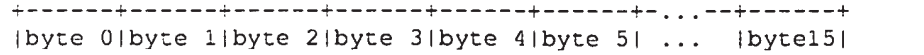
F: The fractional part of the number's mantissa, base 2. 112 bits are devoted to this field.

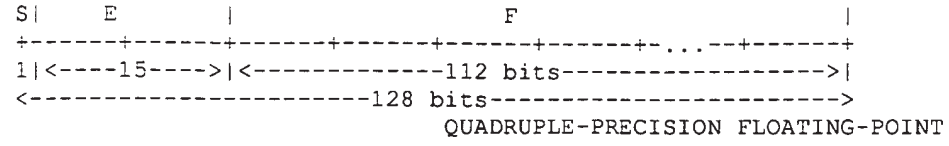
Therefore, the floating-point number is described by:

```
(-1)**S * 2**(E-Bias) * 1.F
```

It is declared as follows:

```
quadruple identifier;
```





QUADRUPLE-PRECISION FLOATING-POINT

Just as the most and least significant bytes of a number are 0 and 3, the most and least significant bits of a quadruple-precision floating-point number are 0 and 127. The beginning bit (and most

significant bit) offsets of S, E , and F are 0, 1, and 16, respectively. Note that these numbers refer to the mathematical positions of the bits, and NOT to their actual physical locations (which vary from medium to medium).

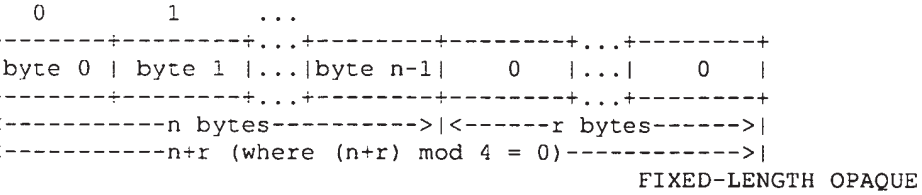
The encoding for signed zero, signed infinity (overflow), and denormalized numbers are analogs of the corresponding encodings for single and double-precision floating-point numbers [5], [6]. The "NaN" encoding as it applies to quadruple-precision floating-point numbers is system dependent and should not be interpreted within XDR as anything other than "NaN".

3.9 Fixed-length Opaque Data

At times, fixed-length uninterpreted data needs to be passed among machines. This data is called "opaque" and is declared as follows:

```
opaque identifier[n];
```

where the constant n is the (static) number of bytes necessary to contain the opaque data. If n is not a multiple of four, then the n bytes are followed by enough (0 to 3) residual zero bytes, r, to make the total byte count of the opaque object a multiple of four.



FIXED-LENGTH OPAQUE

3.10 Variable-length Opaque Data

The standard also provides for variable-length (counted) opaque data, defined as a sequence of n (numbered 0 through n-1) arbitrary bytes to be the number n encoded as an unsigned integer (as described below), and followed by the n bytes of the sequence.

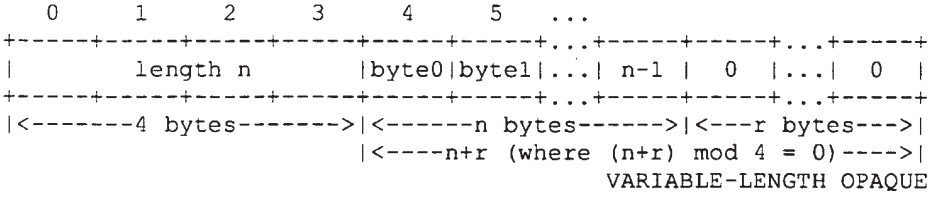
Byte m of the sequence always precedes byte m+1 of the sequence, and byte 0 of the sequence always follows the sequence's length (count). If n is not a multiple of four, then the n bytes are followed by enough (0 to 3) residual zero bytes, r, to make the total byte count a multiple of four. Variable-length opaque data is declared in the following way:

```
opaque identifier<m>;
```

```
   or
      opaque identifier<>;
```

The constant m denotes an upper bound of the number of bytes that the sequence may contain. If m is not specified, as in the second declaration, it is assumed to be (2**32) - 1, the maximum length. The constant m would normally be found in a protocol specification. For example, a filing protocol may state that the maximum data transfer size is 8192 bytes, as follows:

```
      opaque filedata<8192>;
```



It is an error to encode a length greater than the maximum described in the specification.

3.11 String

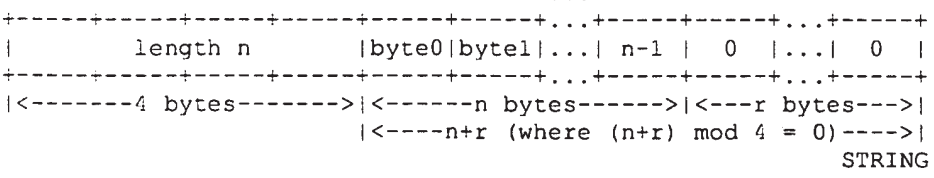
The standard defines a string of n (numbered 0 through n-1) ASCII bytes to be the number n encoded as an unsigned integer (as described above), and followed by the n bytes of the string. Byte m of the string always precedes byte m+1 of the string, and byte 0 of the string always follows the string's length. If n is not a multiple of four, then the n bytes are followed by enough (0 to 3) residual zero bytes, r, to make the total byte count a multiple of four. Counted byte strings are declared as follows:

```
      string object<m>;
   or
      string object<>;
```

The constant m denotes an upper bound of the number of bytes that a string may contain. If m is not specified, as in the second

declaration, it is assumed to be (2**32) - 1, the maximum length. The constant m would normally be found in a protocol specification. For example, a filing protocol may state that a file name can be no longer than 255 bytes, as follows:

```
      string filename<255>;
```



It is an error to encode a length greater than the maximum described

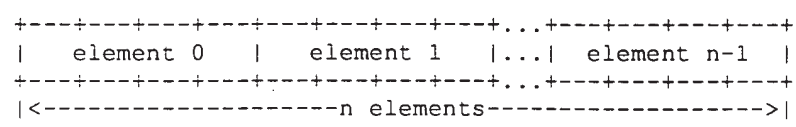
in the specification.

3.12 Fixed-length Array

Declarations for fixed-length arrays of homogeneous elements are in the following form:

```
type-name identifier[n];
```

Fixed-length arrays of elements numbered 0 through n-1 are encoded by individually encoding the elements of the array in their natural order, 0 through n-1. Each element's size is a multiple of four bytes. Though all elements are of the same type, the elements may have different sizes. For example, in a fixed-length array of strings, all elements are of type "string", yet each element will vary in its length.



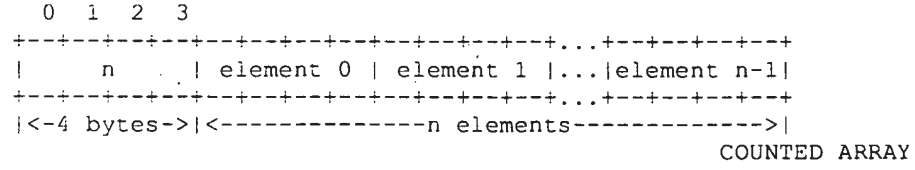
FIXED-LENGTH ARRAY

3.13 Variable-length Array

Counted arrays provide the ability to encode variable-length arrays of homogeneous elements. The array is encoded as the element count n (an unsigned integer) followed by the encoding of each of the array's elements, starting with element 0 and progressing through element n- 1. The declaration for variable-length arrays follows this form:

```
type-name identifier<m>;
```
or
```
type-name identifier<>;
```

The constant m specifies the maximum acceptable element count of an array; if m is not specified, as in the second declaration, it is assumed to be (2**32) - 1.



COUNTED ARRAY

It is an error to encode a value of n that is greater than the maximum described in the specification.

3.14 Structure

Structures are declared as follows:

```
struct {
   component-declaration-A;
   component-declaration-B;
```

```
   ...
} identifier;
```

The components of the structure are encoded in the order of their declaration in the structure. Each component's size is a multiple of four bytes, though the components may be different sizes.

```
+-------------+-------------+...
| component A | component B |...                    STRUCTURE
+-------------+-------------+...
```

3.15 Discriminated Union

A discriminated union is a type composed of a discriminant followed by a type selected from a set of prearranged types according to the value of the discriminant. The type of discriminant is either "int", "unsigned int", or an enumerated type, such as "bool". The component types are called "arms" of the union, and are preceded by the value of the discriminant which implies their encoding. Discriminated unions are declared as follows:

```
union switch (discriminant-declaration) {
case discriminant-value-A:

   arm-declaration-A;
case discriminant-value-B:
   arm-declaration-B;
...
default: default-declaration;
} identifier;
```

Each "case" keyword is followed by a legal value of the discriminant. The default arm is optional. If it is not specified, then a valid encoding of the union cannot take on unspecified discriminant values. The size of the implied arm is always a multiple of four bytes.

The discriminated union is encoded as its discriminant followed by the encoding of the implied arm.

```
  0   1   2   3
+---+---+---+---+---+---+---+---+
|  discriminant |  implied arm  |          DISCRIMINATED UNION
+---+---+---+---+---+---+---+---+
|<---4 bytes--->|
```

3.16 Void

An XDR void is a 0-byte quantity. Voids are useful for describing operations that take no data as input or no data as output. They are also useful in unions, where some arms may contain data and others do not. The declaration is simply as follows:

```
void;
```

Voids are illustrated as follows:

```
  ++
```

```
  ||                                                    VOID
  ++
-->><-- 0 bytes
```

3.17 Constant

The data declaration for a constant follows this form:

```
const name-identifier = n;
```

"const" is used to define a symbolic name for a constant; it does not declare any data. The symbolic constant may be used anywhere a regular constant may be used. For example, the following defines a symbolic constant DOZEN, equal to 12.

```
const DOZEN = 12;
```

3.18 Typedef

"typedef" does not declare any data either, but serves to define new identifiers for declaring data. The syntax is:

```
typedef declaration;
```

The new type name is actually the variable name in the declaration part of the typedef. For example, the following defines a new type called "eggbox" using an existing type called "egg":

```
typedef egg eggbox[DOZEN];
```

Variables declared using the new type name have the same type as the new type name would have in the typedef, if it was considered a variable. For example, the following two declarations are equivalent in declaring the variable "fresheggs":

```
eggbox  fresheggs; egg     fresheggs[DOZEN];
```

When a typedef involves a struct, enum, or union definition, there is another (preferred) syntax that may be used to define the same type. In general, a typedef of the following form:

```
typedef <<struct, union, or enum definition>> identifier;
```

may be converted to the alternative form by removing the "typedef" part and placing the identifier after the "struct", "union", or "enum" keyword, instead of at the end. For example, here are the two ways to define the type "bool":

```
typedef enum {     /* using typedef */
   FALSE = 0,
   TRUE = 1
} bool;

enum bool {        /* preferred alternative */
   FALSE = 0,
   TRUE = 1
};
```

The reason this syntax is preferred is one does not have to wait until the end of a declaration to figure out the name of the new type.

3.19 Optional-data

Optional-data is one kind of union that occurs so frequently that we give it a special syntax of its own for declaring it. It is declared as follows:

```
type-name *identifier;
```

This is equivalent to the following union:

```
union switch (bool opted) {
case TRUE:
   type-name element;
case FALSE:
   void;
} identifier;
```

It is also equivalent to the following variable-length array declaration, since the boolean "opted" can be interpreted as the length of the array:

```
type-name identifier<1>;
```

Optional-data is not so interesting in itself, but it is very useful for describing recursive data-structures such as linked-lists and trees. For example, the following defines a type "stringlist" that encodes lists of arbitrary length strings:

```
struct *stringlist {
   string item<>;
   stringlist next;
};
```

It could have been equivalently declared as the following union:

```
union stringlist switch (bool opted) {
case TRUE:
   struct {
      string item<>;
      stringlist next;
   } element;
case FALSE:
   void;
};
```

or as a variable-length array:

```
struct stringlist<1> {

   string item<>;
   stringlist next;
};
```

Both of these declarations obscure the intention of the stringlist type, so the optional-data declaration is preferred over both of them. The optional-data type also has a close correlation to how recursive data structures are represented in high-level languages such as Pascal or C by use of pointers. In fact, the syntax is the same as that of the C language for pointers.

3.20 Areas for Future Enhancement

The XDR standard lacks representations for bit fields and bitmaps, since the standard is based on bytes. Also missing are packed (or binary-coded) decimals.

The intent of the XDR standard was not to describe every kind of data that people have ever sent or will ever want to send from machine to machine. Rather, it only describes the most commonly used data-types of high-level languages such as Pascal or C so that applications written in these languages will be able to communicate easily over some medium.

One could imagine extensions to XDR that would let it describe almost any existing protocol, such as TCP. The minimum necessary for this are support for different block sizes and byte-orders. The XDR discussed here could then be considered the 4-byte big-endian member of a larger XDR family.

4. DISCUSSION

(1) Why use a language for describing data? What's wrong with diagrams?

There are many advantages in using a data-description language such as XDR versus using diagrams. Languages are more formal than diagrams and lead to less ambiguous descriptions of data. Languages are also easier to understand and allow one to think of other issues instead of the low-level details of bit-encoding. Also, there is a close analogy between the types of XDR and a high-level language such as C or Pascal. This makes the implementation of XDR encoding and decoding modules an easier task. Finally, the language specification itself is an ASCII string that can be passed from machine to machine to perform on-the-fly data interpretation.

(2) Why is there only one byte-order for an XDR unit?

Supporting two byte-orderings requires a higher level protocol for determining in which byte-order the data is encoded. Since XDR is not a protocol, this can't be done. The advantage of this, though, is that data in XDR format can be written to a magnetic tape, for example, and any machine will be able to interpret it, since no higher level protocol is necessary for determining the byte-order.

(3) Why is the XDR byte-order big-endian instead of little-endian? Isn't this unfair to little-endian machines such as the VAX(r), which has to convert from one form to the other?

Yes, it is unfair, but having only one byte-order means you have to

be unfair to somebody. Many architectures, such as the Motorola 68000* and IBM 370*, support the big-endian byte-order.

(4) Why is the XDR unit four bytes wide?

There is a tradeoff in choosing the XDR unit size. Choosing a small size such as two makes the encoded data small, but causes alignment problems for machines that aren't aligned on these boundaries. A large size such as eight means the data will be aligned on virtually every machine, but causes the encoded data to grow too big. We chose four as a compromise. Four is big enough to support most architectures efficiently, except for rare machines such as the eight-byte aligned Cray*. Four is also small enough to keep the encoded data restricted to a reasonable size.

(5) Why must variable-length data be padded with zeros?

It is desirable that the same data encode into the same thing on all machines, so that encoded data can be meaningfully compared or checksummed. Forcing the padded bytes to be zero ensures this.

(6) Why is there no explicit data-typing?

Data-typing has a relatively high cost for what small advantages it may have. One cost is the expansion of data due to the inserted type fields. Another is the added cost of interpreting these type fields and acting accordingly. And most protocols already know what type they expect, so data-typing supplies only redundant information. However, one can still get the benefits of data-typing using XDR. One way is to encode two things: first a string which is the XDR data description of the encoded data, and then the encoded data itself. Another way is to assign a value to all the types in XDR, and then define a universal type which takes this value as its discriminant and for each value, describes the corresponding data type.

5. THE XDR LANGUAGE SPECIFICATION

5.1 Notational Conventions

This specification uses an extended Back-Naur Form notation for describing the XDR language. Here is a brief description of the notation:

(1) The characters '|', '(', ')', '[', ']', '"', and '*' are special. (2) Terminal symbols are strings of any characters surrounded by double quotes. (3) Non-terminal symbols are strings of non-special characters. (4) Alternative items are separated by a vertical bar ("|"). (5) Optional items are enclosed in brackets. (6) Items are grouped together by enclosing them in parentheses. (7) A '*' following an item means 0 or more occurrences of that item.

For example, consider the following pattern:

```
"a " "very" (", " "very")* [" cold " "and "]  " rainy "
("day" | "night")
```

An infinite number of strings match this pattern. A few of them are:

```
        "a very rainy day"
        "a very, very rainy day"
        "a very cold and  rainy day"
        "a very, very, very cold and  rainy night"
```

5.2 Lexical Notes

(1) Comments begin with '/*' and terminate with '*/'. (2) White space serves to separate items and is otherwise ignored. (3) An identifier is a letter followed by an optional sequence of letters, digits or underbar ('_'). The case of identifiers is not ignored. (4) A constant is a sequence of one or more decimal digits, optionally preceded by a minus-sign ('-').

