ASN.1 & BER

ASN.1

Purpose of ASN.1

One of the fundamental problems confronting users communicating with different systems is the efficient transfer of data in such a way that the data received is the same data transmitted. In the OSI model, the representation of data types and structures to facilitate this transfer is a function of the Application Layer; the encoding of the data into a specific sequence of bits for transfer isattributed to the Presentation Layer.

ASN.1

The fundamental unit of ASN.1 is the module. The sole purpose of a module is to name a collection of type definitions and/or value definitions (assignments) that constitute a data specification.

ASN.1

A **type definition** is used to define and name a new type by means of a **type assignment** and a **value definition** is used to define and name a specific value, when it is necessary, by means of a **value assignment**.

ASN.1 - Type Assignment

A **type assignment** consists of a type **reference** (the name of the type), the character sequence **::=** (“is defined as”), and the appropriate type.

ASN.1 – Type Assignment example

```
InventoryList {1 2 0 0 6 1} DEFINITIONS ::=
        BEGIN
                {
                        ItemId ::= SEQUENCE
                                {
                                partnumber IA5String,
                                quantity INTEGER,
                                wholesaleprice REAL,
                                saleprice REAL
                                }
                        StoreLocation ::= ENUMERATED
                                {
                                Baltimore (0),
                                Philadelphia (1),
                                Washington (2)
                                }
                }
        END
```

ASN.1 - Value assignment

A **value assignment** consists of a value **reference** (the name of the value), the type of the value, **::=** (``is assigned the value''), and a valid value notation. A value reference must begin with a lower case letter, but otherwise has the same syntax as a type assignment.

ASN.1 - Value assignment example

```
{
        ItemId ::= SEQUENCE
         {                                                    partnumber
IA5String,
                quantity INTEGER,
                wholesaleprice REAL,                    saleprice REAL
        }
```

```
gadget ItemId ::=
        {
                partnumber "7685B2",
                quantity 73,
                wholesaleprice 13.50,
                saleprice 24.95
        }
```

ASN.1 – Simple Types

Simple Types	Tag	Typical Use
BOOLEAN	1	Model logical, two-state variable values
INTEGER	2	Model integer variable values
BIT STRING	3	Model binary data of arbitrary length
OCTET STRING	4	Model binary data whose length is a multiple of eight
NULL	5	Indicate effective absence of a sequence element
OBJECT IDENTIFIER	6	Name information objects
REAL	9	Model real variable values
ENUMERATED	10	Model values of variables with at least three states
CHARACTER STRING	*	Models values that are strings of characters from a specified characterset

ASN.1 – Character String Type

Character String Type	Tag	Character Set
`NumericString`	18	0,1,2,3,4,5,6,7,8,9, and space
`PrintableString`	19	Upper and lower case letters, digits, space, apostrophe, left/right parenthesis, plus sign, comma, hyphen, full stop, solidus, colon, equal sign, question mark
`TeletexString (T61String)`	20	The Teletex character set in CCITT's T61, space, and delete
`VideotexString`	21	The Videotex character set in CCITT's T.100 and T.101, space, and delete
`VisibleString (ISO646String)`	26	Printing character sets of international ASCII, and space
`IA5String`	22	International Alphabet 5 (International ASCII)
`GraphicString`	25	All registered G sets, and space
`GraphicString`	27	All registered C and G sets, space and delete

ASN.1 – Structured types

Structured Types	Tag	Typical Use
SEQUENCE	16	Models an ordered collection of variables of different type
SEQUENCE OF	16	Models an ordered collection of variables of the same type
SET	17	Model an unordered collection of variables of different types
SET OF	17	Model an unordered collection of variables of the same type
CHOICE	*	Specify a collection of distinct types from which to choose one type
SELECTION	*	Select a component type from a specified CHOICE type
ANY	*	Enable an application to specify the type **Note:** ANY is a deprecated ASN.1 Structured Type. It has been replaced with X.680 Open Type.

