

The Application layer

Mattias Jansson, KTHNOC/SUNET
fimblo@sUNET.se

January 29th, 2004

Contents

1	Application Protocols	4
1.1	Binary Fixed Field	6
1.1.1	Example: DNS	7
1.1.2	Pros & Cons	9
1.2	Tree Based	10
1.2.1	TLV	11
1.2.2	Abstract Syntax Notation # 1	12
1.2.3	Pros & Cons	14
1.3	ABNF based text protocols	15
1.3.1	Pros & Cons	18
2	Some application level protocols	19

2.1	Some testing programs	20
2.1.1	Ping	21
2.1.2	telnet	25
2.1.3	nc	27
2.2	HTTP	28
2.2.1	HTTP - example	29
2.2.2	Some HTTP commands	31
2.2.3	Some HTTP status codes	32
2.3	FTP	34
2.3.1	FTP - connection types	35
2.3.2	FTP modes	36
2.3.3	FTP - example	37
2.3.4	Some FTP commands	39

2.3.5	Some FTP status codes	40
2.4	SMTP	41
2.4.1	SMTP example	42
2.4.2	SMTP syntax	43
2.5	MUA protocols	44
2.5.1	Classic email format	44
2.5.2	Multipurpose Internet Mail Extensions	46
2.5.3	So how does it work?	48
2.5.4	Content-Type	49
2.5.5	Content-Transfer-Encoding	50
2.5.6	Message Header Extensions	52

1 Application Protocols

The application layer contains protocols which enable applications to talk to each other.

For example, there is an application protocol (HTTP) which both the web browser and the web server adheres to (and implements), enabling them to transfer data between them.

Application Protocols

Common types of Application Protocol message formats:

- Binary Fixed Field protocols
- Tree based protocols
- ABNF text protocols

1.1 Binary Fixed Field

A binary fixed field protocol predefines exactly *what* information is to be *where* in the message. Binary fixed field protocols are most common in the underlying layers of the TCP/IP stack.

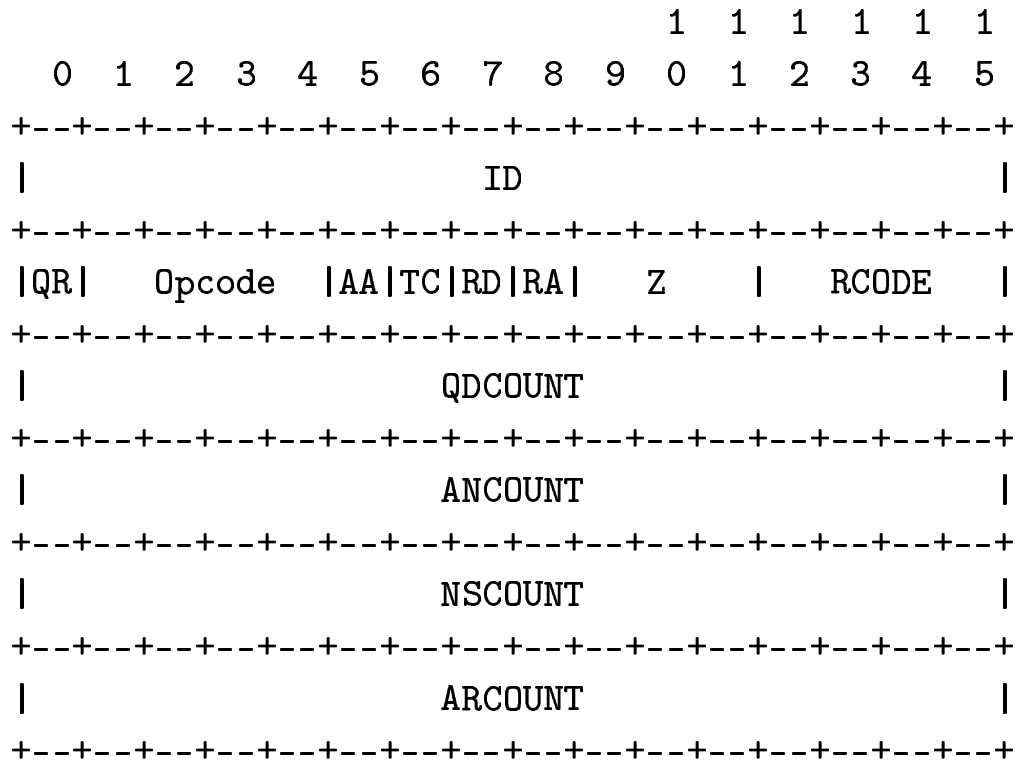
In these protocols, the semantics are hard-coded into the application^a.