5.3 Syntax Information

```
      declaration:
           type-specifier identifier
         | type-specifier identifier "[" value "]"
         | type-specifier identifier "<" [ value ] ">"
         | "opaque" identifier "[" value "]"
         | "opaque" identifier "<" [ value ] ">"
         | "string" identifier "<" [ value ] ">"
         | type-specifier "*" identifier
         | "void"

      value:
           constant
         | identifier

      type-specifier:
           [ "unsigned" ] "int"
         | [ "unsigned" ] "hyper"
         | "float"
         | "double"
         | "quadruple"
         | "bool"
         | enum-type-spec
         | struct-type-spec
         | union-type-spec
         | identifier

      enum-type-spec:
         "enum" enum-body

      enum-body:
         "{"
            ( identifier "=" value )
            ( "," identifier "=" value )*
         "}"

      struct-type-spec:
         "struct" struct-body

      struct-body:
         "{"
```

```
        ( declaration ";" )
        ( declaration ";" )*
    "}"

union-type-spec:
    "union" union-body

union-body:
    "switch" "(" declaration ")" "{"
        ( "case" value ":" declaration ";" )
        ( "case" value ":" declaration ";" )*
        [ "default" ":" declaration ";" ]
    "}"

constant-def:
    "const" identifier "=" constant ";"

type-def:
        "typedef" declaration ";"
    | "enum" identifier enum-body ";"
    | "struct" identifier struct-body ";"
    | "union" identifier union-body ";"

definition:
        type-def
    | constant-def

specification:
        definition *
```

5.4 Syntax Notes

(1) The following are keywords and cannot be used as identifiers: "bool", "case", "const", "default", "double", "quadruple", "enum", "float", "hyper", "opaque", "string", "struct", "switch", "typedef", "union", "unsigned" and "void".

(2) Only unsigned constants may be used as size specifications for arrays. If an identifier is used, it must have been declared previously as an unsigned constant in a "const" definition.

(3) Constant and type identifiers within the scope of a specification are in the same name space and must be declared uniquely within this scope.

(4) Similarly, variable names must be unique within the scope of struct and union declarations. Nested struct and union declarations create new scopes.

(5) The discriminant of a union must be of a type that evaluates to an integer. That is, "int", "unsigned int", "bool", an enumerated type or any typedefed type that evaluates to one of these is legal. Also, the case values must be one of the legal values of the discriminant. Finally, a case value may not be specified more than once within the scope of a union declaration.

6. AN EXAMPLE OF AN XDR DATA DESCRIPTION

Here is a short XDR data description of a thing called a "file", which might be used to transfer files from one machine to another.

```
const MAXUSERNAME = 32;     /* max length of a user name */
const MAXFILELEN = 65535;   /* max length of a file      */
const MAXNAMELEN = 255;     /* max length of a file name */

/*
 * Types of files:
 */
enum filekind {
   TEXT = 0,        /* ascii data */
   DATA = 1,        /* raw data   */
   EXEC = 2         /* executable */
};

/*
 * File information, per kind of file:
 */
union filetype switch (filekind kind) {
case TEXT:
   void;                            /* no extra information */
case DATA:
   string creator<MAXNAMELEN>;      /* data creator         */
case EXEC:
   string interpretor<MAXNAMELEN>; /* program interpretor  */
};

/*
 * A complete file:
 */
struct file {
   string filename<MAXNAMELEN>; /* name of file    */
   filetype type;               /* info about file */
   string owner<MAXUSERNAME>;   /* owner of file   */
   opaque data<MAXFILELEN>;     /* file data       */
};
```

Suppose now that there is a user named "john" who wants to store his lisp program "sillyprog" that contains just the data "(quit)". His file would be encoded as follows:

```
OFFSET  HEX BYTES       ASCII    COMMENTS
------  ---------       -----    --------
 0      00 00 00 09     ....     -- length of filename = 9
 4      73 69 6c 6c     sill     -- filename characters
 8      79 70 72 6f     ypro     -- ... and more characters ...
12      67 00 00 00     g...     -- ... and 3 zero-bytes of fill
16      00 00 00 02     ....     -- filekind is EXEC = 2
20      00 00 00 04     ....     -- length of interpretor = 4
24      6c 69 73 70     lisp     -- interpretor characters
28      00 00 00 04     ....     -- length of owner = 4
32      6a 6f 68 6e     john     -- owner characters
36      00 00 00 06     ....     -- length of file data = 6
40      28 71 75 69     (qui     -- file data bytes ...
44      74 29 00 00     t)..     -- ... and 2 zero-bytes of fill
```

7. TRADEMARKS AND OWNERS

SUN WORKSTATION	Sun Microsystems, Inc.
VAX	Digital Equipment Corporation
IBM-PC	International Business Machines Corporation
Cray	Cray Research
NFS	Sun Microsystems, Inc.
Ethernet	Xerox Corporation.
Motorola 68000	Motorola, Inc.
IBM 370	International Business Machines Corporation

APPENDIX A: ANSI/IEEE Standard 754-1985

The definition of NaNs, signed zero and infinity, and denormalized numbers from [3] is reproduced here for convenience. The definitions for quadruple-precision floating point numbers are analogs of those for single and double-precision floating point numbers, and are defined in [3].

In the following, 'S' stands for the sign bit, 'E' for the exponent, and 'F' for the fractional part. The symbol 'u' stands for an undefined bit (0 or 1).

For single-precision floating point numbers:

Type	S (1 bit)	E (8 bits)	F (23 bits)
signalling NaN	u	255 (max)	.0uuuuu---u (with at least one 1 bit)
quiet NaN	u	255 (max)	.1uuuuu---u
negative infinity	1	255 (max)	.000000---0
positive infinity	0	255 (max)	.000000---0
negative zero	1	0	.000000---0
positive zero	0	0	.000000---0

For double-precision floating point numbers:

Type	S (1 bit)	E (11 bits)	F (52 bits)
signalling NaN	u	2047 (max)	.0uuuuu---u (with at least one 1 bit)
quiet NaN	u	2047 (max)	.1uuuuu---u
negative infinity	1	2047 (max)	.000000---0
positive infinity	0	2047 (max)	.000000---0
negative zero	1	0	.000000---0
positive zero	0	0	.000000---0

For quadruple-precision floating point numbers:

Type	S (1 bit)	E (15 bits)	F (112 bits)
signalling NaN	u	32767 (max)	.0uuuuu---u (with at least one 1 bit)
quiet NaN	u	32767 (max)	.1uuuuu---u
negative infinity	1	32767 (max)	.000000---0
positive infinity	0	32767 (max)	.000000---0
negative zero	1	0	.000000---0
positive zero	0	0	.000000---0

Subnormal numbers are represented as follows:

Precision	Exponent	Value
Single	0	(-1)**S * 2**(-126) * 0.F
Double	0	(-1)**S * 2**(-1022) * 0.F
Quadruple	0	(-1)**S * 2**(-16382) * 0.F

APPENDIX B: REFERENCES

[1] Brian W. Kernighan & Dennis M. Ritchie, "The C Programming Language", Bell Laboratories, Murray Hill, New Jersey, 1978.

[2] Danny Cohen, "On Holy Wars and a Plea for Peace", IEEE Computer, October 1981.

[3] "IEEE Standard for Binary Floating-Point Arithmetic", ANSI/IEEE Standard 754-1985, Institute of Electrical and Electronics Engineers, August 1985.

[4] "Courier: The Remote Procedure Call Protocol", XEROX Corporation, XSIS 038112, December 1981.

[5] "The SPARC Architecture Manual: Version 8", Prentice Hall, ISBN 0-13-825001-4.

[6] "HP Precision Architecture Handbook", June 1987, 5954-9906.

[7] Srinivasan, R., "Remote Procedure Call Protocol Version 2", RFC 1831, Sun Microsystems, Inc., August 1995.

Security Considerations

Security issues are not discussed in this memo.

Author's Address

Raj Srinivasan
Sun Microsystems, Inc.
ONC Technologies
2550 Garcia Avenue
M/S MTV-5-40
Mountain View, CA 94043
USA

Phone: 415-336-2478
Fax: 415-336-6015
EMail: raj@eng.sun.com

---

t.120

A Primer on the T.120 Series Standard

DataBeam™

A PRIMER ON THE T.120 SERIES STANDARDS

The T.120 standard contains a series of communication and application protocols and services that provide support for real-time, multipoint data communications. These multipoint facilities are important building blocks for a whole new range of collaborative applications, including desktop data conferencing, multi-user applications, and multi-player gaming.

Broad in scope, T.120 is a comprehensive specification that solves several problems that have historically slowed market growth for applications of this nature. Perhaps most importantly, T.120 resolves complex technological issues in a manner that is acceptable to both the computing and telecommunications industries.

Established by the International Telecommunications Union (ITU), T.120 is a family of open standards that was defined by leading data communication practitioners in the industry. Over 100 key international vendors, including Apple, AT&T, British Telecom, Cisco Systems, Intel, MCI, Microsoft, and PictureTel, have committed to implementing T.120-based products and services.

Broad vendor support means that end users will be able to choose from a variety of interoperable products.

While T.120 has emerged as a critical element in the data communications landscape, the only information that currently exists on the topic is a weighty and complicated set of standards documents. This primer bridges this information gap by summarizing T.120's major benefits, fundamental architectural elements, and core capabilities.

KEY BENEFITS OF T.120

So why all the excitement about T.120? The bottom line is that it provides exceptional benefits to end users, vendors, and developers tasked with implementing real-time applications. The following list is a high-level overview of the major benefits associated with the T.120 standard:

Multipoint Data Delivery

T.120 provides an elegant abstraction for developers to create and manage a multipoint domain with ease. From an application perspective, data is seamlessly delivered to multiple parties in "realtime."

Interoperability

T.120 allows endpoint applications from multiple vendors to interoperate. T.120 also specifies how applications may interoperate with (or through) a variety of network bridging products and services that also support the T.120 standard.

Reliable Data Delivery

Error-corrected data delivery ensures that all endpoints will receive each data transmission.

Multicast Enabled Delivery

In muliticast enabled networks, T.120 can employ reliable (ordered, guaranteed) and unreliable delivery services. Unreliable data delivery is also available without multicast. By using multicast, the T.120 infrastructure reduces network congestion and improves performance for the end user. The T.120 infrastructure can use both unicast and multicast simultaneously, providing a flexible solution for mixed unicast and multicast networks. The Multicast Adaptation Protocol (MAP) is expected to be ratified in early 1998.

Network Transparency

Applications are completely shielded from the underlying data transport mechanism being used. Whether the transport is a high-speed LAN or a simple dial-up modem, the application developer is only concerned with a single, consistent set of application services.

Platform Independence

Because the T.120 standard is completely free from any platform dependencies, it will readily take advantage of the inevitable advances in computing technology. In fact, DataBeam's customers have already ported the T.120 source code easily from Windows to a variety of environments, including OS/2, MAC/OS, several versions of UNIX, and other proprietary real-time operating systems.

T.120 BENEFITS

- ✔ Multipoint Data Delivery
- ✔ Interoperability
- ✔ Reliable Data Delivery
- ✔ Multicast Enabled Delivery
- ✔ Network Transparency
- ✔ Platform Independence
- ✔ Network Independence
- ✔ Support for Varied Topologies
- ✔ Application Independence
- ✔ Scalability
- ✔ Co-existence with Other Standards
- ✔ Extendability

Network Independence

The T.120 standard supports a broad range of transport options, including the Public Switched Telephone Networks (PSTN or POTS), Integrated Switched Digital Networks (ISDN), Packet Switched Digital Networks (PSDN), Circuit Switched Digital Networks (CSDN), and popular local area network protocols (such as TCP/IP and IPX via reference protocol). Furthermore, these vastly different network transports, operating at different speeds, can easily co-exist in the same multipoint conference.

Support for Varied Topologies

Multipoint conferences can be set up with virtually no limitation on network topology. Star topologies, with a single Multipoint Control Unit (MCU) will be common early on. The standard also supports a wide variety of other topologies ranging from those with multiple, cascaded MCUs to topologies as simple as a daisy-chain. In complex multipoint conferences, topology may have a significant impact on efficiency and performance.

Application Independence

Although the driving market force behind T.120 was teleconferencing, its designers purposely sought to satisfy a much broader range of application needs. Today, T.120 provides a generic, real-time communications facility that can be used by many different applications. These applications include interactive gaming, virtual reality and simulations, real-time subscription news feeds, and process control applications.

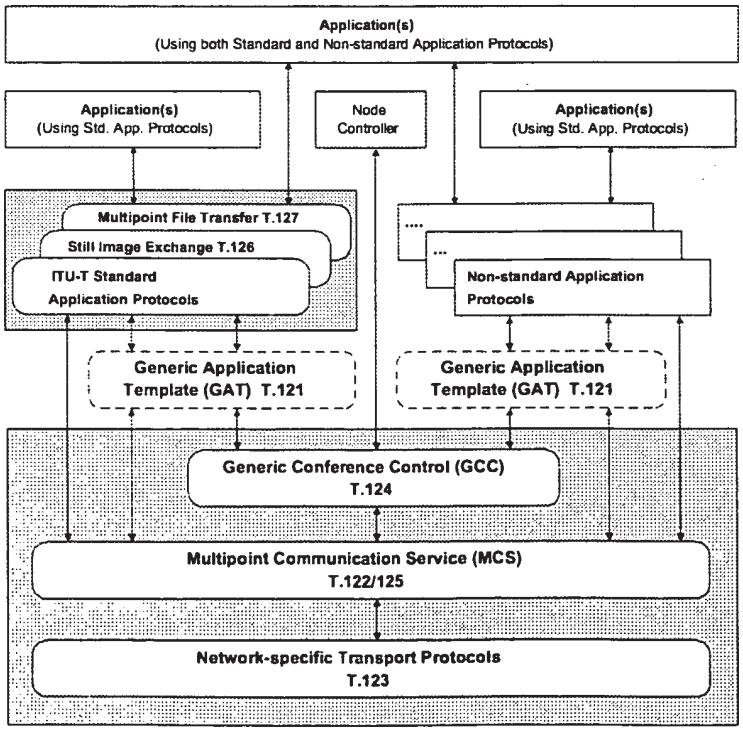
Scalability

T.120 is defined to be easily scalable from simple PC-based architectures to complex multi-processor environments characterized by their high performance. Resources for T.120 applications are plentiful, with practical limits imposed only by the confines of the specific platform running the software.

Co-existence with Other Standards

T.120 was designed to work alone or within the larger context of other ITU standards, such as the H.32x family of video conferencing standards. T.120 also supports and cross-references other important ITU standards, such as V.series modems.

FIGURE 1: MODEL OF ITU T.120 SERIES ARCHITECTURE



Extendability

The T.120 standard can be freely extended to include a variety of new capabilities, such as support for new transport stacks (like ATM or Frame Relay), improved security measures, and new application-level protocols.

Architectural Overview

The T.120 architecture relies on a multi-layered approach with defined protocols and service definitions between layers. Each layer presumes that all layers exist below. Figure 1 provides a graphical representation of the T.120 architecture.

The lower level layers (T.122, T.123, T.124, and T.125) specify an application-independent mechanism for providing multipoint data communication services to any application that can use these facilities. The upper level layers (T.126 and T.127) define protocols for specific conferencing applications, such as shared whiteboarding and multipoint file transfer. Applications using these standardized protocols can co-exist in the same conference with applications using proprietary protocols. In fact, a single application may even use a mix of standardized and non-standardized protocols.

Component Overview

The following overview describes the key characteristics and concepts behind each individual component of the T.120 standard. This overview starts at the bottom of the T.120 stack and progresses upward.

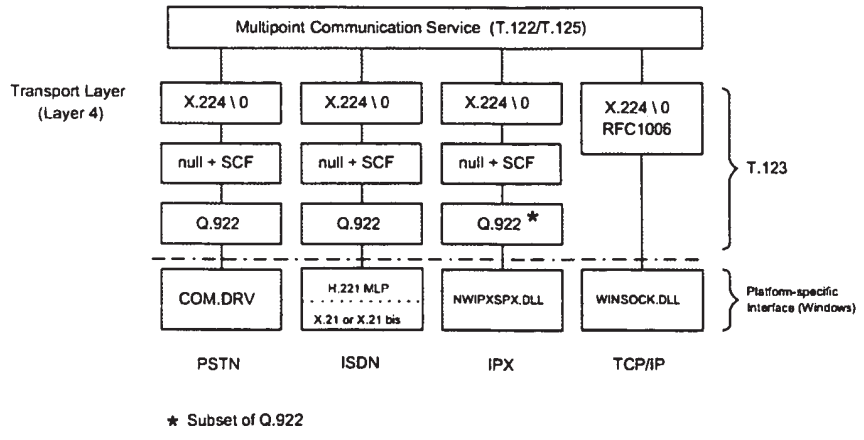
Transport Stacks - T.123

T.120 applications expect the underlying transport to provide reliable delivery of its Protocol Data Units (PDUs) and to segment and sequence that data. T.123 specifies transport profiles for each of the following:

- Public Switched Telephone Networks (PSTN)
- Integrated Switched Digital Networks (ISDN)
- Circuit Switched Digital Networks (CSDN)
- Packet Switched Digital Networks (PSDN)
- TCP/IP
- Novell Netware IPX (via reference profile)

As highlighted below in Figure 2, the T.123 layer presents a uniform OSI transport interface and services (X.214/X.224)

Figure 2: Cross-section of T.123 Transports (Basic Mode Profiles)



to the MCS layer above. The T.123 layer includes built-in error correction facilities so application developers do not have to rely on special hardware facilities to perform this function.

