HyperText Transfer Protocol

- HTTP - just a different file transfer protocol
- Simplified, standard client/server model
  - just one connection for commands and data
- Data transferred is structured
  - Similar to MIME
    - Content-Type
- Nothing Hyper about the protocol
  - except HTML is a common file format to transfer
    - HyperText Markup Language
  - originally the only file format transferred

HTTP Commands

- Multi-line format
  - Command
    - typically GET (also HEAD, POST)
  - Headers
    - Additional information for the server
      - Authorisation
      - Conditional
      - Pragma
      - Software Identifier
  - Empty Line
  - Data
    - POST only

HTTP Responses

- Status line
  - contains 3 digit code, like SMTP, FTP
- Headers
  - Content-Type
  - Expires
  - Last-Modified
  - ...
- Empty Line
- Body
- <close connection>
  - Optional for new client/server
  - Provided body size is known
    - Content-Length header

HTTP Problem

- Virtual Hosts
  - Want one host to appear to be many servers
    - Server web pages for many organisation
  - Client looks up address of desired server
    - connects to that address
  - Server sees connection arriving at its address
    - Which hostname lookup produced that address?
- Initial solution
  - Different address for every virtual host
    - All reach the same computer
    - It can see what address as used
      - knows which web pages to provide
  - Uses many addresses

Extended HTTP

- Client tells server which hostname it used
  - New header value
  - Easy to add into HTTP format
    - Server now knows which virtual host is wanted
- But
  - Still needs to deal with old clients
  - Must have a default host

Network Management

- Small Network
  - Can connect to each node
    - Routers...
  - Query it
  - Configure it
- Large Network
  - Too many routers
  - Need automated management
- SNMP
  - Simple Network Management Protocol

SNMP

- Protocol to exchange management information
- Definition of information database
  - Management Information Base
- Protocol allows
  - Fetching information
  - Storing information
    - Into MIB
- Storing allows configuration
- Fetching allows monitoring

RFC3410
RFC3419
RFC2578
RFC2580
RFC1213

Defining SNMP

- Abstract Syntax Notation One (1)
  - ASN.1
- Language to define data types
  - Describes self-defining data
    - Recipient can work out what the data types are
    - Doesn't have to simply know
- Used for
  - SNMP protocol packets
  - MIB data structures
- Defined in BNF type grammar

ASN.1

- From CCITT (ITU) originally
  - Part of X.400 (e-mail) standards
  - Presentation layer
- Now ISO standard
- Simple data types
  - Integer
  - Real number
  - Boolean
  - Character string
    - Any character set possible
  - Octet String
  - Bit string
  - Object Identifier OID
  - . . .

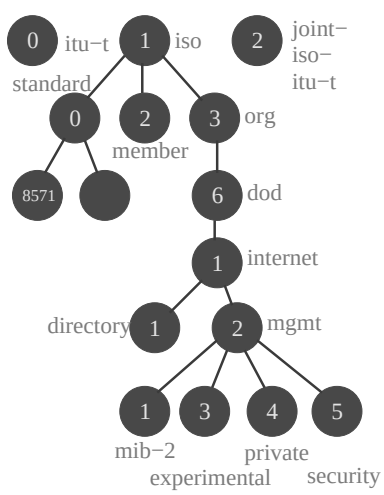
ASN.1 (2)

- Constructed data types
  - Enumerated Types
  - Sequences
  - Sets
  - Choice
  - Constrained
  - Tagged
- Definitions
  - Modules
  - Include
  - Import

Object Identifiers

- Namespace
  - defines a set of names
- Tree structured
- Numeric
  - With names for readability
- Upper levels of tree define registration authority
  - Authority assigns branches to organisations
  - Organisations assign branches to projects
  - ...
- mib-2 .1.3.6.1.2.1
  - .iso.org.dod.internet.mgmt.mib-2
    - .iso.org.dod.internet.mgmt.experimental (3)
    - .iso.org.dod.internet.mgmt.private (4)
    - .iso.org.dod.internet.mgmt.security (5)

OID tree



- ITU-T (CCITT)
  - Phone company
- ISO
  - Member
    - Country
      - 3 digit code
- Joint
  - ASN.1
  - ...

SMI & Textual Conventions

- Notational aides
  - Defined in ASN.1
  - Simplify other definitions
    - Think macros
      - or procedures
    - for data types

```
-- indistinguishable from INTEGER,
-- but never needs more than
-- 32-bits for a two's complement representation
Integer32 ::=
      INTEGER (-2147483648..2147483647)
```

SMI & Textual Conventions (2)

```
ObjectSyntax ::=
    CHOICE {
        simple
            SimpleSyntax,
          -- note that SEQUENCEs for conceptual tables and
          -- rows are not mentioned here...
        application-wide
            ApplicationSyntax
    }

SimpleSyntax ::=
    CHOICE {
        -- INTEGERs with a more restrictive range
        -- may also be used
        integer-value          -- includes Integer32
            INTEGER (-2147483648..2147483647),
        -- OCTET STRINGs with a more restrictive size
        -- may also be used
        string-value
            OCTET STRING (SIZE (0..65535)),
        objectID-value
            OBJECT IDENTIFIER
    }
```