ASN.1 – Structured types

example

```
{
      "American", "1106", { 320, 107, 213 }, { "BWI", "LAX" }, 10
}

AirlineFlight ::= SEQUENCE
      {
      airline IA5String,
      flight NumericString,
      seats SEQUENCE
        {
        maximum INTEGER,
        occupied INTEGER,
        vacant INTEGER
        },
      airport SEQUENCE
        {
        origin IA5String,
        stop1 [0] IA5String OPTIONAL,
        stop2 [1] IA5String OPTIONAL,
        destination IA5String
        },
      crewsize ENUMERATED
        {
        six (6),
        eight (8),
        ten (10)
        },
      cancel BOOLEAN DEFAULT FALSE
      }
```

ASN.1 – Tagged

Type **TAGGED** is used to enable the receiving system to correctly decode values from several datatypes that a protocol determines may be transmitted at any given time. **TAGGED** has no value notation of its own. Its type notation consists of three elements: a user-defined tag, possibly followed by **IMPLICIT** or **EXPLICIT**, followed by the value notation of the type being tagged.

ASN.1 – Tagged

The user-defined tag consists of a class and class number contained in braces. Class is

UNIVERSAL, APPLICATION, PRIVATE, CONTEXT-SPECIFIC.

The **UNIVERSAL** class is restricted to the ASN.1 built-in types. It defines an application-independent data type that must be distinguishable from all other data types. The other three classes are **user defined**.

The **APPLICATION** class distinguishes data types that have a wide, scattered use within a particular presentation context.

PRIVATE distinguishes data types within a particular organization or country.

CONTEXT-SPECIFIC distinguishes members of a sequence or set, the alternatives of a CHOICE, or universally tagged set members.

Only the class number appears in braces for this data type; the term COONTEXT-SPECIFIC does not appear.

ASN.1 – Tagged

a)
```
seats  SET
  {
   maximum    INTEGER,
   occupied   INTEGER,
   vacant     INTEGER
  }
```

b)
```
seats  SET
  {
   maximum   [APPLICATION 0] INTEGER,
   occupied  [APPLICATION 1] INTEGER,
   vacant    [APPLICATION 2] INTEGER
  }
```

c)
```
seats  SET
  {
   maximum   [APPLICATION 0] IMPLICIT INTEGER,
   occupied  [APPLICATION 1] IMPLICIT INTEGER,
   vacant    [APPLICATION 2] IMPLICIT INTEGER
  }
```

d)
```
seats  SET
  {
   maximum   [0] INTEGER,
   occupied  [1] INTEGER,
   vacant    [2] INTEGER
  }
```

ASN.1 – Tagged

As we indicated in the above discussion of type SET, the representation in **a)** is invalid in ASN.1 because its instances can be ambiguous.

The tagging in representations **b), c), and d)** overcome the problem and allow instances to be transmitted uniquely.

IMPLICIT in **c)** indicates that an original tag is replaced by any of the three user-defined tags.

EXPLICIT tagging would be appropriate when strong-type-checking is more important than compact representation; it can be used when the original tag is accompanied by a user-defined tag.

The context-specific tagging in **d)** is similar to the **APPLICATION** class tagging in **b)** except that the class of the tag is not specifically transmitted.

ASN.1 – Listing of Universal Tags

Universal Tag Number	Description
0	reserved for BER
1	BOOLEAN
2	INTEGER
3	BIT STRING
4	OCTET STRING
5	NULL
6	OBJECT IDENTIFIER
7	ObjectDescriptor