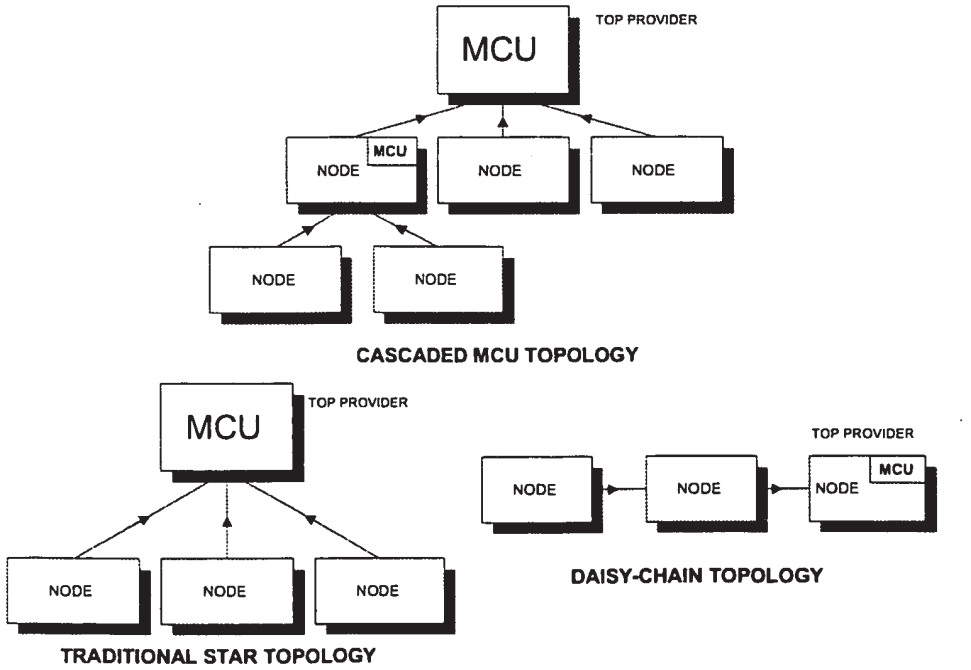
In a given computing environment, a transport stack typically plugs into a local facility that provides an interface to the specific transport connection. For example, in the Windows environment, DataBeam's transport stacks plug into COMM.DRV for modem communications, WINSOCK.DLL for TCP/IP and UDP/IP communications, and NWIPXSPX. DLL for Novell IPX communications support.

The MCU is a logical construct whose role may be served by a node on a desktop or by special-purpose equipment within the network.

The Multicast Adaptation Protocol (MAP) service layer is a new extension to MCS. MAP manages unicast- and multicast-based transports. MAP can be used with any transport where multicast is available, such as IP networks. While multicast provides unreliable delivery, many applications using T.120 require reliable services. Developers can incorporate a variety of multicast error correction schemes into MAP, thereby selecting the scheme most closely aligned with their application.

In 1996, the ITU is expected to adopt extensions to support important new transport facilities, such as Asynchronous Transfer Mode (ATM) and H.324 POTS videophone. It is necessary to note that developers can easily produce a proprietary transport stack (supporting, for example, AppleTalk) that transparently uses the services above T.123. An important function of MCUs or T.120-enabled bridges, routers, or gateways is to provide transparent interworking across different network boundaries.

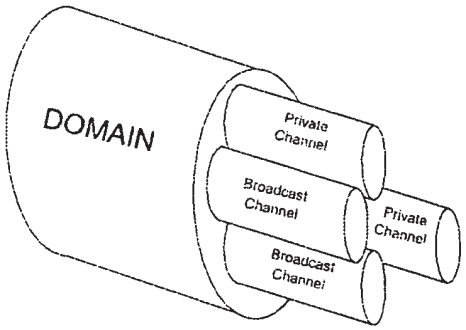
FIGURE 3: EXAMPLES OF VALID MCS TOPOLOGIES



Multipoint Communication Service (MCS) - T.122, T.125

T.122 defines the multipoint services available to the developer, while T.125 specifies the data transmission protocol. Together they form MCS, the multipoint "engine" of the T.120 conference. MCS relies on T.123 to deliver the data. (Use of MCS is entirely independent of the actual T.123 transport stack(s) that is loaded.)

FIGURE 4: CHANNEL DIAGRAM



MCS is a powerful tool that can be used to solve virtually any multipoint application design requirement. MCS is an elegant abstraction of a complex organism. Learning to use MCS effectively is the key to successfully developing real-time applications.

How MCS Works

In a conference, multiple endpoints (or MCS *nodes*) are logically connected together to form what T.120 refers to as a *domain*. Domains generally equate to the concept of a conference. An application may actually be *attached* to multiple domains simultaneously. For example, the chairperson of a large online conference may simultaneously monitor information being discussed among several activity groups.

In a T.120 conference, nodes connect upward to a Multipoint Control Unit (MCU). The MCU model in T.120 provides a reliable approach that works in both public and private networks. Multiple MCUs may be easily chained together in a single domain. Figure 3 illustrates three potential topology structures. Each domain has a single *Top Provider* or MCU that houses the information base critical to the conference. If the Top Provider either fails or leaves a conference, the conference is terminated. If a lower level MCU (i.e., not the Top Provider) fails, only the nodes on the tree below that MCU are dropped from the conference. Because all nodes contain MCS, they are all potentially "MCUs."

One of the critical features of the T.120 approach is the ability to direct data. This feature allows applications to communicate efficiently. MCS applications direct data within a domain via the use of *channels*. An application can choose to use multiple channels simultaneously for whatever purposes it needs (for example, separating annotation and file transfer operations). Application instances choose to obtain information by *subscribing* to whichever channel(s) contains the desired data. These channel assignments can be dynamically changed during the life of the conference. Figure 4 presents an overview of multiple channels in use within a domain.

It is the application developer's responsibility to determine how to use channels

TABLE 1: CHANNEL SETUP EXAMPLE

Channel	Type	Priority	Routing
1	Error Control Channels	Top	Standard
2	Annotations	High	Uniform
3	Bitmap Images	Medium	Uniform
4	File Transfer	Low	Standard

within an application. For example, an application may send control information along a single channel and application data along a series of channels that may vary depending upon the type of data being sent. The application developer may also take advantage of the MCS concept of *private channels* to direct data to a discrete subset of a given conference.

Data may be sent with one of four *priority* levels. MCS applications may also specify that data is routed along the quickest path of delivery using the *standard* send command. If the application uses the *uniform* send command, it ensures that data from multiple senders will arrive at all destinations in the same order. Uniform data always travels all the way up the tree to the Top Provider. Table 1 provides an example of how a document conferencing application could set up its channels. Reliable or unreliable data delivery is determined by the application.

There are no constraints on the size of the data sent from an application to MCS. Segmentation of data is automatically performed on behalf of the application. However, after receiving the data it is the application's responsibility to reassemble the data by monitoring flags provided when the data is delivered.

Tokens are the last major facility provided by MCS. Services are provided to grab, pass, inhibit, release, and query tokens. Token resources may be used as either exclusive (i.e., locking) or non-exclusive entities.

Tokens can be used by an application in a number of ways. For example, an application may specify that only the holder of a specific token, such as the conductor, may send information in the conference.

Another popular use of tokens is to coordinate tasks within a domain. For example, suppose a teacher wants to be sure that every student in a distance learning session answered a particular question before displaying the answer. Each node in the underlying application *inhibits* a specific token after receiving the request to answer the question. The token is released by each node when an answer is provided. In the background, the teacher's application continuously polls the state of the token. When all nodes have released the token, the application presents the teacher with a visual cue that the class is ready for the answer.

Generic Conference Control (GCC) - T.124

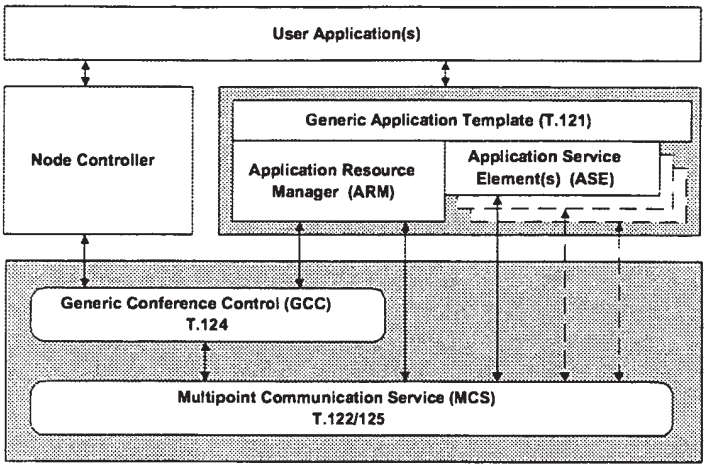
Generic Conference Control provides a comprehensive set of facilities for establishing and managing the multipoint conference. It is with GCC that we first see features that are specific to the electronic conference.

At the heart of GCC is an important information base about the state of the various conferences it may be servicing. One node, which may be the MCU itself, serves as the Top Provider for GCC information. Any actions or requests from lower GCC nodes ultimately filter up to this Top Provider.

One of GCC's most important roles is to maintain information about the nodes and applications that are in a conference.

Using mechanisms in GCC, applications create conferences, join conferences, and invite others to conferences. As endpoints join and leave conferences, the information base in GCC is updated and can be used to automatically notify all endpoints when these actions occur. GCC also knows who is the Top Provider for the conference. However, GCC does not contain detailed topology information about the means by which nodes from lower branches are connected to the conference.

FIGURE 5: T.121 GENERIC APPLICATION TEMPLATE



Every application in a conference must register its unique *application key* with GCC. This enables any subsequent joining nodes to find compatible applications. Furthermore, GCC provides robust facilities for applications to exchange capabilities and arbitrate feature sets. In this way, applications from different vendors can readily establish whether or not they can interoperate and at what feature level. This arbitration facility is the mechanism used to ensure backward compatibility between different versions of the same application.

GCC also provides conference security. This allows applications to incorporate password protection or "lock" facilities to prevent uninvited users from joining a conference.

Another key function of GCC is its ability to dynamically track MCS resources. Since multiple applications can use MCS at the same time, applications rely on GCC to prevent conflicts for MCS resources, such as channels and tokens. This ensures that applications do not step on each other by attaching to the same channel or requesting a token already in use by another application.

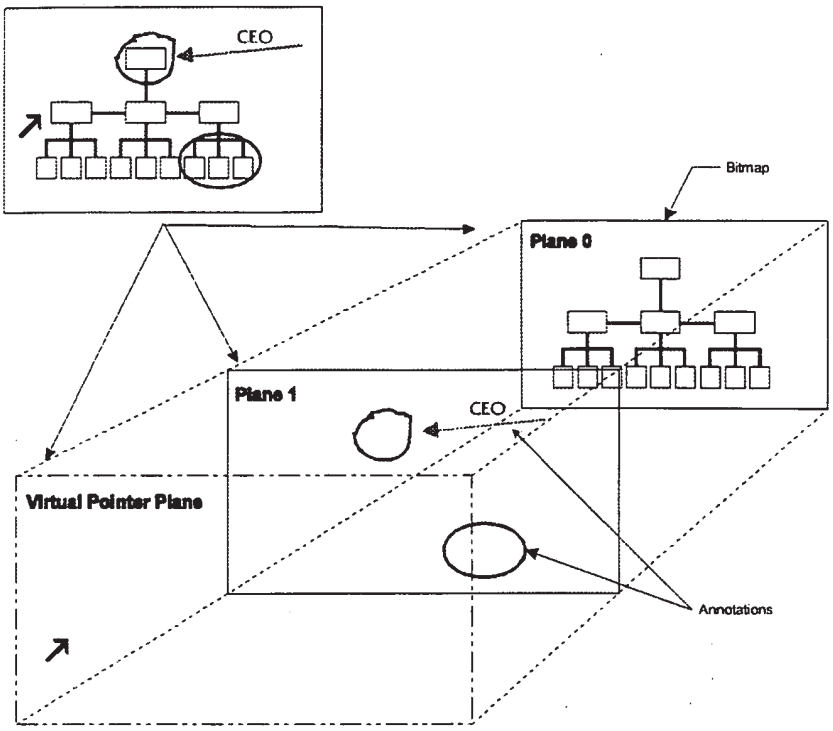
Finally, GCC provides capabilities for supporting the concept of *conductorship* in a conference. GCC allows the application to identify the conductor and a means in which to transfer the conductor's "baton." The developer is free to decide how to use these conductorship facilities within the application.

T.124 Revised

As part of the ongoing enhancement process for the T.120 standards, the ITU has completed a draft revision of T.124. The new version, called T.124 Revised, introduces a number of changes to improve scalability. The most significant changes address the need to distribute roster information to all nodes participating in a conference, as well as improvements in the efficiency of sending roster refresh information (from the Top Provider) any time a node joins or leaves a conference.

To improve the distribution of roster information, the concept of Node Categories was introduced. These categories provide a way for a T.124 node to join or leave a conference without affecting the roster information that was distributed throughout a conference. In addi-

FIGURE 6: T.126 WORKSPACE DIAGRAM



tion, the Full Roster Refresh, which was previously sent any time a new node joined a conference, was eliminated by sending out roster details from the Top Provder. These changes will not affect backward compatibility to earlier revisions of T.124. This revision will go to the ITU for Decision in March of 1998.

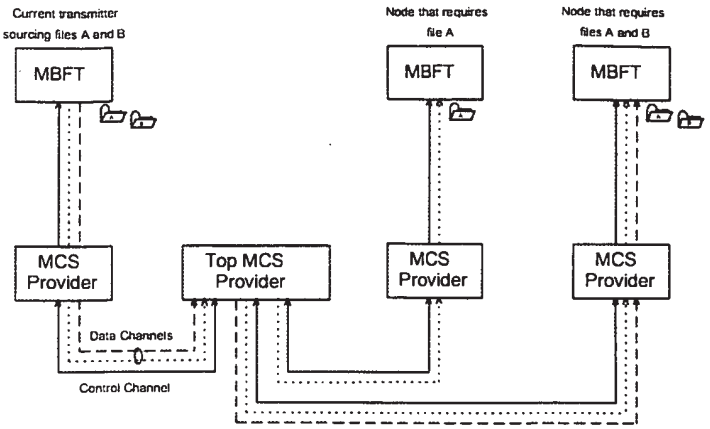
Generic Application Template (GAT) - T.121

T.121 provides a template for T.120 resource management that developers should use as a guide for building application protocols. T.121 is mandatory for standardized application protocols and is highly recommended for non-standard application protocols. The template ensures consistency and reduces the potential for unforeseen interaction between different protocol implementations.

Within the T.121 model, GAT defines a generic Application Resource Manager (ARM). This entity manages GCC and MCS resources on behalf of the application protocol-specific functionality defined as an Application Service Element (ASE). Figure 5 demonstrates the GAT model within the T.120 architecture. Simply put, GAT provides a consistent model for managing T.120 resources required by the application to which the developer adds application-specific functionality.

GAT's functionality is considered to be generic and common to all application protocols. GAT's services include enrolling the application in GCC and attaching to MCS domains. GAT also manages channels, tokens, and capabilities on behalf of the application. On a broader scale, GAT responds to GCC indica-

FIGURE 7: T.127 FILE TRANSFER MODEL



tions and can invoke peer applications on other nodes in the conference.

Still Image Exchange and Annotation (SI) - T.126

T.126 defines a protocol for viewing and annotating still images transmitted between two or more applications. This capability is often referred to as document conferencing or *shared whiteboarding.*

An important benefit of T.126 is that it readily shares visual information between applications that are running on dramatically different platforms. For example, a Windows-based desktop application could easily interoperate with a collaboration program running on a PowerMac. Similarly, a group-oriented conferencing system, without a PC-style interface, could share data with multiple users running common PC desktop software.

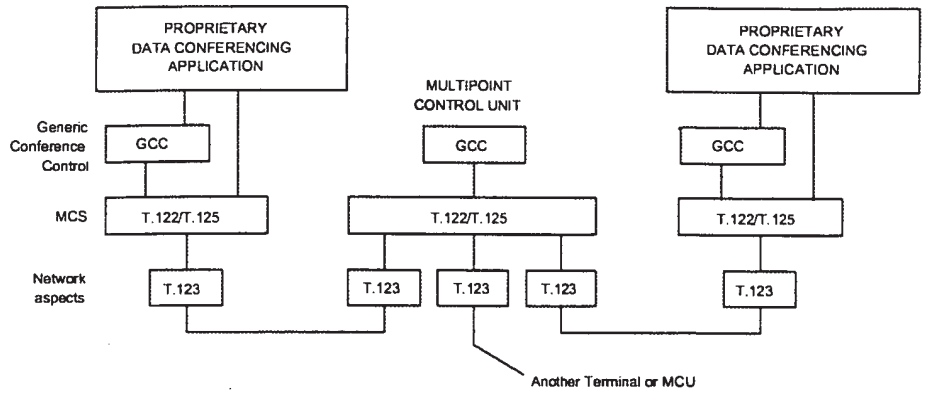
As Figure 6 illustrates, T.126 presents the concept of shared virtual *workspaces* that are manipulated by the endpoint applications. Each workspace may contain a collection of objects that include bitmap images and annotation primitives, such as rectangles and freehand lines. Bitmaps typically originate from application information, such as a word processing document or a presentation slide. Because of their size, bitmaps are often compressed to improve performance over lower-speed communication links.