Two examples of a binary fixed field protocol residing on the application layer would be DNS and BGP.

^aThus the need for reserved sections.

1.1.1 Example: DNS

The DNS header, taken from RFC 1035.



Example: DNS (con't)

ID A 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.

QR A one bit field that specifies whether this message is a query (0), or a response (1).

OPCODE A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response.

... and so on.

1.1.2 Pros & Cons

Depending on the application in focus, a binary fixed field protocol could be appropriate.

- Pros:
 - Compact: Smaller payloads, faster transmission time, smaller memory usage per message, etc
 - Fixed syntax and simple semantics, therefore easy for applications to digest.
- Cons:
 - Its fixed...so its hard to extend!
 - Not human readable.
 - big/little endian problems

1.2 Tree Based

- Tree based protocols are built in a tree structure (*Surprise!*)
- They come in both text and binary flavours.
- The semantics are described outside of the application. eg an XML schema
- They tend to be recursive in structure.
- Some major types: TLV, ASN.1, and XML.

1.2.1 TLV

- *Type*: contains a predefined code, indicating what kind of data the *value* field contains.
- *Length*: Contains the size (in bytes) of the *value* field.
- *Value*: Contains the payload.

IS-IS and OSPFv3 are both examples of protocols using TLVs.

1.2.2 Abstract Syntax Notation # 1

A standard used to define PDU formats (Protocol Data Unit)

Used frequently in ISO protocols, but also to a certain extent in TCP/IP protocols.

Some examples: SNMP, UMTS, LDAP, NFSv4 and many security protocols.

Here an example of a tiny part of the SNMP definition:

```
PDU ::=
  SEQUENCE {
    request-id Integer32,
    error-status INTEGER (
      noError(0),
      tooBig(1),
      noSuchName(2),
      badValue(3),
      readOnly(4),

      ...

      inconsistentName(18)
    ),
    error-index INTEGER (0..max-bindings),
    variable-bindings VarBindList
  }
```

1.2.3 Pros & Cons

- Pros:
 - Easy to extend.
 - Flexible
- Cons:
 - Can easily become very complex.
 - Demands some overhead, could be silly to use if the application is small.

1.3 ABNF based text protocols

- Augmented Backus-Naur Form
- Defines the syntax and semantics of a protocol
- RFC 2234 defines ABNF, but it appears in (slightly different forms in earlier RFCs (822, 2068, etc)

ABNF based text protocols (con't)

For example:

<code>name = elements crlf</code>	a rule
<code>crlf = %d13.d10</code>	characters to end a line
<code>“literal”</code>	a string, case insensitive
<code>element1 / element2</code>	an alternative
<code>(element1 element2)</code>	a strict sequence
<code>DIGIT = %x30-39</code>	a range of characters
<code><a>*element</code>	element repetition
<code>[foo bar]</code>	optional elements

...and more ...

ABNF based text protocols (con't)

Another example; In RFC 2048, the HTTP URL is defined as:

`http_URL` = `“http:” “//” host [“:” port] [abs_path]`

`host` = A legal Internet host domain name or IP address (in dotted-decimal form) as defined by Section 2.1 of RFC 1123

`port` = `*DIGIT`

`abs_path` = `“/” rel_path`

`rel_path` = `[path] [“;” params] [“?” query]`

1.3.1 Pros & Cons

- Pros:
 - Easy to extend.
 - Flexible.
 - Human readable (easy to debug)
- Cons:
 - Not compact.
 - Demands some overhead, could be silly to use if the application is small.

2 Some application level protocols

- HTTP
- FTP
- SMTP
- MUA protocols

2.1 Some testing programs ...

- Ping
- Telnet
- nc

2.1.1 Ping

There are many different implementations of ping, but they are all used to see if a remote host is alive. Some examples:

- ICMP ping
- UDP ping
- TCP ping
- ...

ICMP Ping

Generally speaking, when you icmp ping a remote host, an ICMP ECHO_REQUEST is sent to the destination, and you should receive an ICMP ECHO_RESPONSE from the remote host.