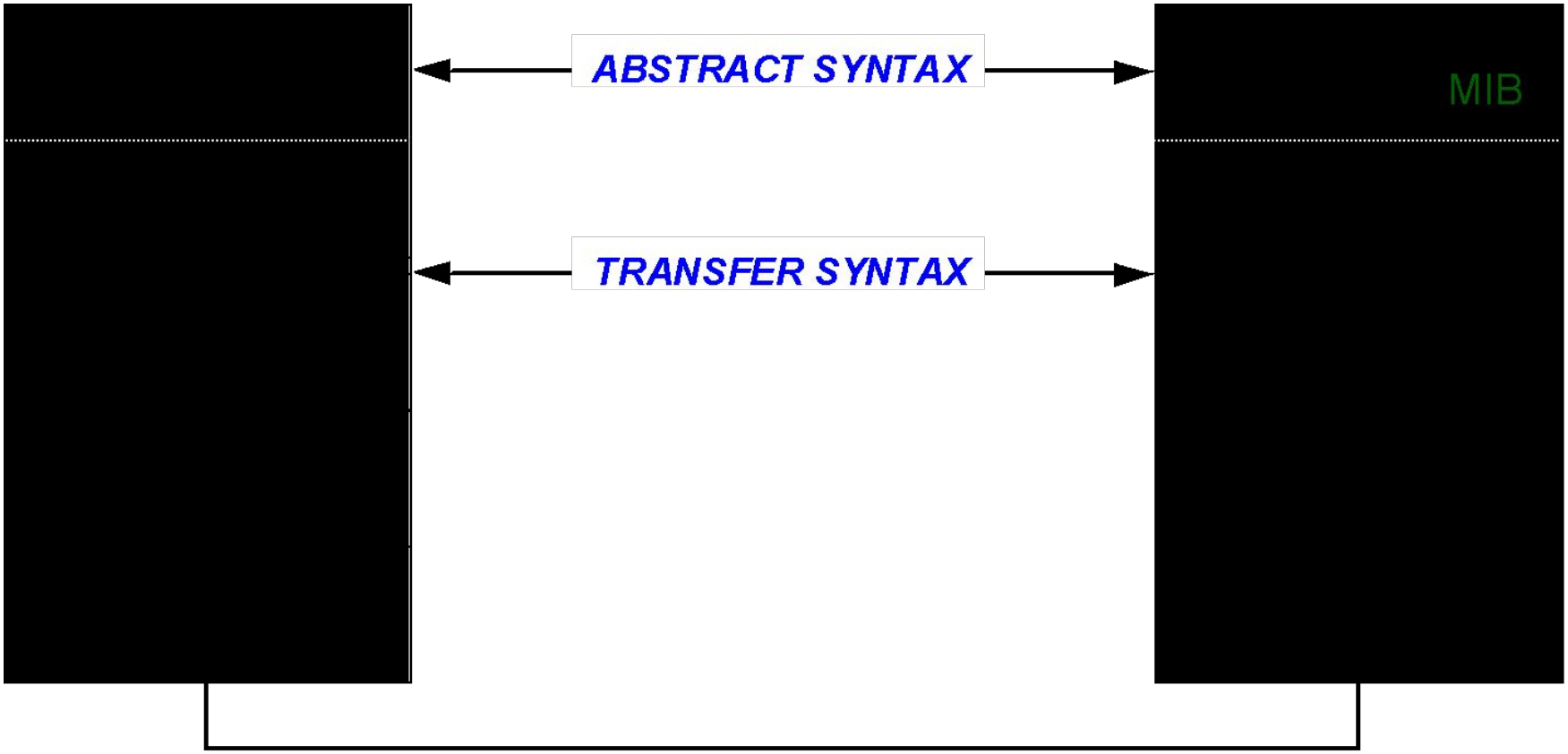
ASN.1 – Listing of Universal Tags

8	INSTANCE OF, EXTERNAL
9	REAL
10	ENUMERATED
11	EMBEDDED PDV
12	UTF8String
13	RELATIVE-OID
16	SEQUENCE, SEQUENCE OF
17	SET, SET OF
18	NumericString
19	PrintableString

ASN.1 – Listing of Universal Tags

20	TeletexString, T61String
21	VideotexString
22	IA5String
23	UTCTime
24	GeneralizedTime
25	GraphicString
26	VisibleString, ISO646String
27	GeneralString
28	UniversalString
29	CHARACTER STRING
30	BMPString

Abstract Syntax und Transfer Syntax



Задача

Дефиниране на типове данни за следния обект:

LAN – мрежова карта:

- Индекс : 0
- Производител : 3Com
- Получени пакети: 521
- Изпратени пакети: 130

