References


Reading

Protocol

Requirements for a "good protocol"

Well defined

Complete

Parsable

Extendable

Available protocol document
### Assignment 3 Protocol

<table>
<thead>
<tr>
<th>Client Command</th>
<th>Server Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>login;screenName:foo;password:bar;;</td>
<td>ok:success;;</td>
</tr>
<tr>
<td>transmitMessage:Hello World;;</td>
<td>ok:success;;</td>
</tr>
<tr>
<td>transmitMessage:Hello \2;;</td>
<td>ok:success;;</td>
</tr>
<tr>
<td>messages;block:1;;</td>
<td>ok:2;</td>
</tr>
<tr>
<td></td>
<td>text:Hello \2:sender:foo:time:02/03/2009 13:29:45;</td>
</tr>
<tr>
<td></td>
<td>text:Hello World:sender:foo:time:02/03/2009 13:29:42;;</td>
</tr>
<tr>
<td>fuss;;</td>
<td>error:Invalid command f;;</td>
</tr>
<tr>
<td>quit;;</td>
<td>ok:quit;;</td>
</tr>
</tbody>
</table>
Every bit of data sent in either direction has to have its place in the protocol description.

Protocol is a Language

Common formal description:
BNF and Augmented BNF

Format of the description language needs to be part of the protocol document.

Examples are important
The protocol must cover all possible situations.

Garbage data
Old client or server (different protocol versions)
Illegal requests
Boundary conditions
Etc.
Both clients and servers are computer programs.

A computer program's IQ is generally 0.

**Design goals**

Distinct information packets or messages

Allow parsing independent of semantics

Consistency

Allow for code reuse

Flexibility
Allow parsing independent of semantics

<table>
<thead>
<tr>
<th>Client foo Command</th>
<th>Server Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>login;nickname:foo;password:foopass;;</td>
<td>ok:success;;</td>
</tr>
<tr>
<td>waitingList;;</td>
<td>ok:1;nickname:bar;;</td>
</tr>
<tr>
<td>startconversation:bar;;</td>
<td>acceptconversation;; (assuming bar accepts)</td>
</tr>
<tr>
<td>message;text:Hello;;</td>
<td>message;text:Message from bar: sender:bar;time: 02/08/2010 20:13:37;;</td>
</tr>
<tr>
<td>quit;;</td>
<td>ok:quit;;</td>
</tr>
</tbody>
</table>

How does the server parse each set of commands?

The client parse each response
Available

Different groups may write clients and servers at different times.

Central registry for Internet protocols

Self regulating:
  RFC - Request For Comment
  IETF - Internet Engineering Task Force

Official:
  ISO
  ANSI
Protocol Types

Typical synchronous

Client sends request to server
Server responds with a reply

HTTP, POP, SMTP, GOPHER, XMODEM

Typical asynchronous

Client and server both send information to each other concurrently.

TELNET, RLOGIN, ZMODEM

A hybrid protocol is also possible
Protocol Design Issues

Protocol design is difficult!
Learn from examples

Some issues

Protocol extendibility and versioning

Byte order used for sending values

ASCII vs. Binary protocol

Synchronous vs. Asynchronous

State

Timeouts
HTTP
HTTP

Stateless (http 1.0)

Assigned port 80

Basic Server-Client Interaction (http 1.0)

Client: Open connection

Server: Accept/Reject connection

Client: Send request

Server: Send response to request

Connection closed
HTTP Message Format

HTTP-message = Simple-Request       (HTTP/0.9 messages)
| Simple-Response
| Full-Request       (HTTP/1.0 messages)
| Full-Response

Full-Request = Request-Line
  *( General-Header | Request-Header | Entity-Header )
  CRLF
  [ Entity-Body ]

Full-Response = Status-Line
  *( General-Header | Request-Header | Entity-Header )
  CRLF
  [ Entity-Body ]

HTTP-header = field-name "::" [ field-value ] CRLF

Entity-Body = *OCTET
HTTP Full Request

Request-Line   = Method SP URI SP HTTP-Version CRLF

rohan 13-> telnet www.eli.sdsu.edu 80
Trying 130.191.226.80...
Connected to www.eli.sdsu.edu.
Escape character is '^[].
GET /courses/fall00/cs580/index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 05 Sep 2000 19:31:14 GMT
Server: Apache/1.3.9 (Unix) PHP/3.0.12
Last-Modified: Mon, 04 Sep 2000 21:03:56 GMT
ETag: "14c199-7e8-39b40e3c"
Accept-Ranges: bytes
Content-Length: 2024
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<HTML>
<HEAD>
   <TITLE>CS 580: Course Web Site</TITLE>
   … stuff removed here…
Connection closed by foreign host.

2 CRLF's end the full request
Positional Data verses Name-Value Pairs

Which is more error prone?
Name-Value Pairs & Orderer

MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT

Server: CERN/3.0
Content-Type: text/html
MIME-Version: 1.0
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT
Date: Thursday, 21-Mar-96 17:00:45 GMT
Adding new Fields

MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Forwarded: by http://rohan.sdsu.edu/ for cs.sdsu.edu
Content-Length: 2686

WhitneyInfo: Hi Mom

Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT
Name-Value Pairs are your Friends
Don't Program without them
How to Indicate the End of a Message

Use termination sequence

Make the length of the message known
HTTP uses both

Header ends in CRLFCRLF
Header contains length in bytes of message body

HTTP/1.0 200 Document follows
MIME-Version: 1.0
Server: CERN/3.0
Date: Thursday, 21-Mar-96 17:00:45 GMT
Content-Type: text/html
Content-Length: 2686
Last-Modified: Tuesday, 27-Feb-96 05:34:12 GMT
Detecting End of a Message

What if the terminating sequence is part of the message?