T.126 is designed to provide a minimum set of capabilities required to share information between disparate applications. Because T.126 is simply a protocol, it does not provide any of the API-level structures that allow application developers to easily incorporate shared whiteboarding into an application. These types of facilities can only be found in toolkit-level implementations of the standard (such as DataBeam's Shared Whiteboard Application Toolkit, known as SWAT).

Multipoint Binary File Transfer - T.127

T.127 specifies a means for applications to transmit files between multiple endpoints in a conference. Files can be transferred to all participants in the conference or to a specified subset of the conference. Multiple file transfer operations may occur simultaneously in any given conference and developers can specify priority levels for the file delivery. Finally, T.127 provides options for compressing files before delivering the data. Figure 7 dis-

FIGURE 8: NETWORK-LEVEL INTEROPERABILITY DIAGRAM



plays a view of conference-wide and individual file transfers.

Node Controller

The Node Controller manages defined GCC Service Access Points (SAPs). This provides the node flexibility in responding to GCC events. Most of these GCC events relate to establishing conferences, adding or removing nodes from a conference, and breaking down and distributing information. The Node Controller's primary responsibility is to translate these events and respond appropriately.

Some GCC events can be handled automatically; for example, when a remote party joins a conference, each local Node Controller can post a simple message informing the local user that "Bill Smith has joined the conference." Other events may require user intervention; for example, when a remote party issues an invitation to join a conference, the local Node Controller posts a dialog box stating that "Mary Jones has invited you to the Design Review conference. <Accept> <Decline>."

Node controllers can be MCU-based, terminal-based, or dual-purpose. DataBeam's application, FarSite, for example, contains a dual-purpose Node Controller. The range of functionality found within a Node Controller can vary dramatically by implementation.

Only one Node Controller can exist on an active T.120 endpoint. Therefore, if multiple applications need to simultaneously use T.120 services, the Node Controller needs to be accessible to each application. The local interface to the Node Controller is application- and vendor-specific and is not detailed in the T.120 documentation.

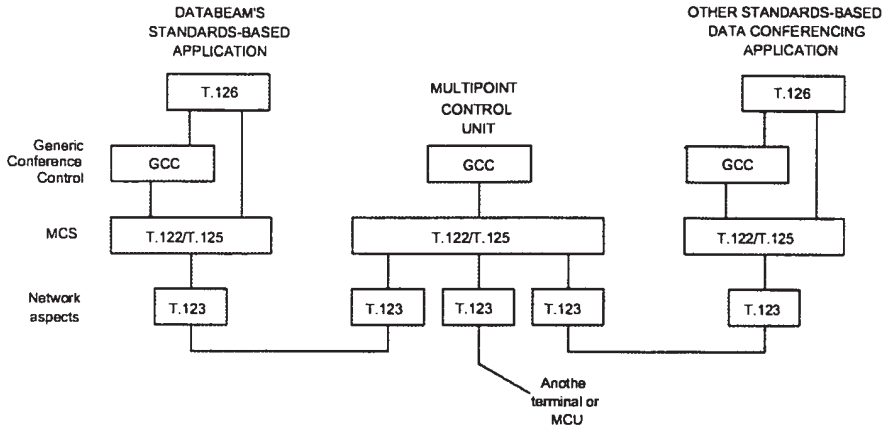
INTEROPERABILITY

Buyers overwhelmingly rate interoperability as the number one purchase criteria in their evaluation of teleconferencing products. For most end users, interoperability translates to "my application can talk to your application"—regardless of which vendor supplied the product or on what platform it runs. When examining the T.120 standard closely, buyers can see that it provides for two levels of interoperability: application-level interoperability and network-level interoperability.

Network-level Interoperability

Network-level interoperability means that a given product can interwork with like products through the infrastructure of

FIGURE 9: APPLICATION-LEVEL INTEROPERABILITY DIAGRAM



network products and services that support T.120. For example, T.120-based conferencing bridges (MCUs) that can support hundreds of simultaneous users are now being developed. If an application supports only the lower layers of T.120, customers can use these MCUs to host a multipoint conference only if everyone in the conference is using the exact same product. Figure 8 displays network interoperability through a conference of *like products.*

Application-level Interoperability

The upper levels of T.120 specify protocols for common conferencing applications, such as shared whiteboarding and binary file transfer. Applications supporting these protocols can interoperate with any other application that provides similar support, regardless of the vendor or platform used. For example, through T.126, users of DataBeam's FarSite application will be able to share and mark up documents with users of group conferencing systems. This interoperability will exist in simple point-to-point conferences as well as large multipoint conferences using a conference bridge. Figure 9 represents application-level interoperability between two standards-based applications connected in a conference.

In the short-term, network-level interoperability will be the most common form of T.120 support found in conferencing applications. This is largely due to the fact that the lower-level T.120 layers were ratified by the ITU more than a year in advance of the application-level layers. However, end users will not be satisfied with network interoperability alone. For the market to grow, vendors will have to deliver the same application-level interoperability (or endpoint interoperability) that customers enjoy today with fax machines and telephones.

RATIFICATION OF THE T.120 AND FUTURE T.130 STANDARDS

The Recommendations for the core multipoint communications infrastructure components (T.122, T.123, T.124 and T.125) were ratified by the ITU between March of 1993 and March of 1995. The first of the application standards (T.126 and T.127) was approved in

March of 1995. An overview of the T.120 series was approved in February of 1996 as Recommendation T.120. T.121 (GAT) was also approved at that time. Stable drafts of these recommendations existed for some time prior to the ratification, thereby providing a means for DataBeam to actively develop products in parallel to the standardization effort.

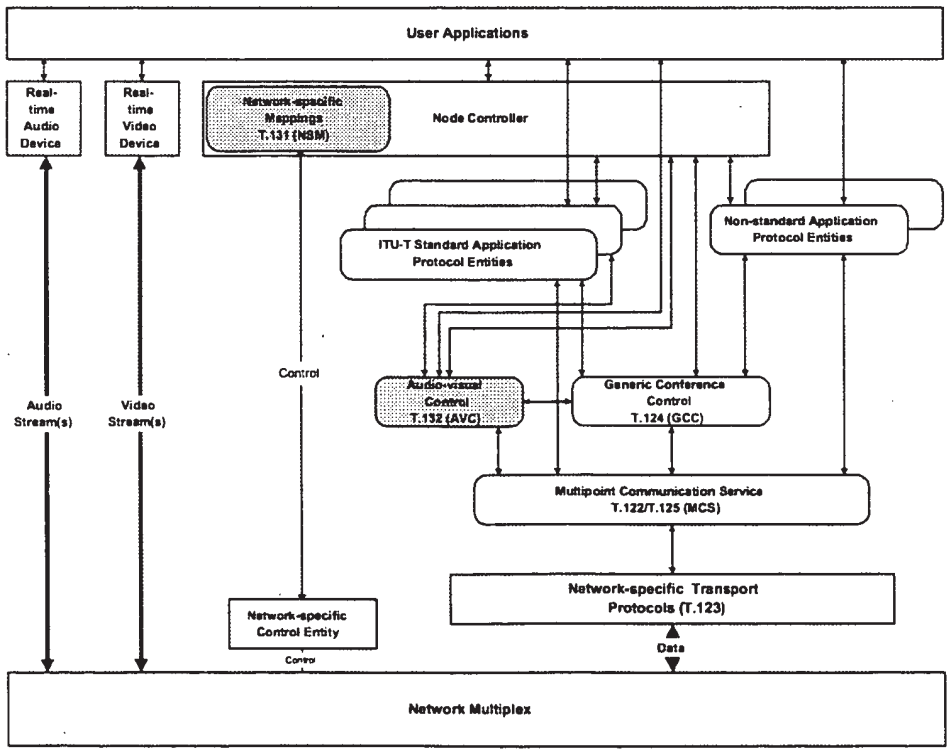
The existing ratified standards are being actively discussed for possible amendments and extensions. This commonly occurs when implementation and interoperability issues arise.

T.130 Audio-visual Control For Multimedia Conferencing

The T.130 series of recommendations define an architecture, a management and control protocol, and a set of services which together make up an Audio-Visual Control system (AVC). This system supports the use of real-time streams and services in a multimedia conferencing environment. The protocol and services section, outlined in T.132, consists of two parts: management and control. Together, they allow Network Elements, such as the traditional MCU, Gateway, or Conference Server, to provide T.132 audio and video services to their endpoints. Some of the services include Stream Identification, On-Air Identification, Video Switching, Audio Mixing, Remote Device Control, and Continuous Presence.

The T.130 series is built upon existing ITU-T conferencing recommendations such as the H.320 audio-visual conferencing series and the T.120 series for multipoint data conferencing. The T.130 series is compatible with systems, such as H.323, in which audio and video are

FIGURE 10: AUDIO-VISUAL CONTROL ARCHITECTURE



transmitted independently of T.120, as well as systems which are capable of transmitting multiple media types within a common multiplex.

Unlike other standardized methods for managing real-time streams within a conference, T.130 provides some unique capabilities:

- Contains a network- and platform-independent control protocol for managing real-time streams
- Coordinates operations across network boundaries
- Processes and distributes media streams within a conference environment
- Delivers of Quality of Service (QoS) to multimediacommunications applications
- Provides distributed conference management
- Leverages the functionality of existing multimedia protocols

T.130 can be used in any conferencing scenario where there is a need for multipoint audio or video. T.130 relies upon the services of GCC and MCS to transmit control data, but the audio and video streams are transported in independent logical channels due to the transmission requirements of real-time data flows. (See Figure 10).

T.130 and T.132 were determined in March of 1997 and should be ratified in January of 1998. T.131, which defines network-specific mappings to allow AVC to communicate with the underlying Multimedia Control Protocol, such as H.245, should be determined in the Fall of 1997.

Vendor Community Support for T.120

More than 100 multinational companies have pledged their support for the T.120 standard and more are being added to this list every week. Public supporters of T.120 include international market leaders, such as Apple, AT&T, British Telecom, Cisco Systems, Deutsche Telecom, IBM, Intel, MCI, Microsoft, Motorola, PictureTel, and DataBeam.

Most supporters of T.120 are also members of the International Multimedia Teleconferencing Consortium (IMTC). The goals of the IMTC are to promote the awareness and adoption of ITU teleconferencing standards, including T.120 and H.32x. The IMTC provides a forum for interoperability testing and helps to define Application Programming Interfaces (APIs). DataBeam's co-founder and chief technical officer, C. J. "Neil" Starkey, serves as the president of the IMTC. Previously, Starkey served for six years as chairman of the ITU study group that defined T.120.

New Markets for T.120 Deployment

The teleconferencing community is the first market segment to adopt the T.120 standard. Because the technology is broad in scope, it can be effectively used by a number of other application software vendors and equipment providers.

The computing paradigm is rapidly extending past today's personal productivity model. Over the next two years, we will witness the development of a new generation of application software that incorporates multi-party collaboration. Independent Software Vendors (ISVs) have begun to adopt T.120 as the means in which to incorporate real-time collabora-

tion capabilities into common desktop applications, such as word processing and presentation graphics. Engineering products, such as Computer Aided Design (CAD) software, are also on the migration path to T.120 technology. Other ISVs with a strong interest in T.120 include developers of fax, remote control, document imaging, and "overtime" collaboration products, such as Lotus Notes.

With T.120 technology in the hands of operating system providers and horizontal application vendors, network equipment providers are beginning to take notice. For vendors of PBXs, network bridges, hubs, routers and switches, T.120 represents an important opportunity to provide value-added capabilities within their network products. In the short-term, these features will represent an opportunity for competitive advantage. However, within the next year, T.120 support will be a required feature.

Finally, we can envision a whole range of T.120 applications in the areas of interactive video, network gaming, and simulations. From Nintendo to DOOM to set-top boxes, the need for bidirectional multipoint data communications is acute. The ability to use a common set of APIs and protocols that are broadly supported from the desktop through the network will drive the adoption of T.120 into these important emerging markets.

IMTC, ITU, AND T.120

Standards have played an important part in the establishment and growth of several consumer and telecommunications markets. By creating a basic commonality, standards ensure compatibility among products from different manufacturers. This encourages companies to produce varying solutions and encourages end users to purchase the solutions without fear of obsolescence or incompatibility.

The work of both the IMTC and the ITU represents organized efforts to promote a basic connectivity protocol that will encourage the growth of the multimedia telecommunications market. The Standards First™ initiative, which is supported by many industry leaders, requires a minimum of H.320 and T.120 compliance, which is enough to establish this basic connectivityprotocol. Manufacturers are then able to build on the basic compliance by adding features to their products, creating Standards Plus equipment.

With Standards First, the IMTC has the end users' interests in mind. By ensuring interoperability among equipment from competing manufacturers, Standards First also ensures that a customer's initial investment is protected and future system upgrades are possible. The IMTC is helping to educate the industry and the public about the importance, function, and status of standards. In addition, the organization provides a coordination point for industry leaders to communicate their interests to the ITU-T. As the multipoint multimedia teleconferencing industry continues its rapid growth, the development and implementation of standards for interoperability, and the work of the IMTC, will be instrumental in securing the market's future.

IMPLEMENTING T.120

With the T.120 set of standards in place, third-party developers are faced with yet another challenge— implementation. DataBeam's Collaborative Computing Toolkit Series (CCTS™) has jump-started the conferencing industry by providing the first standards-based toolkits for developing multipoint, data-sharing applications. These toolkits encapsulate the complex system-wide, multipoint communications stacks that allow application developers to rapidly embed sophisticated real-time, data-sharing capabilities into new or

existing products. Simply stated, CCTS provides a seamless solution for parties developing standards-based communication solutions.

As a result, DataBeam envisions an acceleration in the development of software applications and network infrastructure products such as, PBXs, bridges, routers, network switches, and LAN servers, that incorporate T.120. In addition, the industry will grow well beyond today's existing paradigms and the world will begin to see a whole range of new products and services that incorporate T.120. Users waiting for the standards dust to settle can now feel confident that with the support of vendors like Microsoft, DataBeam's T.120-based Collaborative Computing Toolkit Series is the best solution for industry-wide interoperability.

T.120 Information Sources

DataBeam Corporation
3191 Nicholasville Road
Lexington, Kentucky 40503
USA

Phone: (606) 245-3500
Fax: (606) 245-3528
E-Mail: info@databeam.com
Web Page: http://www.databeam.com

International Telecommunications Union
Sales Service
Place des Nations
CH-1211 Genéve 20
Switzerland

Phone: +41 22 730 6141
Fax: +41 22 730 5194
E-Mail: sales@itu.ch
Web Page: http://www.itu.ch

International Multimedia Teleconferencing Consortium, Inc.
111 Deerwood Road, Suite 372
San Ramon, California 94583
USA

Phone: (510) 743-4455
Fax: (510) 743-9011
E-Mail: dkamlani@imtc.fabrik.com
Web Page: http://www.imtc.org/imtc

Updated May 14,1997.

An Overview of TCP/IP Protocols and the Internet

Gary C. Kessler
Hill Associates, Inc.
kumquat@hill.com

23 April 1999

This paper was originally submitted to the InterNIC, and posted on their Gopher site, on 5 August 1994. This document is an updated version of that paper.

Contents

1. Introduction

An increasing number of people are using the Internet and, many for the first time, are using the tools and utilities that at one time were only available on a limited number of computer systems (and only for really intense users!). One sign of this growth in use has been the significant number of TCP/IP and Internet books, articles, courses, and even TV shows that have become available in the last several years; there are so many such books that publishers are reluctant to authorize more because bookstores have reached their limit of shelf space! This memo provides a broad overview of the Internet and TCP/IP, with an emphasis on history, terms, and concepts. It is meant as a brief guide and starting point, referring to many other sources for more detailed information.

2. What are TCP/IP and the Internet?

While the TCP/IP protocols and the Internet *are* different, their histories are most definitely intertwingled! This section wil discuss some of the history. For additional information and insight, readers are urged to read two excellent histories of the Internet: *Casting The Net: From ARPANET to INTERNET and beyond...* by Peter Salus (Addison-Wesley, 1995) and *Where Wizards Stay Up Late: The Origins of the Internet* by Katie Hafner and Mark Lyon (Simon & Schuster, 1997).

2.1. The Evolution of TCP/IP (and the Internet)

Prior to the 1960s, what little computer communication existed comprised simple text and binary data, carried by the mos common telecommunications network technology of the day; namely, circuit switching, the technology of the telephone networks for nearly a hundred years. Because most data traffic is bursty in nature (i.e., most of the transmissions occur during a very short period of time), circuit switching results in highly inefficient use of network resources. In 1962, Paul Baran, of the Rand Corporation, described a robust, efficient, store-and-forward data network in a report for the U.S. Air Force; Donald Davies suggested a similar idea in independent work for the Postal Service in the U.K., and coined the term *packet* for the data units that would be carried. According to Baran and Davies, packet switching networks could be designed so that all components operated independently, eliminating single point-of-failure problems. In addition, networ communication resources appear to be dedicated to individual users but, in fact, statistical multiplexing and an upper lim on the size of a transmitted entity result in fast, economical data networks.