SMI & Textual Conventions (3)

```
ApplicationSyntax ::=
    CHOICE {
        ipAddress-value
            IpAddress,
        counter-value
            Counter32,
        timeticks-value
            TimeTicks,
        arbitrary-value
            Opaque,
        big-counter-value
            Counter64,
        unsigned-integer-value  -- includes Gauge32
            Unsigned32
    }
-- in network-byte order
```

Defining the protocol

- Protocol Data Units (PDU)

```
PDUs ::= CHOICE {
     get-request      GetRequest-PDU,
     get-next-request GetNextRequest-PDU,
     get-bulk-request GetBulkRequest-PDU,
     response         Response-PDU,
     set-request      SetRequest-PDU,
     inform-request   InformRequest-PDU,
     snmpV2-trap      SNMPv2-Trap-PDU,
     report           Report-PDU }

-- PDUs
GetRequest-PDU ::= [0] IMPLICIT PDU
GetNextRequest-PDU ::= [1] IMPLICIT PDU
Response-PDU ::= [2] IMPLICIT PDU
SetRequest-PDU ::= [3] IMPLICIT PDU
-- [4] is obsolete
GetBulkRequest-PDU ::= [5] IMPLICIT BulkPDU
InformRequest-PDU ::= [6] IMPLICIT PDU
SNMPv2-Trap-PDU ::= [7] IMPLICIT PDU
Report-PDU ::= [8] IMPLICIT PDU
```

PDU definition

```
PDU ::= SEQUENCE {
       request-id INTEGER (-214783648..214783647),
       error-status                -- sometimes ignored
           INTEGER {
               noError(0),
               tooBig(1),
               noSuchName(2),      -- for proxy compatibility
               badValue(3),        -- for proxy compatibility
               readOnly(4),        -- for proxy compatibility
               genErr(5),
               noAccess(6),
               wrongType(7),
               wrongLength(8),
               wrongEncoding(9),
               wrongValue(10),
               noCreation(11),
               inconsistentValue(12),
               resourceUnavailable(13),
               commitFailed(14),
               undoFailed(15),
               authorizationError(16),
               notWritable(17),
               inconsistentName(18)
           },
       error-index                 -- sometimes ignored
           INTEGER (0..max-bindings),
       variable-bindings           -- values are sometimes ignored
           VarBindList
   }
```

Variable Bindings

```
VarBind ::= SEQUENCE {
        name ObjectName,
        CHOICE {
            value          ObjectSyntax,
            unSpecified    NULL,
                        -- in retrieval requests
                        -- exceptions in responses
            noSuchObject   [0] IMPLICIT NULL,
            noSuchInstance [1] IMPLICIT NULL,
            endOfMibView   [2] IMPLICIT NULL
        }
    }

-- variable-binding list
VarBindList ::= SEQUENCE (SIZE (0..max-bindings))
                    OF VarBind
```

SNMP Protocol Operations

```
GetRequest-PDU ::= [0] IMPLICIT PDU
GetNextRequest-PDU ::= [1] IMPLICIT PDU
Response-PDU ::= [2] IMPLICIT PDU
SetRequest-PDU ::= [3] IMPLICIT PDU
GetBulkRequest-PDU ::= [5] IMPLICIT BulkPDU
InformRequest-PDU ::= [6] IMPLICIT PDU
SNMPv2-Trap-PDU ::= [7] IMPLICIT PDU
Report-PDU ::= [8] IMPLICIT PDU
```

- Get
  - Fetch value of MIB variable
    - OID given in request
- GetNext
  - Fetch value of MIB variable
    - previous OID given in request
    - OID of variable returned in response
- Set
  - Set the value of a MIB variable
    - OID given in request

Management Information Base

```
RFC1213-MIB DEFINITIONS ::= BEGIN

IMPORTS
        mgmt, NetworkAddress, IpAddress, Counter, Gauge,
                TimeTicks
            FROM RFC1155-SMI
        OBJECT-TYPE
                FROM RFC-1212;

--  This MIB module uses the extended OBJECT-TYPE macro as
--  defined in [14];

--  MIB-II (same prefix as MIB-I)

mib-2       OBJECT IDENTIFIER ::= { mgmt 1 }

-- groups in MIB-II

system      OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces  OBJECT IDENTIFIER ::= { mib-2 2 }
at          OBJECT IDENTIFIER ::= { mib-2 3 }
ip          OBJECT IDENTIFIER ::= { mib-2 4 }
icmp        OBJECT IDENTIFIER ::= { mib-2 5 }
tcp         OBJECT IDENTIFIER ::= { mib-2 6 }
udp         OBJECT IDENTIFIER ::= { mib-2 7 }
```