Type assignments

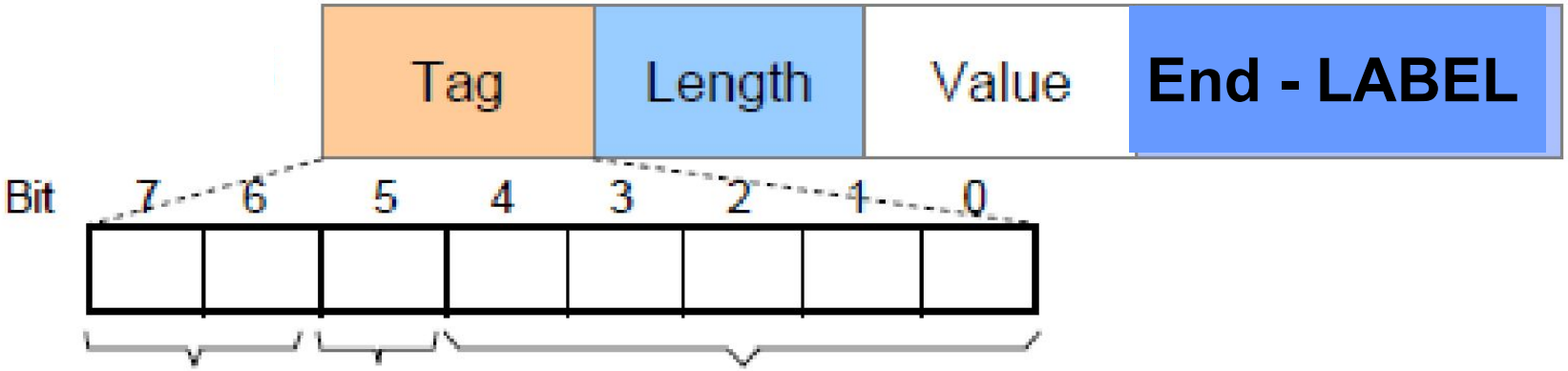
```
Interface ::= [APPLICATION 0] IMPLICIT SET {
     index    [0] INTEGER,
     descr        IA5STRING,
     type     [1] INTEGER{other(0),ethernetCsmacd(6)},
     addr         InterfaceAddress,
     data         InterfaceData
}

InterfaceData ::= [APPLICATION 2] IMPLICIT SEQUENCE {
     packetsIn        [0] IMPLICIT INTEGER,
     packetsOut       [1] IMPLICIT INTEGER,
     packetsTotal     [2] IMPLICIT INTEGER OPTIONAL
}
```

Value assignments

LAN – Network Card **Interface** : ::= {

```
        index   0,
        descr   "3Com"
        type    6,
        data    {
                packetsIn     521,
                packetsOut    130,
        }
}
```

BER - кодиране



Type
00: Universal
01: Application
10: Context Specific
11: Private

Datatype
0: simple
1: structured

Tag Number
0 .. 30
31: next Byte gives Tag Number

BER - кодиране

Interface	= 01|1|00000	(Appl. Spec., Constructed, Implicit Tag 0)	=	**60**
index	= 10|1|00000	(Context spec., Constructed, Explicit Tag 0)	=	**A0**
	= 00|0|00010	(UNIVERSAL, Primitive, Integer)	=	**02**
descr	= 00|0|10110	(UNIVERSAL, Primitive, IA5STRING)	=	**16**
type	= 10|1|00001	(Context spec., Constructed, Explicit Tag 1)	=	**A1**
	= 00|0|00010	(UNIVERSAL, Primitive, Integer)	=	**02**
data	= 01|1|00010	(Appl. Spec., Constructed, Implicit Tag 0)	=	**62**
packetsIn	= 10|0|00000	(Context spec., Primitive, Implicit Tag 0)	=	**80**
packetsOut	= 10|0|00001	(Context spec., Primitive, Implicit Tag 1)	=	**81**

BER - кодиране

```
LAN – Network Card              Interface ::= {
                                  60   19
                index  0,
                  A0   03   02   01   00
                descr  "3Com"
                  16   04   33 43 6F 6D
                type   6,
                  A1   03   02   01   06
                data   {
                  62   07
                       packetsIn     521,
                         80   02   02 09
                       packetsOut    130,
                         81   01   82
                }
}
```

BER - кодиране