The modern Internet began as a U.S. Department of Defense (DoD) funded experiment to interconnect DoD-funded research sites in the U.S. In December 1968, the Advanced Research Projects Agency (ARPA) awarded a contract to design and deploy a packet switching network to Bolt Beranek and Newman (BBN). In September 1969, the first node of the ARPANET was installed at UCLA. With four nodes by the end of 1969, the ARPANET spanned the continental U.S. by 1971 and had connections to Europe by 1973.

The original ARPANET gave life to a number of protocols that were new to packet switching. One of the most lasting results of the ARPANET was the development of a user-network protocol that has become the standard interface

between users and packet switched networks; namely, ITU-T (formerly CCITT) Recommendation X.25. This "standard" interface encouraged BBN to start Telenet, a commercial packet-switched data service, in 1974; after much renaming, Telenet is now a part of Sprint's X.25 service.

The initial host-to-host communications protocol introduced in the ARPANET was called the Network Control Protocol (NCP). Over time, however, NCP proved to be incapable of keeping up with the growing network traffic load. In 1974, a new, more robust suite of communications protocols was proposed and implemented throughout the ARPANET, based upon the Transmission Control Protocol (TCP) and Internet Protocol (IP). TCP and IP were originally envisioned functionally as a single protocol, thus the protocol suite, which actually refers to a large collection of protocols and applications, is usually referred to simply as *TCP/IP*. The original versions of both TCP and IP that are in common use today were written in September 1981, although both have had several modifications applied to them (in addition, the IP version 6, or IPv6, specification was released in December 1995). In 1983, the DoD mandated that all of their computer systems would use the TCP/IP protocol suite for long-haul communications, further enhancing the scope and importance of the ARPANET.

In 1983, the ARPANET was split into two components. One component, still called ARPANET, was used to interconnect research/development and academic sites; the other, called MILNET, was used to carry military traffic and became part of the Defense Data Network. That year also saw a huge boost in the popularity of TCP/IP with its inclusion in the communications kernel for the University of California s UNIX implementation, 4.2BSD (Berkeley Software Distribution) UNIX.

In 1986, the National Science Foundation (NSF) built a backbone network to interconnect four NSF-funded regional supercomputer centers and the National Center for Atmospheric Research (NCAR). This network, dubbed the NSFNET, was originally intended as a backbone for other networks, not as an interconnection mechanism for individual systems. Furthermore, the "Appropriate Use Policy" defined by the NSF limited traffic to non-commercial use. The NSFNET continued to grow and provide connectivity between both NSF-funded and non-NSF regional networks, eventually becoming the backbone that we know today as the Internet. Although early NSFNET applications were largely multiprotocol in nature, TCP/IP was employed for interconnectivity (with the ultimate goal of migration to Open Systems Interconnection).

The NSFNET originally comprised 56-kbps links and was completely upgraded to T1 (1.544 Mbps) links in 1989. Migration to a "professionally-managed" network was supervised by a consortium comprising Merit (a Michigan state regional network headquartered at the University of Michigan), IBM, and MCI. Advanced Network & Services, Inc. (ANS) a non-profit company formed by IBM and MCI, was responsible for managing the NSFNET and supervising the transition of the NSFNET backbone to T3 (44.736 Mbps) rates by the end of 1991. During this period of time, the NSF also funded a number of regional Internet service providers (ISPs) to provide local connection points for educational institutions and NSF-funded sites.

In 1993, the NSF decided that it did not want to be in the business of running and funding networks, but wanted instead to go back to the funding of research in the areas of supercomputing and high-speed communications. In addition, there was increased pressure to commercialize the Internet; in 1989, a trial gateway connected MCI, CompuServe, and Internet mail services, and commercial users were now finding out about all of the capabilities of the Internet that once belonged exclusively to academic and hard-core users! In 1991, the Commercial Internet Exchange (CIX) Association was formed by General Atomics, Performance Systems International (PSI), and UUNET Technologies to promote and provide a commercial Internet backbone service. Nevertheless, there remained intense pressure from non-NSF ISPs to open the network to all users.

In 1994, a plan was put in place to reduce the NSF's role in the public Internet. The new structure comprises three parts:

1. *Network Access Points (NAPs)*, where individual ISPs would interconnect. Although the NSF is only funding four such NAPs (Chicago, New York, San Francisco, and Washington, D.C.), several non-NSF NAPs are also in operation.
2. The *very High Speed Backbone Network Service*, a network interconnecting the NAPs and NSF-funded centers, operated by MCI. This network was installed in 1995 and operated at OC-3 (155.52 Mbps); it was completely upgraded to OC-12 (622.08 Mbps) in 1997.
3. The *Routing Arbiter*, to ensure adequate routing protocols for the Internet.

In addition, NSF-funded ISPs were given five years of reduced funding to become commercially self-sufficient. This funding ended by 1998.

In 1988, meanwhile, the DoD and most of the U.S. Government chose to adopt OSI protocols. TCP/IP was now viewed as an interim, proprietary solution since it ran only on limited hardware platforms and OSI products were only a couple of years away. The DoD mandated that all computer communications products would have to use OSI protocols by August 1990 and use of TCP/IP would be phased out. Subsequently, the U.S. Government OSI Profile (GOSIP) defined the set of protocols that would have to be supported by products sold to the federal government and TCP/IP was not included.

Despite this mandate, development of TCP/IP continued during the late 1980s as the Internet grew. TCP/IP developmen had always been carried out in an open environment (although the size of this open community was small due to the small number of ARPA/NSF sites), based upon the creed "We reject kings, presidents, and voting. We believe in rough consensus and running code" [*Dave Clark, M.I.T.*]. OSI products were still a couple of years away while TCP/IP became in the minds of many, the real open systems interconnection protocol suite.

It is not the purpose of this memo to take a position in the OSI vs. TCP/IP debate. Nevertheless, a number of observations are in order. First, the ISO Development Environment (ISODE) was developed in 1990 to provide an approach for OSI migration for the DoD. ISODE software allows OSI applications to operate over TCP/IP. During this same period, the Internet and OSI communities started to work together to bring about the best of both worlds as many TCP and IP features started to migrate into OSI protocols, particularly the OSI Transport Protocol class 4 (TP4) and the Connectionless Network Layer Protocol (CLNP), respectively. Finally, a report from the National Institute for Standards and Technology (NIST) in 1994 suggested that GOSIP should incorporate TCP/IP and drop the "OSI-only" requirement. [**NOTE:** Some industry observers have pointed out that OSI represents the ultimate example of a *sliding window*; OSI protocols have been "two years away" since about 1986.]

2.2. Internet Growth

The ARPANET started with four nodes in 1969 and grew to just under 600 nodes before it was split in 1983. The NSFNET also started with a modest number of sites in 1986. After that, the network has experienced literally exponentia growth. Internet growth between 1981 and 1991 is documented in "Internet Growth (1981-1991)" (RFC 1296).

Network Wizard's distributes a semi-annual Internet Domain Survey. According to them, the Internet had nearly 30 millio reachable hosts by January 1998. The Internet is growing at a rate of about a new network attachment every half-hour, interconnecting more than 200,000 networks. It is estimated that the Internet is doubling in size every ten to twelve months, and has been for the last several years.

And what of the original ARPANET? It grew smaller and smaller during the late 1980s as sites and traffic moved to the Internet, and was decommissioned in July 1990. Cerf & Kahn ("Selected ARPANET Maps," *Computer Communications Review*, October 1990) re-printed a number of network maps documenting the growth (and demise) of the ARPANET.

2.3. Internet Administration

The Internet has no single owner, yet everyone owns (a portion of) the Internet. The Internet has no central operator, yet everyone operates (a portion of) the Internet. The Internet has been compared to anarchy, but some claim that it is not nearly that well organized!

Some central authority is required for the Internet, however, to manage those things that can only be managed centrally, such as addressing, naming, protocol development, standardization, etc. Among the significant Internet authorities are:

- The Internet Society (ISOC), chartered in 1992, is a non-governmental international organization providing coordination for the Internet, and its internetworking technologies and applications. ISOC also provides oversight and communications for the Internet Activities Board.
- The Internet Activities Board (IAB) governs administrative and technical activities on the Internet.
- The Internet Engineering Task Force (IETF) is one of the two primary bodies of the IAB. The IETF's working groups have primary responsibility for the technical activities of the Internet, including writing specifications and protocols. The impact of these specifications is significant enough that ISO accredited the IETF as an internationa standards body at the end of 1994. RFCs 2028 and 2031 describe the organizations involved in the IETF standards process and the relationship between the IETF and ISOC, respectively, while RFC 2418 describes the IETF working group guidelines and procedures. The background and history of the IETF and the Internet standards process can be found in "IETF—History, Background, and Role in Today's Internet."
- The Internet Engineering Steering Group (IESG) is the other body of the IAB. The IESG provides direction to the IETF.
- The Internet Research Task Force (IRTF) comprises a number of long-term reassert groups, promoting research of importance to the evolution of the future Internet.
- The Internet Engineering Planning Group (IEPG) coordinates worldwide Internet operations. This group also assists Internet Service Providers (ISPs) to interoperate within the global Internet.
- The Forum of Incident Response and Security Teams is the coordinator of a number of Computer Emergency Response Teams (CERTs) representing many countries, governmental agencies, and ISPs throughout the world. Internet network security is greatly enhanced and facilitated by the FIRST member organizations.

2.4. Domain Names (and Politics)

Although not directly related to the administration of the Internet for operational purposes, the assignment of Internet domain names is the subject of some controversy and current activity. Internet hosts use a hierarchical naming structure comprising a top-level domain (TLD), domain and subdomain (optional), and host name. The IP address space (and all TCP/IP-related numbers) has historically been managed by the Internet Assigned Numbers Authority (IANA). Domain names are assigned by the TLD naming authority; until April 1998, the Internet Network Information Center (InterNIC) had overall authority of these names, with NICs around the world handling non-U.S. domains. The InterNIC was also responsible for the overall coordination and management of the Domain Name System (DNS), the distributed database that reconciles host names and IP addresses on the Internet.

The InterNIC is an interesting example of changes in the Internet. Starting in 1993, Network Solutions, Inc. (NSI) operated the InterNIC on behalf of the NSF and had exclusive registration authority for the *.com*, *.org*, *.net*, and *.edu* domains. NSI's contract ran out in April 1998 and was extended several times while everyone tried to determine who should pick up the registration for those domains. In October 1998, it was decided that NSI will remain the sole administrator for those domains but that users could register names in those domains with other firms. In addition, NSI's contract was extended to September 2000, although the registration business has to be opened to competition by June 1999.

Meanwhile, the newest body to handle gTLD registrations is the Internet Corporation for Assigned Names and Numbers (ICANN). Formed in October 1998, ICANN is the organization designated by the U.S. National Telecommunications and Information Administration (NTIA) to administer the DNS. Although still surrounded in some controversy (which is well beyond the scope of this paper!), ICANN has received wide industry support. ICANN will form several Support Organizations (SOs) to create policy for the administration of its areas of responsibility, including domain names (DNSO) IP addresses (ASO), and protocol parameter assignments (PSO).

On April 21, 1999, ICANN announced that five companies had been selected to be part of this new *competitive* Shared Registry System for the .com, .net, and .org domains:

- America Online, Inc. (U.S.)
- CORE (Internet Council of Registrars) (International)
- France Telecom/Oléane (France)
- Melbourne IT (Australia)
- register.com (U.S.)

Phase I of the competitive registrar testbed program will run until June, 1999. At that time, the Shared Registry System for the .com, .net, and .org domains will be opened to all ICANN-accredited registrars. ICANN also announced a list of 29 other applicant companies that had met its accreditation standards and will be able to enter the market as a registrar afte Phase I:

- 9 Net Avenue, Inc. (U.S.)
- A Technology Company (Canada)
- Active ISP (Norway)
- All West Communications (U.S.)
- Alldomains.com (U.S.)
- American Domain Name Registry (U.S.)
- AT&T (U.S.)
- Domain Direct (Canada)
- DomainRegistry.com (U.S.)
- eNom, Inc. (U.S.)
- Info Avenue Internet Services (U.S.)
- InfoNetworks (USA & United Kingdom)
- InfoRamp, Inc. (U.S.)
- Interactive Telecom Network, Inc. (U.S.)
- Interdomain, S.A. (Spain)
- Internet Domain Registrars (Canada)
- interQ Incorporated (Japan)
- MS Intergate, Inc. (U.S.)
- NameSecure.com (U.S.)
- Name.Space (U.S.)
- NetBenefit (United Kingdom)
- NetNames (United Kingdom)
- Nominalia (Catalonia)
- Port Information System (Sweden)
- RCN Corporation (U.S.)
- TelePartner AS (Denmark)
- Verio (U.S.)
- Virtual Internet (United Kingdom)
- WebTrends Corporation (U.S.)

The domain name structure is best understood if the name is read from right-to-left. Internet hosts names end with a top-level domain name. World-wide generic top-level domains include:

- *.com*: Commercial organizations (administered by the Shared Registry)
- *.edu*: Educational institutions, although today usually limited to 4-year colleges and universities (administered by the InterNIC)
- *.net*: Network providers (administered by the InterNIC and the Shared Registry)
- *.org*: Non-profit organizations (administered by the InterNIC and the Shared Registry)
- *.int*: Organizations established by international treaty
- *.gov*: U.S. Federal government agencies (delegated to the U.S. Federal Networking Council and administered by the InterNIC)

- *.mil*: U.S. military (managed by the U.S. Defense Data Network)

The host name *entc.tamu.edu*, for example, is assigned to a computer in the Engineering Technology and Industrial Distribution (ETID) Department at Texas A&M University (*tamu*), within the educational top-level domain (*edu*). The host name *golem.hill.com* refers to a host (*golem*) in the Hill Associates domain (*hill*) within the commercial top-level domain (*com*). Guidelines for selecting host names is the subject of RFC 1178.

Other top-level domain names use the two-letter country codes defined in ISO standard 3166; *munnari.oz.au*, for example, is the address of the Internet gateway to Australia and *myo.inst.keio.ac.jp* is a host at the Science and Technology Department of Keio University in Yokohama, Japan. Other ISO 3166-based domain country codes are *ca* (Canada), *de* (Germany), *es* (Spain), *fr* (France), *gb* (Great Britain) [**NOTE:** For some historical reasons, the TLD *.gb* is rarely used; the TLD *.uk* (United Kingdom) seems to be preferred although UK is not an official ISO 3166 country code.], *il* (Israel), *ie* (Ireland), *jp* (Japan), *mx* (Mexico), and *us* (United States). It is important to note that there is not necessarily any correlation between a country code and where a host is actually physically located.

The Western Hemisphere, European, and Asia-Pacific naming registries are managed by the American Registry for Internet Numbers (ARIN), RIPE, and Asia-Pacific NIC (APNIC), respectively. These authorities, in turn, delegate most of the country TLDs to national registries (such as RNP in Brazil and NIC-Mexico), which have ultimate authority to assign local domain names.

Different countries may organize the country-based subdomains in any way that they want. Many countries use a subdomain similar to the TLDs, so that *.com.mx* and *.edu.mx* are the suffixes for commercial and educational institutions in Mexico, and *.co.uk* and *.ac.uk* are the suffixes for commercial and educational institutions in the United Kingdom.

The *us* domain is largely organized on the basis of geography or function. Geographical names in the *us* name space us names of the form *entity-name.city-telegraph-code.state-postal-code*.us. The domain name *cnri.reston.va.us*, for example, refers to the Corporation for National Research Initiatives in Reston, Virginia. Functional branches are also reserved within the name space for schools (K12), community colleges (CC), technical schools (TEC), state government agencies (STATE), councils of governments (COG), libraries (LIB), museums (MUS), and several other generic types of entities. Domain names in the state government name space usually take the form *department*.state.*state-postal-code*.u (e.g., the domain name *dps.state.vt.us* points to the Vermont Department of Public Safety). The K12 name space can vary widely, usually using the form *school.school-district*.k12.*state-postal-code*.us (e.g., the domain *ccs.cssd.k12.vt.us* refers to the Charlotte Central School in the Chittenden South School District in Charlotte, Vermont.) More information about the *us* domain may be found in RFC 1480.

The scheme of TLD assignment and management has worked well for many years, but the pressures of increased commercial activity, network size, and international use have caused controversy about how names can be fairly assigned without violating trademarks and conflicting claims to names. In November 1996, an Internet International Ad Hoc Committee (IAHC) was formed to resolve some of these naming issues and to act as a focal point for the international debate over a proposal to establish additional global naming registries and global Top Level Domains (gTLDs). In February 1997, the IAHC proposed the creation of seven new gTLDs:

- *.firm* for businesses, or firms.
- *.store* for businesses offering goods to purchase.
- *.web* for entities emphasizing activities related to the WWW.
- *.arts* for entities emphasizing cultural and entertainment activities.
- *.rec* for entities emphasizing recreation/entertainment activities.
- *.info* for entities providing information services.

- *.nom* for those wishing individual or personal nomenclature.

The IAHC also proposed that up to 28 new registrars be established to grant second-level domain names under the new gTLDs, all of which will be shared among the new registrars. Furthermore, the three existing gTLDs .com, .net, and .org were also be shared upon conclusion of the NSF contract in the U.S. in 1998.