ICMP Ping (con't)

An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes of ICMP header. If there is space left over, a time stamp is placed after the ICMP header.

When the destination host receives the ECHO_REQUEST, it generates a ECHO_RESPONSE packet with the original ping packet as the payload.

The originating host can, using the initial time stamp and the current time calculate the RTT (Round Trip Time).

UDP Ping

UDP ping works by sending a UDP datagram to an invalid port address at the remote host. If the remote host is alive, it will return an ICMP `PORT_UNREACHABLE` to the source ip address.

As in ICMP ping, a timestamp can be put into the payload of the UDP datagram, so that the source host can calculate the RTT when receiving the returning ICMP packet.

2.1.2 telnet

`telnet` is an application which enables you to establish a connection with other computers, and optionally gives you a command prompt at the other end (in which case it connects to port 23).

Telnet is text-based, and runs over TCP, thus it is a nice tool to test text-based protocols (HTTP, SMTP, FTP, etc)

Nagle's algorithm

Telnet takes advantage of TCP's Nagle algorithm, which in its simplest form says that it is better to send one large datagram than several small ones.

If the risk of losing transferred data is $n\%$ and we send 100 datagrams, we will (assuming the percentage holds) lose n datagrams, which will then have to be retransmitted.

If we had instead sent one large datagram, it will most probably survive the journey to the remote system.

All well-written applications should implement Nagle's algorithm.

2.1.3 nc

nc, pronounced “netcat” is the all-purpose swiss army knife of the Transport layer. It can talk both TCP and UDP, and can also act as a TCP or UDP server.

2.2 HTTP

(Text based, TCP, RFC 2616)

The **Hypertext Transfer Protocol** is the main protocol used to download resources from the world wide web.

In its simplest form, the requestor establishes a TCP connection to the web server on port 80 and sends a string describing what resource it wants, and receives the resource in reply. The session is held in clear text.

The most modern version today is HTTP/1.1.

2.2.1 HTTP - example

In the good old days...

```
fimblo@waoh:~> nc zipf.pilsnet.sunet.se 80
-> GET http://zipf.pilsnet.sunet.se/stuff/blah.html
<- <html>
<-   <b>
<-     blah
<-   </b>
<- </html>
```

HTTP - example (con't)

Today...

```
fimblo@waoh:~> nc zipf.pilsnet.sunet.se 80
-> GET /stuff/blah.html HTTP/1.1
-> Host: zipf.pilsnet.sunet.se
-> User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.5) Gecko/20031214 Firebird/0.7
-> Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.7
-> Accept-Language: en-us,en;q=0.5
-> Accept-Encoding: gzip,deflate
-> Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
-> Keep-Alive: 300
-> Connection: keep-alive

<- HTTP/1.1 200 OK
<- Date: Tue, 27 Jan 2004 20:18:28 GMT
<- Server: Apache/1.3.27 (Unix) (Gentoo/Linux) PHP/4.3.4
<- Last-Modified: Tue, 27 Jan 2004 19:53:47 GMT
<- ETag: "bb4047-2c-4016c1cb"
<- Accept-Ranges: bytes
<- Content-Length: 44
<- Keep-Alive: timeout=15, max=100
<- Connection: Keep-Alive
<- Content-Type: text/html
<-
<- <html>
<-   <b>
<-     Hello there
<-   </b>
<- </html>
```

2.2.2 Some HTTP commands

- GET `http_url`: Download an http resource.
- POST `http_url`: Upload data to an http resource.
- DELETE `http_url`: Delete an http resource.

2.2.3 Some HTTP status codes

Any self-respecting protocol has to declare some way for one entity to tell the other information concerning the query. In HTTP, and in many other text-based protocols, a status code is sent as a reply when needed.

There are five groups of status codes:

- Informational 1xx
- Successful 2xx
- Redirection 3xx
- Client Error 4xx
- Server Error 5xx

Some HTTP status codes (con't)

Some examples:

- 200 Ok
- 404 Not found
- 301 Moved Permanently
- 500 Internal Server Error

2.3 FTP

(Text-based, TCP, RFC 959)

The **File Transfer Protocol** enables clients to transfer files between the local host and the remote server, using the reserved ports 20 and 21 (and other non-reserved ports). FTP conducts its sessions in clear text.