MIB Variables

```
sysDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "A textual description of the entity.  This value
            should include the full name and version
            identification of the system's hardware type,
            software operating-system, and networking
            software.  It is mandatory that this only contain
            printable ASCII characters."
    ::= { system 1 }

sysObjectID OBJECT-TYPE
    SYNTAX  OBJECT IDENTIFIER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The vendor's authoritative identification of the
            network management subsystem contained in the
            entity.  This value is allocated within the SMI
            enterprises subtree (1.3.6.1.4.1) and provides an
            easy and unambiguous means for determining 'what
            kind of box' is being managed.  For example, if
            vendor 'Flintstones, Inc.' was assigned the
            subtree 1.3.6.1.4.1.4242, it could assign the
            identifier 1.3.6.1.4.1.4242.1.1 to its 'Fred
            Router'."
    ::= { system 2 }
```

Interface Variables

```
ifNumber OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of network interfaces (regardless of
            their current state) present on this system."
    ::= { interfaces 1 }

ifTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "A list of interface entries.  The number of
            entries is given by the value of ifNumber."
    ::= { interfaces 2 }

ifEntry OBJECT-TYPE
    SYNTAX  IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "An interface entry containing objects at the
            subnetwork layer and below for a particular
            interface."
    INDEX   { ifIndex }
    ::= { ifTable 1 }
```

Interface Table

```
IfEntry ::=
    SEQUENCE {
        ifIndex           INTEGER,
        ifDescr           DisplayString,
        ifType            INTEGER,
        ifMtu             INTEGER,
        ifSpeed           Gauge,
        ifPhysAddress     PhysAddress,
        ifAdminStatus     INTEGER,
        ifOperStatus      INTEGER,
        ifLastChange      TimeTicks,
        ifInOctets        Counter,
        ifInUcastPkts     Counter,
        ifInNUcastPkts    Counter,
        ifInDiscards      Counter,
        ifInErrors        Counter,
        ifInUnknownProtos Counter,
        ifOutOctets       Counter,
        ifOutUcastPkts    Counter,
        ifOutNUcastPkts   Counter,
        ifOutDiscards     Counter,
        ifOutErrors       Counter,
        ifOutQLen         Gauge,
        ifSpecific        OBJECT IDENTIFIER
    }
```

Interface Table Entries

```
ifIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only        STATUS  mandatory
    DESCRIPTION
            "A unique value for each interface.  Its value
            ranges between 1 and the value of ifNumber.  The
            value for each interface must remain constant at
            least from one re-initialization of the entity's
            network management system to the next re-initialization."
    ::= { ifEntry 1 }

ifDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only        STATUS  mandatory
    DESCRIPTION
            "A textual string containing information about the
             interface.  This string should include the name of the
             manufacturer, the product name and the version ..."
    ::= { ifEntry 2 }

ifAdminStatus OBJECT-TYPE
    SYNTAX  INTEGER {
                up(1),       -- ready to pass packets
                down(2),
                testing(3)   -- in some test mode
            }
    ACCESS  read-write       STATUS  mandatory
    DESCRIPTION
            "The desired state of the interface.  The testing(3) state
             indicates that no operational packets can be passed."
    ::= { ifEntry 7 }
```

Interfaces Table

	Index	Descr	Type	Mtu	Speed	PhysAddress	AdminStatus	...
0	0	Text	Ether	1500			up	
1	1	Text	Loop	30000			up	
2	2	Text	FDDI	8400			down	
3	3	Text	Token Ring	1500			up	
4	4	Text	PPP	1500			up	
5	5	Text	xxx	1280			testing	
6	6	Text	yyy	576			up	

- MIB has many tables
  - All similar to this
  - Defined number of columns
    - Variable name selects a column
  - Variable number of rows
    - Row index selects a row
  - Combination
    - Var name, Index
    - selects a value

Accessing MIB Variables

- PDU contains VarBinding
- VarBinding contains OID
- OID is:
  - Identifier of MIB variable
  - Index
- For simple variables
  - Index = 0
- For table variables
  - Index is value of index variable

`.1.3.6.1.2.1.1.1.0`
... mib-2 system sysDescr (no index)

`.1.3.6.1.2.1.2.2.1.7.3`
... mib-2 interfaces ifTable ifEntry ifAdminStatus (index) 3

Table Indices

- Index may be
  - simple number
  - IP address
    - a.b.c.d (4 values in OID)
  - character string
    - "abc" -> 95.96.97
  - combination
    - multiple indices
- Index value always appears in table

Encoding Rules

- ASN.1 specifies abstract data types
- Encoding rules
  - specify how to encode data into byte stream
- Basic Encoding Rules
- Packed Encoding rules
  - Details unimportant here
  - But encoding is not trivial

Advantages of ASN.1

- ASN.1 compilers exist
  - Convert ASN.1 specification to C data structure
  - And produce code to format packets
    - Encoding Rules
- Reduces chances of implementation error
  - Also allows specification to be verified
- Produces efficient (small) binary encoding
  - Integers use minimum bytes
  - Fully self defining
    - can tell what type a variable is
    - application does not need to know what type to expect
- Easily extensible

Disadvantages of ASN.1

- Complex to understand
- CPU expensive to pack/unpack
- Difficult to debug
  - Hard to decode packets manually
- Numeric coding
  - requires number<->name table
    - for meaningful results
- SIMPLE Network Management Protocol
  - Not very simple
  - And there is much more
    - security...