The IAHC was dissolved in May 1997 with the publication of the Generic Top Level Domain Memorandum of Understanding framework. The Council of Registrars (CORE) an operational body made up of all of the Registrars established under the gTLD-MoU framework.

3. The TCP/IP Protocol Architecture

TCP/IP is most commonly associated with the Unix operating system. While developed separately, they have been historically tied, as mentioned above, since 4.2BSD Unix started bundling TCP/IP protocols with the operating system. Nevertheless, TCP/IP protocols are available for all widely-used operating systems today and native TCP/IP support is provided in OS/2, OS/400, and Windows 95/98/NT, as well as most Unix variants.

Figure 1 shows the TCP/IP protocol architecture; this diagram is by no means exhaustive, but shows the major protocol and application components common to most commercial TCP/IP software packages and their relationship.

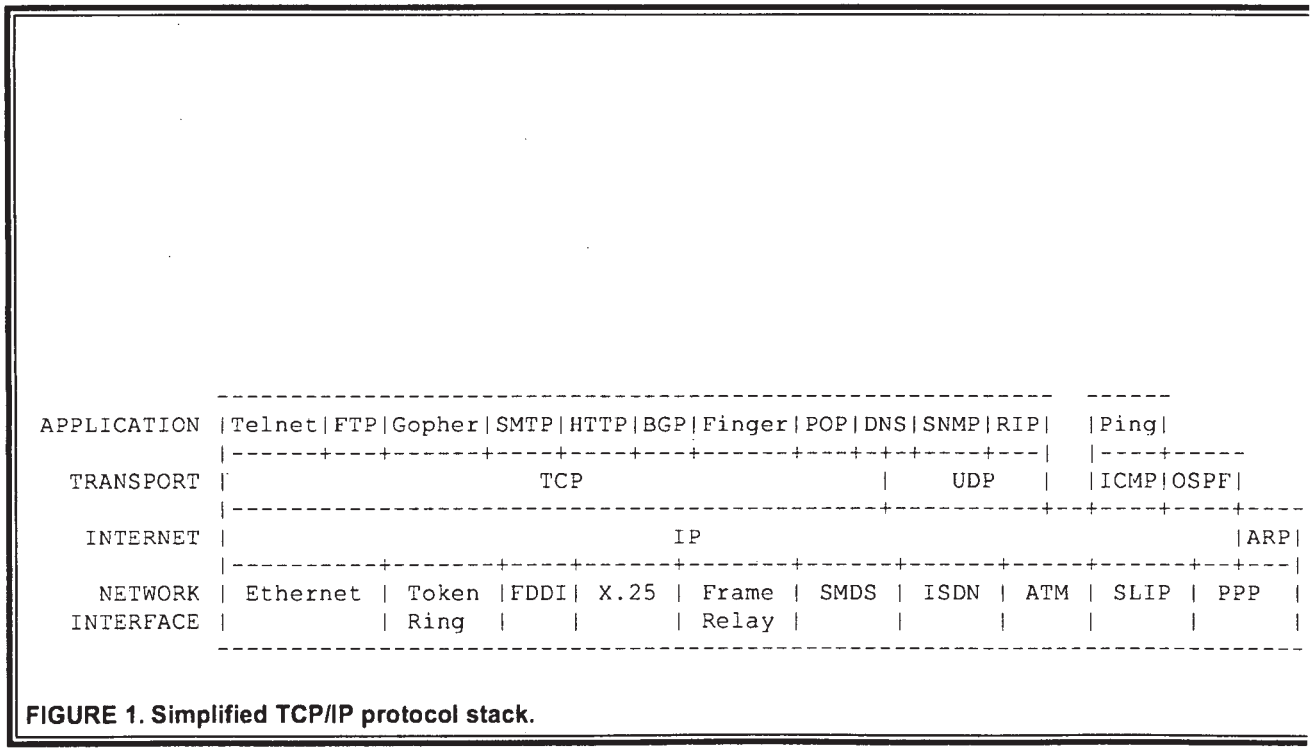


FIGURE 1. Simplified TCP/IP protocol stack.

The sections below will provide a brief overview of each of the layers in the TCP/IP suite and the protocols that compose those layers. A large number of books and papers have been written that describe all aspects of TCP/IP as a protocol suite, including detailed information about use and implementation of the protocols. Readers are referred to *Internetworking with TCP/IP, Vol. I: Principles, Protocols, and Architecture*, 2/e, by D. Comer (Prentice-Hall, 1991), *TCP/IP: Architecture, Protocols, and Implementation with IPv6 and IP Security*, 2nd. ed. by S. Feit (McGraw-Hill, 1997), "TCP/IP Tutorial" by T.J. Socolofsky and C.J. Kale (RFC 1180), and *TCP/IP Illustrated, Volume I: The Protocols* by W.R. Stevens (Addison-Wesley, 1994).

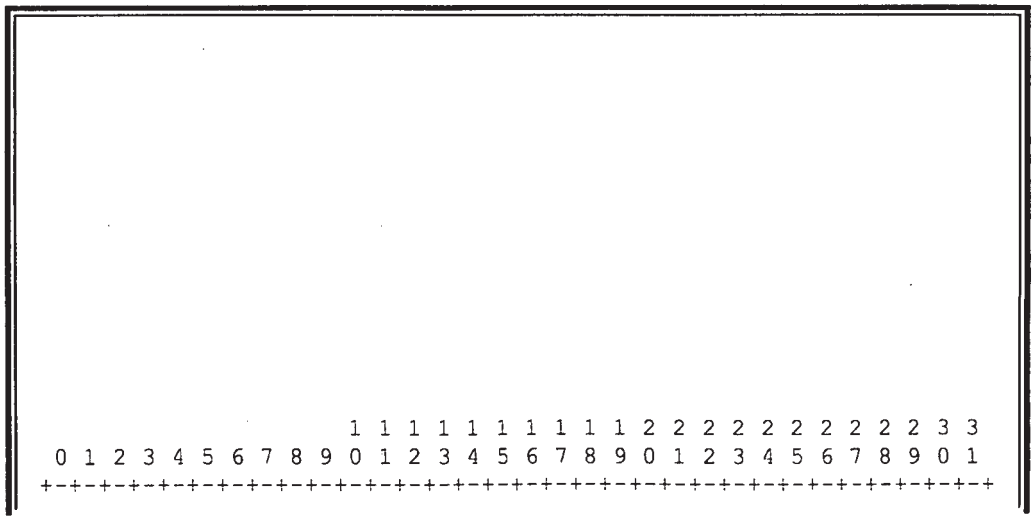
3.1. The Network Interface Layer

The TCP/IP protocols have been designed to operate over nearly any underlying local or wide area network technology. Although certain accommodations may need to be made, IP messages can be transported over all of the technologies shown in the figure, as well as numerous others.

Two of the underlying interface protocols are particularly relevant to TCP/IP. The Serial Line Internet Protocol (SLIP, RFC 1055) and Point-to-Point Protocol (PPP, RFC 1661), respectively, may be used to provide data link layer protocol services where no other underlying data link protocol may be in use, such as in leased line or dial-up environments. Mos commercial TCP/IP software packages for PC-class systems include these two protocols. With SLIP or PPP, a remote computer can attach directly to a host server and, therefore, connect to the Internet using IP rather than being limited to an asynchronous connection. PPP, in addition, provides support for simultaneous multiple protocols over a single connection (see the IANA list of PPP protocols), security mechanisms, and dynamic bandwidth allocation (e.g., when running over ISDN).

3.2. The Internet Layer

The Internet Protocol (RFC 791), provides services that are roughly equivalent to the OSI Network Layer. IP provides a datagram (connectionless) transport service across the network. This service is sometimes referred to as *unreliable* because the network does not guarantee delivery nor notify the end host system about packets lost due to errors or network congestion. IP datagrams contain a message, or one fragment of a message, that may be up to 65,535 bytes (octets) in length. IP does not provide a mechanism for flow control.



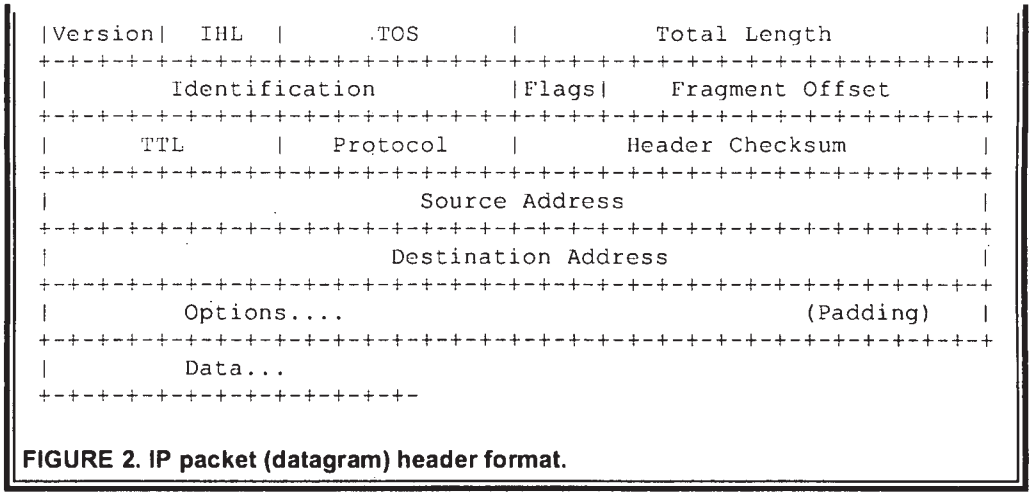


FIGURE 2. IP packet (datagram) header format.

The basic IP packet header format is shown in Figure 2. The format of the diagram is consistent with the RFC; bits are numbered from left-to-right, starting at 0. Each row represents a single *32-bit word*; note that an IP header will be at leas 5 words (20 bytes) in length. The fields contained in the header, and their functions, are:

- *Version:* Specifies the IP version of the packet. The current version of IP is version 4, so this field will contain the binary value `0100`. [NOTE: Actually, many IP version numbers have been assigned besides 4 and 6; see the IANA's list of IP Version Numbers.]
- *Internet Header Length (IHL):* Indicates the length of the datagram header in 32 bit (4 octet) words. A minimum-length header is 20 octets, so this field always has a value of at least 5 (`0101`).
- *Type of Service (TOS):* Allows an originating host to request different classes of service for packets it transmits. Although not generally supported today in IPv4, the TOS field can be set by the originating host in response to service requests across the Transport Layer/Internet Layer service interface, and can specify a service priority (0-7) or can request that the route be optimized for either cost, delay, throughput, or reliability.
- *Total Length:* Indicates the length (in bytes, or octets) of the entire packet, including both header and data. Given the size of this field, the maximum size of an IP packet is 64 KB, or 65,535 bytes. In practice, packet sizes are limited to the maximum transmission unit (MTU).
- *Identification:* Used when a packet is fragmented into smaller pieces while traversing the Internet, this identifier is assigned by the transmitting host so that different fragments arriving at the destination can be associated with each other for reassembly.
- *Flags:* Also used for fragmentation and reassembly. The first bit is called the More Fragments (MF) bit, and is use to indicate the last fragment of a packet so that the receiver knows that the packet can be reassembled. The second bit is the Don't Fragment (DF) bit, which suppresses fragmentation. The third bit is unused (and always se to 0).
- *Fragment Offset:* Indicates the position of this fragment in the original packet. In the first packet of a fragment stream, the offset will be 0; in subsequent fragments, this field will indicates the offset in increments of 8 bytes.
- *Time-to-Live (TTL):* A value from 0 to 255, indicating the number of hops that this packet is allowed to take before discarded within the network. Every router that sees this packet will decrement the TTL value by one; if it gets to 0 the packet will be discarded.
- *Protocol:* Indicates the higher layer protocol contents of the data carried in the packet; options include ICMP (1), TCP (6), UDP (17), or OSPF (89). A complete list of IP protocol numbers can be found at the IANA's list of Protocol Numbers.

- *Header Checksum:* Carries information to ensure that the received IP header is error-free. Remember that IP provides an *unreliable* service and, therefore, this field only checks the header rather than the entire packet.
- *Source Address:* IP address of the host sending the packet.
- *Destination Address:* IP address of the host intended to receive the packet.
- *Options:* A set of options which may be applied to any given packet, such as sender-specified source routing or security indication. The option list may use up to 40 bytes (10 words), and will be padded to a word boundary; IP options are taken from the IANA's list of IP Option Numbers.

3.2.1. IP Addresses

IP addresses are 32 bits in length (Figure 3). They are typically written as a sequence of four numbers, representing the decimal value of each of the address bytes. Since the values are separated by periods, the notation is referred to as *dotted decimal*. A sample IP address is 208.162.106.17.

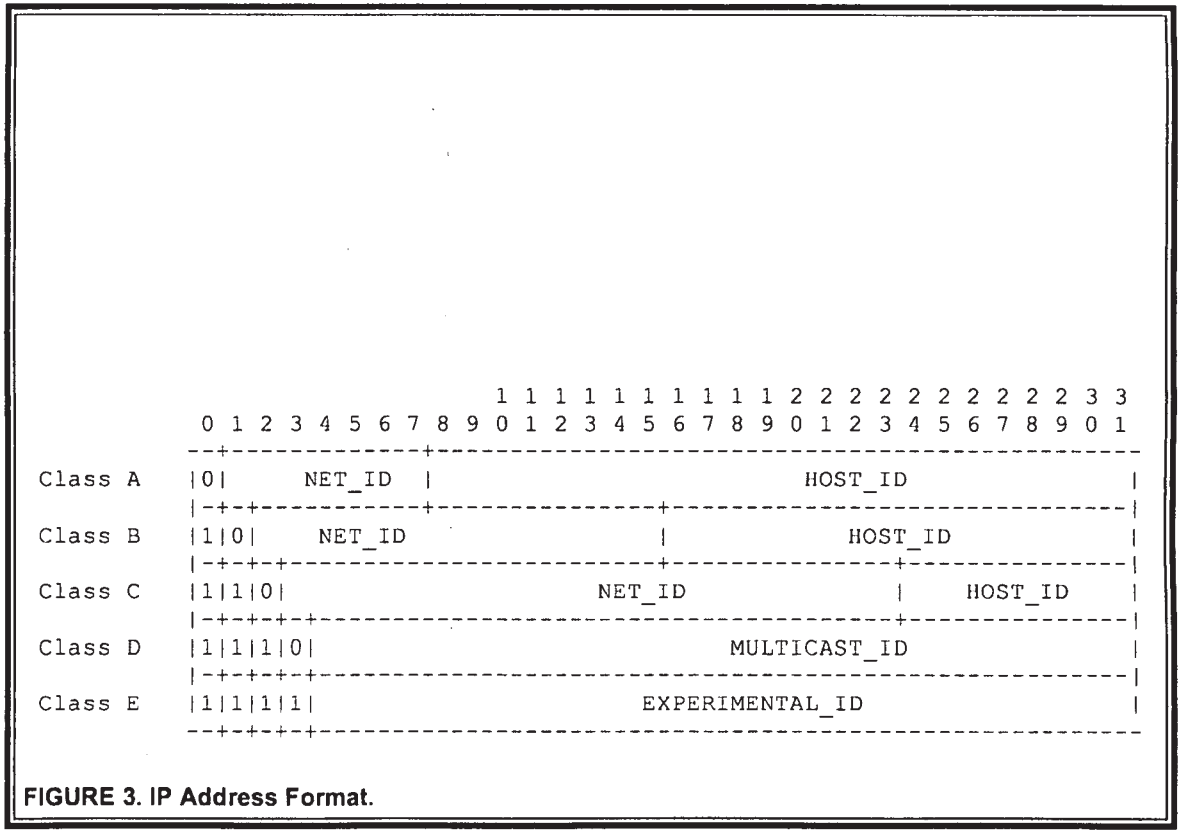


FIGURE 3. IP Address Format.

IP addresses are hierarchical for routing purposes and are subdivided into two subfields. The Network Identifier (NET_ID subfield identifies the TCP/IP subnetwork connected to the Internet. The NET_ID is used for high-level routing between networks, much the same way as the country code, city code, or area code is used in the telephone network. The Host Identifier (HOST_ID) subfield indicates the specific host within a subnetwork.

To accommodate different size networks, IP defines several *address classes*. Classes A, B, and C are used for host addressing and the only difference between the classes is the length of the NET_ID subfield:

- A Class A address has a 7-bit NET_ID and 24-bit HOST_ID. Class A addresses are intended for very large networks and can address up to 16,777,216 (2^{24}) hosts per network. The first digit of a Class A addresses will be number between 1 and 126. Relatively few Class A addresses have been assigned; examples include 4.0.0.0 (BBN Planet) and 9.0.0.0 (IBM).
- A Class B address has a 14-bit NET_ID and 16-bit HOST_ID. Class B addresses are intended for moderate sized networks and can address up to 65,536 (2^{16}) hosts per network. The first digit of a Class B address will be a number between 128 and 191. The Class B address space has long been threatened with being used up and it is has been very difficult to get a new Class B address for some time. Class B address assignment examples includ 128.138.0.0 (WestNet) and 152.163.0.0 (America Online).
- A Class C address has a 21-bit NET_ID and 8-bit HOST_ID. These addresses are intended for small networks and can address only up to 254 (2^8-2) hosts per network. The first digit of a Class C address will be a number between 192 and 223. Most addresses assigned to networks today are Class C (or sub-Class C!); examples include 208.162.102.0 (Hill Associates) and 209.198.87.0 (SoverNet, Bellows Falls, VT).