FTP uses two channels:

- The control connection
- The data connection

2.3.1 FTP - connection types

The `control connection` is used to exchange commands and their replies. This is a TCP session initiated by the client to the server on port 21.

The `data connection` is used to transfer data in a specified mode and type. The data transferred may be a part of a file, an entire file or a number of files.

2.3.2 FTP modes

FTP can run in two modes- **active mode** and **passive mode**. This refers to whether the ftp server will start the data connection or not.

- **active:** The server will start the TCP session for the data connection, thereby connecting to the client to a port and IP specified by the client.
- **passive:** The server will not start a TCP session. Instead, the client will create a TCP session to the server, to a port and IP specified by the server.

2.3.3 FTP - example

Download a directory listing:

```
fimblo@waoh:~> telnet zipf 21
Trying 192.36.125.132...
Connected to zipf.
Escape character is '^]'.
<- 220 ProFTPD 1.2.9rc2 Server (zipf ftp server) [zipf]
-> USER fimblo
<- 331 Password required for fimblo.
-> PASS blahblahblah
<- 230 User fimblo logged in.
-> PWD
<- 257 "/home/fimblo" is current directory.
-> PORT 192,36,125,134,16,0
<- 200 PORT command successful
-> LIST
<- 150 Opening ASCII mode data connection for file list
-> 226 Transfer complete.
-> QUIT
<- 221 Goodbye.
Connection closed by foreign host.
```

FTP - example (con't)

Download an binary file:

```
fimblo@waoh:~> telnet zipf 21
Trying 192.36.125.132...
Connected to zipf.
Escape character is '^]'.
<- 220 ProFTPD 1.2.9rc2 Server (zipf ftp server) [zipf]
-> USER fimblo
<- 331 Password required for fimblo.
-> PASS blahblahblah
<- 230 User fimblo logged in.
-> TYPE I
<- 200 Type set to I
-> PORT 192,36,125,134,16,0
<- 200 PORT command successful
-> RETR bild.jpg
<- 150 Opening BINARY mode data connection for bild.jpg (29744 bytes)
<- 226 Transfer complete.
-> QUIT
<- 221 Goodbye.
Connection closed by foreign host.
```

2.3.4 Some FTP commands

As mentioned earlier, the commands are sent on the control connection. Here are some examples of FTP commands:

- CWD <arg> Change working directory
- RMD <arg> Remove directory
- PWD Print working directory
- TYPE [I|A|E|L <arg>] Set the data transfer type
- RETR <arg> Download a file.
- STOR <arg> Upload a file.
- LIST Download the current working directory's content list.

2.3.5 Some FTP status codes

As in HTTP, FTP has a variety of status codes:

- **1yz Positive Preliminary reply**
The requested action is being initiated; expect another reply before proceeding with a new command.
- **2yz Positive Completion reply**
The requested action has been successfully completed. A new request may be initiated.
- **3yz Positive Intermediate reply**
The command has been accepted but the requested action is waiting for further information before being completed.
- **4yz Transient Negative Completion reply**
The command was not accepted and the requested action did not take place, but the error condition is temporary and the action may be requested again.
- **5yz Permanent Negative Completion reply**
The command was not accepted and the requested action did not take place.

2.4 SMTP

(Text-based, TCP, RFC 2821)

SMTP, or **Simple Mail Transfer Protocol** is the definitive protocol used to transfer email from hosts to mail servers and also between mail servers. SMTP applications listen to TCP port 25. The entire session is held in clear text.

Note that SMTP is *not* used to download email to the client host, see POPv3 (Post Office Protocol) or IMAPv3 (Interactive Mail Access Protocol) for that.

2.4.1 SMTP example

```
fimblo@waoh:~> nc mail.yanson.org 25
<- 220 mail.yanson.org ESMTP Postfix
-> HELO waoh.pilsnet.sunet.se
<- 250 mail.yanson.org Hello waoh.pilsnet.sunet.se [192.36.125.134], wazzup.
-> MAIL FROM: fimblo@sunet.se
<- 250 Ok
-> RCPT TO: m@yanson.org
<- 250 Ok
-> DATA
<- 354 End data with <CR><LF>.<CR><LF>
-> From: Jultomten <santa.claus@northpole.se>
-> Subject: This is the subject line
->
-> hello there
-> .
<- 250 Ok: queued as 092B2132D
-> QUIT
<- 221 Bye
```

2.4.2 SMTP syntax

Like HTTP and FTP, SMTP has special commands and status codes.