The remaining two address classes are used for special functions only and are not commonly assigned to individual hosts. Class D addresses may begin with a value between 224 and 239, and are used for IP multicasting (i.e., sending a single datagram to multiple hosts); the IANA maintains a list of Internet Multicast Addrsses. Class E addresses begin wit a value between 240 and 255, and are reserved for experimental use.

Several address values are reserved and/or have special meaning. A HOST_ID of 0 (as used above) is a dummy value reserved as a place holder when referring to an entire subnetwork; the address 208.162.106.0, then, refers to the Class C address with a NET_ID of 208.162.106. A HOST_ID of all ones (usually written "255" when referring to an all-ones byte, but also denoted as "-1") is a broadcast address and refers to all hosts on a network. A NET_ID value of 127 is used for loopback testing and the specific host address 127.0.0.1 refers to the *localhost*.

Several NET_IDs have been reserved in RFC 1918 for private network addresses and packets will not be routed over th Internet to these networks. Reserved NET_IDs are the Class A address 10.0.0.0 (formerly assigned to ARPANET), the sixteen Class B addresses 172.16.0.0-172.31.0.0, and the 256 Class C addresses 192.168.0.0-192.168.255.0.

An additional addressing tool is the *subnet mask*. Subnet masks are used to indicate the portion of the address that identifies the network (and/or subnetwork) for routing purposes. The subnet mask is written in dotted decimal and the number of 1s indicates the significant NET_ID bits. For "classful" IP addresses, the subnet mask and number of significant address bits for the NET_ID are:

Class Subnet Mask Number of Bits

A	255.0.0.0	8
B	255.255.0.0	16
C	255.255.255.0	24

Depending upon the context and literature, subnet masks may be written in dotted decimal form or just as a number representing the number of significant address bits for the NET_ID. Thus, `208.162.106.17 255.255.255.0` and `208.162.106.17/24` both refer to a Class C NET_ID of 208.162.106.

Subnet masks can also be used to subdivide a large address space or to combine multiple small address spaces. For example, a network may subdivide their address space to define multiple logical networks by segmenting the HOST_ID subfield into a Subnetwork Identifier (SUBNET_ID) and (smaller) HOST_ID. For example, a user might be assigned the Class B address space 172.16.0.0 which might be segmented into a 16-bit NET_ID, 4-bit SUBNET_ID, and 12-bit HOST_ID. In this case, the subnet mask for routing to the NET_ID on the Internet would be 255.255.0.0 (or "/16"), while the mask for routing to individual subnets within the larger Class B address space would be 255.255.240.0 (or "/20").

Alternatively, a single user might be assigned the four Class C addresses 192.168.128.0, 192.168.129.0, 192.168.130.0 and 192.168.131.0, and use the subnet mask 255.255.252.0 (or "/22") for routing to this domain. This use of subnet masks in routing tables to consolidate addresses uses a process called *Classless Interdomain Routing (CIDR)*, describe in RFCs 1518 and 1519. It should be obvious from this example that CIDR address consolidation results in smaller route tables; in the example here, routing information for four Class C addresses can be specified in a single router table entry

As of January 1996, there were 95 Class A addresses, 5892 Class B addresses, and 128,378 Class C addresses assigned; this number is undoubtedly larger today, particularly in the Class C space. Because CIDR is becoming so widely used, however, these numbers are not a true reflection of the number of networks attached to the public Internet because multiple addresses may be assigned to a single organizational entity.

3.2.2. The Domain Name System

While IP addresses are 32 bits in length, most users do not memorize the numeric addresses of the hosts to which they attach; instead, people are more comfortable with host names. Most IP hosts, then, have both a numeric IP address and a name. While this is convenient for people, however, the name must be translated back to a numeric address for routing purposes.

Earlier discussion in this paper described the domain naming structure of the Internet. In the early ARPANET, every host maintained a file called `HOSTS.TXT` that contained a list of all hosts, which included the IP address, host name, and alias (es). This was an adequate measure while the ARPANET was small and had a slow rate of growth, but was not a scalable solution as the network grew.

[NOTE: `HOSTS.TXT` files are still found on Unix systems although usually used to reconcile names of hosts on the local network to cut down on local DNS traffic. On Microsoft Windows systems, the file is called `HOSTS` and can typically be found in the `c:\windows` folder.]

To handle the fast rate of new names on the network, the Domain Name System (DNS) was created. The DNS is a distributed database containing host name and IP address information for all domains on the Internet. There is a single *authoritative name server* for every domain that contains all DNS-related information about the domain; each domain als has at least one secondary name server that also contains a copy of this information. Thirteen *root servers* around the globe (most in the U.S., actually, with the remainder in Asia and Europe) maintain a list of all of these authoritative name servers.

When a host on the Internet needs to obtain a host's IP address based upon the host's name, a DNS request is made by the initial host to the to a local name server. The local name server may be able to respond to the request with information that is either configured or cached at the name server; if necessary information is not available, the local name server forwards the request to one of the root servers. The root server, then, will determine an appropriate name server for the target host and the DNS request will be forwarded to the domain's name server.

Name servers contain the following types of information:

- *A-record:* An address record maps a hostname to an IP address.
- *PTR-record:* A pointer record maps an IP address to a hostname.
- *NS-record:* A name server record lists the authoritative name server(s) for a given domain.
- *MX-record:* A mail exchange record lists the mail servers for a given domain. As an example, consider the author' e-mail address, *kumquat@hill.com*. Note that the "hill.com" portion of the address is a domain name, not a host name, and mail has to be sent to a specific host. The MX-records in the *hill.com* name database specifies the hos *mail.hill.com* is the mail server for this domain.

More information about the DNS can be found from the World Internetworking Alliance (WIA) Web site. Additional DNS references include *DNS and BIND* by P. Albitz and C. Liu (O'Reilly & Associates) and "Setting up Your own DNS" by G. Kessler. The concepts, structure, and delegation of the DNS are described in RFCs 1034 and 1591. In addition, the IANA maintains a list of DNS parameters.

3.2.3. ARP and Address Resolution

Early IP implementations ran on hosts commonly interconnected by Ethernet local area networks (LAN). Every transmission on the LAN contains the local network, or medium access control (MAC), address of the source and destination nodes. MAC addresses are 48-bits in length and are non-hierarchical, so routing cannot be performed using the MAC address. MAC addresses are never the same as IP addresses.

When a host needs to send a datagram to another host on the same network, the sending application must know both the IP and MAC addresses of the intended receiver; this is because the destination IP address is placed in the IP packet and the destination MAC address is placed in the LAN MAC protocol frame. (If the destination host is on another network the sender will look instead for the MAC address of the default gateway, or router.)

Unfortunately, the sender's IP process may not know the MAC address of the intended receiver on the same network. The Address Resolution Protocol (ARP), described in RFC 826, provides a mechanism so that a host can learn a receiver's MAC address when knowing only the IP address. The process is actually relatively simple: the host sends an ARP Request packet in a frame containing the MAC broadcast address; the ARP request advertises the destination IP address and asks for the associated MAC address. The station on the LAN that recognizes its own IP address will send an ARP Response with its own MAC address. As Figure 1 shows, ARP message are carried directly in the LAN frame and ARP is an independent protocol from IP. The IANA maintains a list of all ARP parameters.

Other address resolution procedures have also been defined, including:

- Reverse ARP (RARP), which allows a disk-less processor to determine its IP address based on knowing its own MAC address
- Inverse ARP (InARP), which provides a mapping between an IP address and a frame relay virtual circuit identifier
- ATMARP and ATMInARP provide a mapping between an IP address and ATM virtual path/channel identifiers.
- LAN Emulation ARP (LEARP), which maps a recipient's ATM address to its LAN Emulation (LE) address (which takes the form of an IEEE 802 MAC address).

[NOTE: IP hosts maintain a cache storing recent ARP information. The ARP cache can be viewed from a Unix or DOS (i Windows 95/98/NT) command line using the `arp -a` command.]

3.2.4. IP Routing: OSPF, RIP, and BGP

As an OSI Network Layer protocol, IP has the responsibility to route packets. It performs this function by looking up a

packet's destination IP NET_ID in a routing table and forwarding based on the information in the table. But it is *routing protocols*, and *not* IP, that populate the routing tables with routing information. There are three routing protocols commonly associated with IP and the Internet, namely, RIP, OSPF, and BGP.

OSPF and RIP are primarily used to provide routing within a particular domain, such as within a corporate network or within an ISP's network. Since the routing is *inside* of the domain, these protocols are generically referred to as *interior gateways protocols*.

The Routing Information Protocol version 2 (RIP-2), described in RFC 2453, describes how routers will exchange routing table information using a distance-vector algorithm. With RIP, neighboring routers periodically exchange their entire routing tables. RIP uses hop count as the metric of a path's cost, and a path is limited to 16 hops. Unfortunately, RIP has become increasingly inefficient on the Internet as the network continues its fast rate of growth. Current routing protocols for many of today's LANs are based upon RIP, including those associated with NetWare, AppleTalk, VINES, and DECne The IANA maintains a list of RIP message types.

The Open Shortest Path First (OSPF) protocol is a link state routing algorithm that is more robust than RIP, converges faster, requires less network bandwidth, and is better able to scale to larger networks. With OSPF, a router broadcasts only changes in its links' status rather than entire routing tables. OSPF Version 2, described in RFC 1583, is rapidly replacing RIP in the Internet.

The Border Gateway Protocol version 4 (BGP-4) is an *exterior gateway protocol* because it is used to provide routing information between Internet routing domains. BGP is a distance vector protocol, like RIP, but unlike almost all other distance vector protocols, BGP tables store the actual route to the destination network. BGP-4 also supports policy-base routing, which allows a network's administrator to create routing policies based on political, security, legal, or economic issues rather than technical ones. BGP-4 also supports CIDR. BGP-4 is described in RFC 1771, while RFC 1268 describes use of BGP in the Internet. In addition, the IANA maintains a list of BGP parameters.

Figure 1 shows the protocol relationship of RIP, OSPF, and BGP to IP. A RIP message is carried in a UDP datagram which, in turn, is carried in an IP packet. An OSPF message, on the other hand, is carried directly in an IP datagram. BGP messages, in a total departure, are carried in TCP segments over IP. Although all of the TCP/IP books mentioned above discuss IP routing to some level of detail, *Routing in the Internet* by Christian Huitema is one of the best available references on this specific subject.

3.2.5. ICMP

The Internet Control Message Protocol, described in RFC 792, is an adjunct to IP that notifies the sender of IP datagram about abnormal events. This collateral protocol is particularly important in the connectionless environment of IP.

The commonly employed ICMP message types include:

- *Destination Unreachable:* Indicates that a packet cannot be delivered because the destination host cannot be reached. The reason for the non-delivery may be that the host or network is unreachable or unknown, the protoco or port is unknown or unusable, fragmentation is required but not allowed (DF-flag is set), or the network or host is unreachable for this type of service.
- *Echo* and *Echo Reply:* These two messages are used to check whether hosts are reachable on the network. One host sends an Echo message to the other, optionally containing some data, and the receiving host responds with an Echo Reply containing the same data. These messages are the basis for the Ping command.
- *Parameter Problem:* Indicates that a router or host encountered a problem with some aspect of the packet's

Header.
- *Redirect:* Used by a host or router to let the sending host know that packets should be forwarded to another address. *For security reasons, Redirect messages should usually be blocked at the firewall.*
- *Source Quench:* Sent by a router to indicate that it is experiencing congestion (usually due to limited buffer space and is discarding datagrams.
- *TTL Exceeded:* Indicates that a datagram has been discarded because the TTL field reached 0 or because the entire packet was not received before the fragmentation timer expired.
- *Timestamp* and *Timestamp Reply:* These messages are similar to the Echo messages, but place a timestamp (with millisecond granularity) in the message, yielding a measure of how long remote systems spend buffering and processing datagrams, and providing a mechanism so that hosts can synchronize their clocks.

ICMP messages are carried in IP packets. The IANA maintains a complete list of ICMP parameters.

3.2.6. IP version 6

The official version of IP that has been in use since the early 1980s is *version 4*. Due to the tremendous growth of the Internet and new emerging applications, it was recognized that a new version of IP was becoming necessary. In late 1995, IP version 6 (IPv6) was entered into the Internet Standards Track. The primary description of IPv6 is contained in RFC 1883 and a number of related specifications, including ICMPv6.

IPv6 is designed as an evolution from IPv4, rather than a radical change. Primary areas of change relate to:

- Increasing the IP address size to 128 bits
- Better support for traffic types with different quality-of-service objectives
- Extensions to support authentication, data integrity, and data confidentiality

For more information about IPv6, check out:

- *IPng: Internet Protocol Next Generation* by Scott Bradner and Allison Mankin (Addison-Wesley, 1996)
- *IPv6: The New Internet Protocol* by Christian Huitema (Prentice-Hall, 1996).
- "IPv6: The Next Generation Internet Protocol" by Gary Kessler.
- *IPng and the TCP/IP Protocols* by Stephen Thomas (John Wiley & Sons, 1996)
- IPng Working Group page (IETF)
- IP Next Generation Web Page (Sun)
- 6bone Web Page (LBL)

3.3. The Transport Layer Protocols

The TCP/IP protocol suite comprises two protocols that correspond roughly to the OSI Transport and Session Layers; these protocols are called the Transmission Control Protocol and the User Datagram Protocol (UDP). One can argue tha it is a misnomer to refer to "TCP/IP applications," as most such applications actually run over TCP or UDP, as shown in Figure 1.

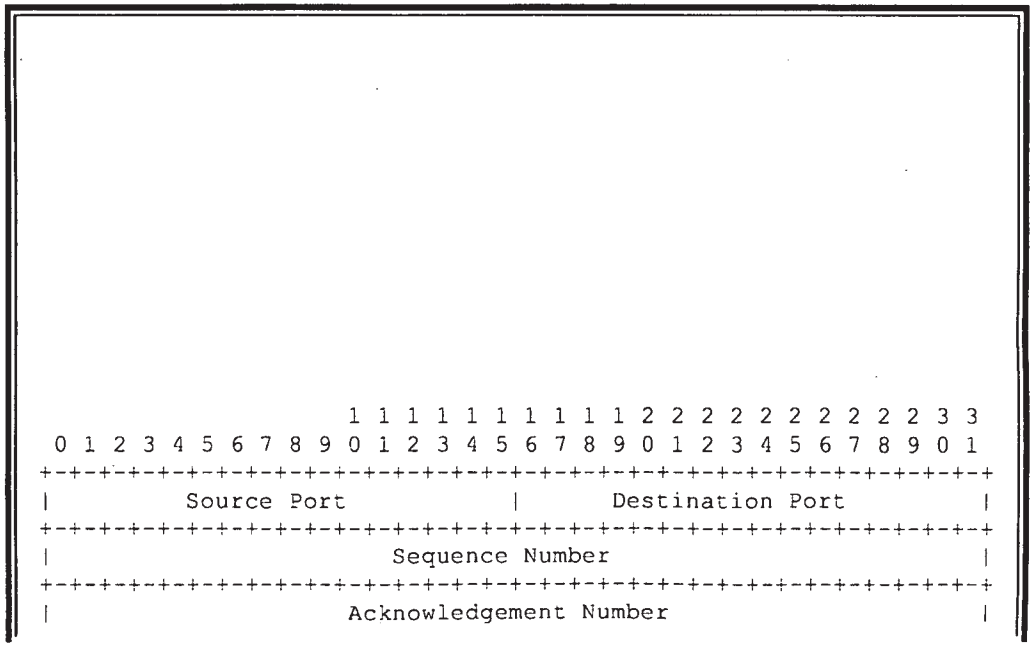
Higher-layer applications are referred to by a port identifier in TCP/UDP messages. The port identifier and IP address together form a *socket*, and the end-to-end communication between two hosts is uniquely identified on the Internet by th four-tuple (source port, source address, destination port, destination address). *Well-known port numbers* denote the server side of a connection and include:

Port #	Protocol	Application
20	TCP	FTP data transfer
21	TCP	FTP control
23	TCP	Telnet
25	TCP	SMTP
43	TCP	whois
53	TCP/UDP	DNS
70	TCP	Gopher
79	TCP	finger
80	TCP	HTTP
110	TCP	POPv3
161	UDP	SNMP
162	UDP	SNMP-trap
520	UDP	RIP

A complete list of port numbers that have been assigned can be found in the IANA's list of Port Numbers.

3.3.1. TCP

TCP, described in RFC 793, provides a virtual circuit (connection-oriented) communication service across the network. TCP includes rules for formatting messages, establishing and terminating virtual circuits, sequencing, flow control, and error correction. Most of the applications in the TCP/IP suite operate over the *reliable* transport service provided by TCP.



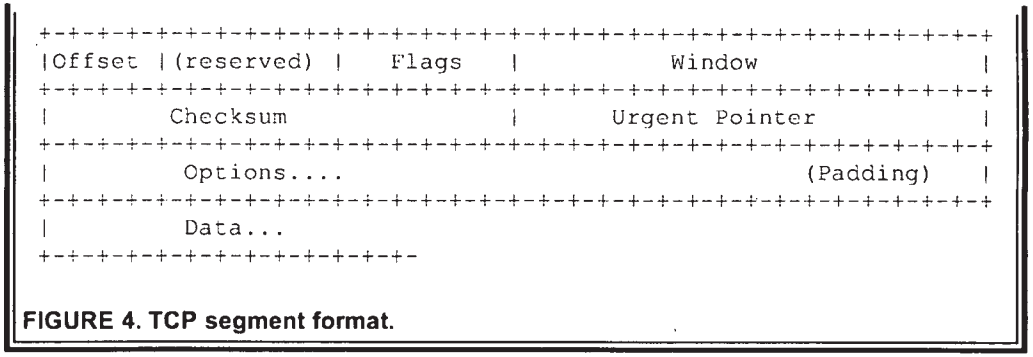


FIGURE 4. TCP segment format.

The TCP data unit is called a *segment*; the name is due to the fact that TCP does not recognize messages, per se, but merely sends a block of bytes from the byte stream between sender and receiver. The fields of the segment (Figure 4) are:

- *Source Port* and *Destination Port:* Identify the source and destination ports to identify the end-to-end connection and higher-layer application.
- *Sequence Number:* Contains the sequence number of this segment's first data byte in the overall connection byte stream; since the sequence number refers to a byte count rather than a segment count, sequence numbers in contiguous TCP segments are not numbered sequentially.
- *Acknowledgment Number:* Used by the sender to acknowledge receipt of data; this field indicates the sequence number of the next byte expected from the receiver.
- *Data Offset:* Points to the first data byte in this segment; this field, then, indicates the segment header length.
- *Control Flags:* A set of flags that control certain aspects of the TCP virtual connection. The flags include:
  - *Urgent Pointer Field Significant (URG):* When set, indicates that the current segment contains urgent (or high-priority) data and that the Urgent Pointer field value is valid.
  - *Acknowledgment Field Significant (ACK):* When set, indicates that the value contained in the Acknowledgment Number field is valid. This bit is usually set, except during the first message during connection establishment.
  - *Push Function (PSH):* Used when the transmitting application wants to force TCP to immediately transmit the data that is currently buffered without waiting for the buffer to fill; useful for transmitting small units of data.
  - *Reset Connection (RST):* When set, immediately terminates the end-to-end TCP connection.
  - *Synchronize Sequence Numbers (SYN):* Set in the initial segments used to establish a connection, indicating that the segments carry the initial sequence number.
  - *Finish (FIN):* Set to request normal termination of the TCP connection in the direction this segment is traveling; completely closing the connection requires one FIN segment in each direction.
- *Window:* Used for flow control, contains the value of the *receive window size* which is the number of transmitted bytes that the sender of this segment is willing to accept from the receiver.
- *Checksum:* Provides rudimentary bit error detection for the segment (including the header and data).
- *Urgent Pointer.* Urgent data is information that has been marked as high-priority by a higher layer application; this data, in turn, usually bypasses normal TCP buffering and is placed in a segment between the header and "norma data. The Urgent Pointer, valid when the URG flag is set, indicates the position of the first octet of nonexpedited data in the segment.
- *Options:* Used at connection establishment to negotiate a variety of options; maximum segment size (MSS) is the most commonly used option and, if absent, defaults to an MSS of 536. The IANA maintains a list of all TCP Option Numbers.

3.3.2. UDP

UDP, described in RFC 768, provides an end-to-end datagram (connectionless) service. Some applications, such as those that involve a simple query and response, are better suited to the datagram service of UDP because there is no time lost to virtual circuit establishment and termination. UDP's primary function is to add a port number to the IP address to provide a socket for the application.

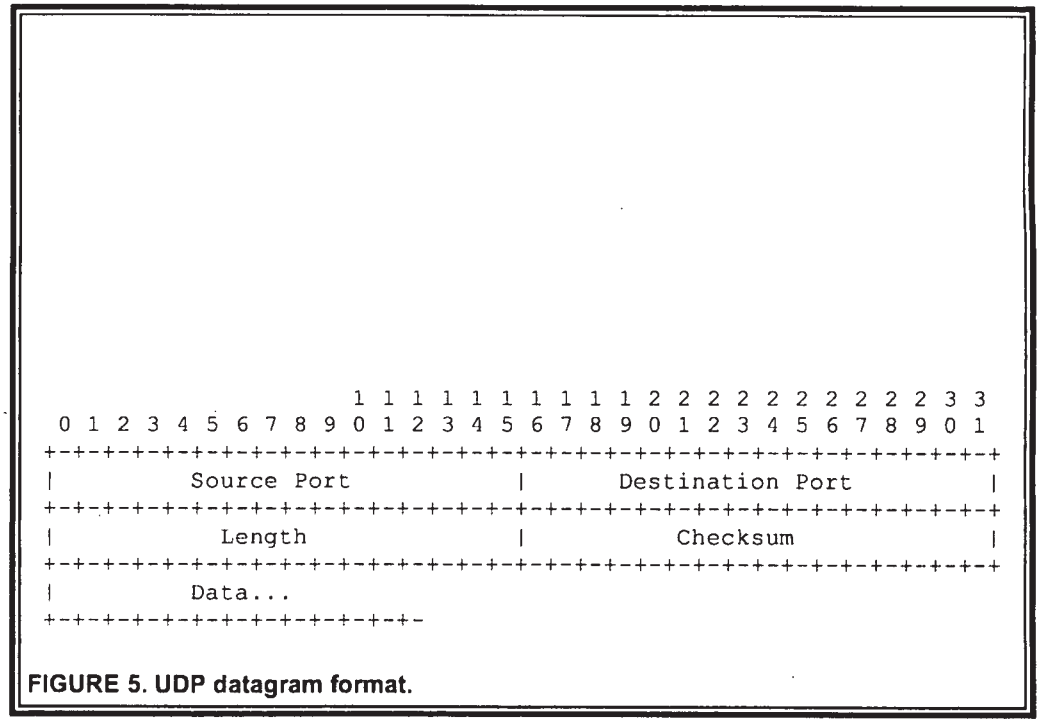


FIGURE 5. UDP datagram format.

The fields of a UDP datagram (Figure 5) are:

- *Source Port:* Identifies the UDP port at the source side of the connection; use of this field is optional in UDP and may be set to 0.
- *Destination Port:* Identifies the destination port of the end-to-end connection.
- *Length:* Indicates the total length of the UDP datagram.
- *Checksum:* Provides rudimentary bit error detection for the datagram (including the header and data).

3.4. Applications

The TCP/IP Application Layer protocols support the applications and utilities that are the Internet. Commonly used protocols include:

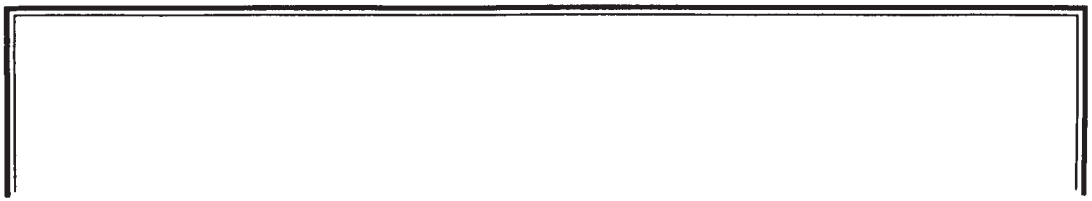
- *Telnet:* Short for *Telecommunication Network*, a virtual terminal protocol allowing a user logged on to one TCP/IP host to access other hosts on the network (RFC 854).

- *FTP:* The File Transfer Protocol allows a user to transfer files between local and remote host computers (RFC 959).
- *Archie:* A utility that allows a user to search all registered anonymous FTP sites for files on a specified topic.
- *Gopher:* A tool that allows users to search through data repositories using a menu-driven, hierarchical interface, with links to other sites (RFC 1436).
- *SMTP:* The Simple Mail Transfer Protocol is the standard protocol for the exchange of electronic mail over the Internet (RFC 821). SMTP is used between e-mail servers on the Internet or to allow an e-mail client to send mail to a server. RFC 822 specifically describes the mail message body format, and RFCs 1521 and 1522 describe MIME (Multipurpose Internet Mail Extensions). Reference books on electronic mail systems include *!%@:: Addressing and Networks* by D. Frey and R. Adams (O'Reilly & Associates, 1993) and *THE INTERNET MESSAGE: Closing the Book With Electronic Mail* by M. Rose (PTR Prentice Hall, 1993).
- *HTTP:* The Hypertext Transfer Protocol is the basis for exchange of information over the World Wide Web (WWW). Various versions of HTTP are in use over the Internet, with HTTP version 1.0 (RFC 1945) being the mos current. WWW pages are written in the Hypertext Markup Language (HTML), an ASCII-based, platform-independent formatting language (RFC 1866).
- *Finger:* Used to determine the status of other hosts and/or users (RFC 1288).
- *POP:* The Post Office Protocol defines a simple interface between a user's mail client software and an e-mail server; POP is used to download mail from the server to the client and allows the user to manage their mailboxes The current version is POP3 (RFC 1460).
- *DNS:* The Domain Name System (described in slightly more detail in Section 3.2.2 above) defines the structure of Internet names and their association with IP addresses, as well as the association of mail and name servers with domains.
- *SNMP:* The Simple Network Management Protocol defines procedures and management information databases for managing TCP/IP-based network devices. SNMP (RFC 1157) is widely deployed in local and wide area network. SNMP Version 2 (SNMPv2, RFC 1441) adds security mechanisms that are missing in SNMP, but is also very complex; widespread use of SNMPv2 has yet to be seen. Additional information on SNMP and TCP/IP-based network management can be found in *SNMP* by S. Feit (McGraw-Hill, 1994) and *THE SIMPLE BOOK: An Introduction to Internet Management*, 2/e, by M. Rose (PTR Prentice Hall, 1994).
- *Ping:* The Packet Internet Groper, a utility that allows a user at one system to determine the status of other hosts and the latency in getting a message to that host. Uses ICMP Echo messages.
- *Whois/NICNAME:* Utilities that search databases for information about Internet domains and domain contact information (RFC 954).
- *Traceroute:* A tool that displays the route that packets will take when traveling to a remote host.

A guide to using most of these applications can be found in "A Primer on Internet and TCP/IP Tools and Utilities" (FYI 30/RFC 2151) by Gary Kessler & Steve Shepard (also available in HTML, Postscript, and Word).

3.5. Summary

As this discussion has shown, *TCP/IP* is not merely a pair of communication protocols but is a suite of protocols, applications, and utilities. Increasingly, these protocols are referred to as the *Internet Protocol Suite*, but the older name will not disappear anytime soon.



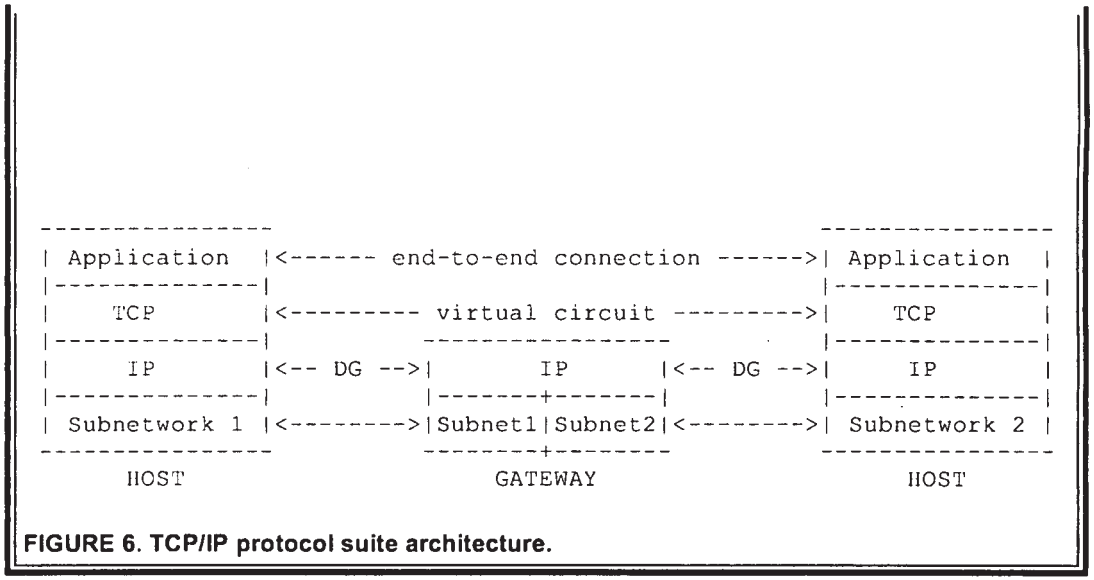


FIGURE 6. TCP/IP protocol suite architecture.

Figure 6 shows the relationship between the various protocol layers of TCP/IP. Applications and utilities reside in host, o end-communicating, systems. TCP provides a reliable, virtual circuit connection between the two hosts. (UDP, not shown, provides an end-to-end datagram connection at this layer.) IP provides a datagram (DG) transport service over any intervening subnetworks, including local and wide area networks. The underlying subnetwork may employ nearly any common local or wide area network technology.

Note that the term *gateway* is used for the device interconnecting the two subnets, a device usually called a *router* in LAN environments or *intermediate system* in OSI environments. In OSI terminology, a *gateway* is used to provide protocol conversion between two networks and/or applications.

4. Other Information Sources

This memo has only provided background information about the TCP/IP protocols and the Internet. There is a wide rang of additional information that the reader can access to further use and understand the tools and scope of the Internet. The real fun begins now!

Internet specifications, standards, reports, humor, and tutorials are distributed as Request for Comments (RFC) documents. RFCs are all freely available on-line, and most are available in ASCII text format.

Internet standards are documented in a subset of the RFCs, identified with an "STD" designation. RFC 2026 describes the Internet standards process and STD 1 always contains the official list of Internet standards.

For Your Information (FYI) documents are another RFC subset, specifically providing background information for the Internet community. The FYI notes are described in RFC 1150.

Frequently Asked Question (FAQ) lists may be found for a number of topics, ranging from ISDN and cryptography to the Internet and Gopher. Two such FAQs are of particular interest to Internet users: "FYI on Questions and Answers - Answers to Commonly asked 'New Internet User' Questions" (RFC 1594) and "FYI on Questions and Answers: Answers to Commonly Asked 'Experienced Internet User' Questions" (RFC 1207). All three of these documents point to even mor information sources.

5. Acronyms and Abbreviations

ARP	Address Resolution Protocol
ARPANET	Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
BSD	Berkeley Software Development
CCITT	International Telegraph and Telephone Consultative Committee
CIX	Commercial Internet Exchange
DARPA	Defense Advanced Research Projects Agency
DNS	Domain Name System
DoD	U.S. Department of Defense
FAQ	Frequently Asked Questions lists
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
FYI	For Your Information series of RFCs
GOSIP	U.S. Government Open Systems Interconnection Profile
HTML	Hypertext Markup Language
HTTP	Hypetext Transfer Protocol
IAB	Internet Activities Board
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
ISOC	Internet Society
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
MAC	Medium (or media) access control
Mbps	Megabits (millions of bits) per second
NICNAME	Network Information Center name service
NSF	National Science Foundation
NSFNET	National Science Foundation Network

OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PPP	Point-to-Point Protocol
RARP	Reverse Address Resolution Protocol
RIP	Routing Information Protocol
RFC	Request For Comments
SLIP	Serial Line IP
SMDS	Switched Multimegabit Data Service
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
STD	Internet Standards series of RFCs
TCP	Transmission Control Protocol
TLD	Top-level domain
UDP	User Datagram Protocol

6. Author's Address

Gary C. Kessler
Hill Associates
106 Highpoint Center
Colchester, VT 05446

+1 802-879-3375 (home office)
+1 630-604-5529 (fax)

E-mail: kumquat@hill.com or kumquat@sover.net
http://www.hill.com or http://www.sover.net/~kessfam

The Common Object Request Broker: Architecture and Specification

Revision 2.6
December 2001

PATENT

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

NOTICE

The information contained in this document is subject to change without notice. The material in this document details an Object Management Group specification in accordance with the license and notices set forth on this page. This document does not represent a commitment to implement any portion of this specification in any company's products.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR PARTICULAR PURPOSE OR USE. In no event shall The Object Management Group or any of the companies listed above be liable for errors contained herein or for indirect, incidental, special, consequential, reliance or cover damages, including loss of profits, revenue, data or use, incurred by any user or any third party. The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Right in Technical Data and Computer Software Clause at DFARS 252.227.7013 OMG® and Object Management are registered trademarks of the Object Management Group, Inc. Object Request Broker, OMG IDL, ORB, CORBA, CORBAfacilities, CORBAservices, COSS, and IIOP are trademarks of the Object Management Group, Inc. X/Open is a trademark of X/Open Company Ltd.

ISSUE REPORTING

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents & Specifications, Report a Bug/Issue.

Contents

Contents

Contents