- `HELO <hostname>`: It is polite to introduce oneself.
- `MAIL FROM <email address>`: Sender email address
- `RCPT TO <email address>`: Recipient email address
- `DATA`: Tells the email server that data follows.
- `QUIT`: Immediately close the connection.

The status codes are similar to that of HTTP and FTP.

2.5 MUA protocols

2.5.1 Classic email format

(Text-based, SMTP, RFC 2822)

RFC 2822 specifies the Standard for the Format of ARPA Internet Text Messages. It covers the rules governing the format of email messages.

A message is defined to contain two sections- the *header* and the *body*, or payload. The two are separated by a CRLF (characters 13 followed by 10). i.e.:

```
message = (fields / obs-fields)
fields   = various_fields CRLF
          [CRLF body]
body     =>(*998text CRLF)
          *998text
```

Classic email format (con't)

Email messages must be written in US-ASCII (7-bit).

What does this imply?

2.5.2 Multipurpose Internet Mail Extensions

MIME aims at redefining the format of messages to allow for:

- textual message bodies in character sets other than US-ASCII,
- an extensible set of different formats for non-textual message bodies,
- multi-part message bodies, and
- textual header information in character sets other than US-ASCII

MIME (con't)

MIME is covered in five RFCs: 2045, 2046, 2047, 2048, 2049.

Together, they cover:

- The Format of Internet Message Bodies
- Different Media types
- Message Header Extensions for Non-ASCII Text
- How to register new media types
- MIME Conformance

2.5.3 So how does it work?

A Message consists of *entities*. Each entity consists of a header and a body. The two are separated with a line consisting of nothing but a CRLF. The body can contain other entities.

Some header fields:

- Content-Type
- Content-Transfer-Encoding

2.5.4 Content-Type

This field describes what kind of data the content carries.

```
content := "Content-Type" ":" type "/"  
         subtype *("; " parameter)
```

Some examples of `type` would be: `text`, `audio`, `video`, `application`, `extension-token`, and `multipart`.

`plain` and `html` are two `text` subtypes.

`pdf` and `msword` are two `application` subtypes.

2.5.5 Content-Transfer-Encoding

This field describes how the data is encoded.

```
encoding := "Content-Transfer-Encoding" ":" mechanism
```

```
mechanism := 7bit / 8bit / binary /  
             quoted-printable / base64 /  
             ietf-token / x-token
```

Content-Transfer-Encoding (con't)

7bit	7 bit ASCII
8bit	8 bit ASCII
binary	Data where any sequence of octets whatsoever is allowed.
quoted-printable	Converts from arbitrary domain to 7bit ASCII
base64	Converts from arbitrary domain to 7bit ASCII
ietf-token	An extension token defined by a standards-track RFC and registered with IANA.
x-token	The two characters "X-" or "x-" followed, with no intervening white space, by any token

2.5.6 Message Header Extensions

If you want to write non US-ASCII characters in the Subject field for example, you will need this extension, since the message “on the wire” must be in US-ASCII.

```
encoded-word := "=?" charset "?" encoding "?" encoded-text "=?"
```

```
encoding := "Q" / "B"
```

```
charset := ISO-8859-1 / ISO-2022-JP / ...
```

```
"Q" := quoted-printable
```

```
"B" := base64
```

So the subject line: “*Subject: test å*” could be interpreted to become:

```
Subject: test =?iso-8859-1?Q?=E5?="
```

Header Extensions - Quoted-printable

When the quoted-printable extension is used, the characters which cannot easily be reinterpreted as 7-bit ASCII (e.g. the first 128 characters of iso-8859-1) are printed in hexadecimal preceded with an “=”.

For example:

Source	Quoted-printable
abc	abc
abc å	abc =?iso-8859-1?Q?=E5?="

See the man page (7) `iso_8859-1` for the most common Western European character set.

Header Extensions - base64

The base64 technique concatenates three bytes at a time (24 bits) and splits them into four 6-bit series. Each of these 6-bit numbers are looked up in a table, which returns a “safe” character (“A-Z”, “a-z”, “0-9”, “+”, “/”, “=”) used by US-ASCII.

No more than 98 of these characters may be on one line, followed by a CRLF.