What if a HTTP header contains CRLFCRLF
POP3
Post Office Protocol

Purpose: Allow PC's, Macs, etc. to download mail from server

Port number 110

Protocol uses ASCII only

Stateful protocol

Multiple requests & responses on same connection
Format of commands to server

keyword blank argument1 [ blank argumentk ] CRLF

| keyword | = 3, 4 characters, no spaces
| argument | <= 40 characters, no spaces

keyword and arguments are separated by single space character
Server Response

Status keyword additionalInfo

Status is either "+OK" or "-ERR0.3."

A single line response ends in CRLF

If response requires more than one line:

Each line ends in a CRLF
The response ends in CRLF.CRLF
If a line starts with a "." prepend a "." to it

When Client reads the first CRLF how does it know it is at the end of message?
Timeouts

A POP3 server may have an autologout timer

A server must wait at least 10 minutes before timing out an idle client

The POP3 server on cs.sdsu.edu times out in 2 minutes
Client Connect States

Diagram:

- **Authorization**
  - User/Pass success
  - Quit
  - User/Pass fail

- **Transaction**
  - Quit
  - Stat
  - List
  - Retr
  - Rset

- **Update**
  - Stat
  - List
  - Retr
  - Rset

- **Close Connection**
Authorization State

Server acknowledges connection from client with

   +OK "message"

   +OK UCB Pop server (version 2.1.2-R3) at sciences.sdsu.edu starting.

Commands: USER, PASS, APOP, QUIT
**USER PASS**

Combination is used to progress to transaction state

USER must come first
PASS or QUIT must come after USER

Example
Ti 38->**telnet** **cs.sdsu.edu** 110
Trying 130.191.226.116...
Connected to cs.sdsu.edu.
Escape character is '\^[].
+OK QPOP (version 3.1.2) at sciences.sdsu.edu starting.
**USER** whitney
+OK Password required for whitney.
**PASS** typeYourPasswordHere
+OK whitney has 116 visible messages (0 hidden) in 640516 octets.
Transaction State

Commands: STAT, LIST, RETR, RSET, QUIT

STAT
Arguments: none
Returns "+OK" numberOfMessages SizeOfMail

STAT
+OK 22 45595

LIST
Arguments: a message-number (optional)
Returns: size of message in octets

Examples

LIST
+OK 116 visible messages (640516 octets)
1 2980
2 3064 (message 3 - 116 deleted to save space)
116 1290
.

LIST 2
+OK 2 3064
Transaction State

RETR 21
+OK 825 octets
Received: from [130.191.9.18] (ebb2p9.sdsu.edu [130.191.9.18]) by sciences.sdsu.edu (4.1/8.6.10) with SMTP id UAA29486 for <whitney@saturn.sdsu.edu>; Mon, 11 Mar 1996 20:16:07 -0800 (PST)
X-Sender: whitney@cs.sdsu.edu (Unverified)
Message-Id: <v021101100ad6aaaf097b6@[130.191.9.70]>
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
Date: Mon, 11 Mar 1996 20:16:50 -0800
To: whitney@saturn.sdsu.edu
From: whitney@saturn.sdsu.edu (Roger Whitney)
Subject: Sample Mail
X-UIDL: 826604201.000

this is a test
...
the end
---
Roger Whitney | Math & Computer Science Dept.  
whitney@cs.sdsu.edu | San Diego State University  
http://www.eli.sdsu.edu | San Diego, CA 92182-7720
(619) 594-3535  
(619) 594-6746 (fax)
Transaction State

DELE
Arguments: a message-number to delete
Returns: a confirmation of deletion
Marks a message to be deleted

NOOP
Arguments: none
Returns: a positive response
Does nothing

QUIT
Arguments: none
Returns: a positive response
Send POP3 server to UPDATE state

Why NOOP?
Update State

Updates mail box to reflect transactions taken during the transaction state, then logs user out

If session ends by any method except the QUIT command during the transaction state, the update state is not entered
Gnutella
Gnutella

Peer-to-peer

Gnutella program is both a server and a client: servent

No central server

Protocol does not discuss how one knows about other servents
Basic Operation

Servent connects to 1 or more remote servents

Can
  Ping the network
  Send a request for a file to see who has it

To get a file from a servent
  Connect to the servent directly with http request
Basic Protocol

Connect to another servent with

    GNUTELLA CONNECT/<protocol version string>\n\n
Where <protocol version string> is 0.4

If the remote servent accepts the connection it must respond with

    GNUTELLA OK\n\n
Both servents then can then send messages
Requests and Responses

- Ping – who is on the network
- Pong – response to a ping
- Query – search the network for data
- QueryHit – response to query
- Push – Used to allow servants work behind firewall
- Each Request/Response starts with a header
Descriptor ID
16 byte string
Uniquely identifies Request/Response

TTL
Time to live

Number of times message will be forwarded by servents

Many servents will set TTL to 5 if it is larger

Each servent that gets the message reduces TTL by one before forwarding the message
Header

Hops

Number of times message has been forwarded

Each servant that gets the message increase Hop by one before forwarding

Payload Length

Length of rest of message
Ping 0x00

<table>
<thead>
<tr>
<th>Byte offset</th>
<th>Descriptor ID</th>
<th>Payload Descriptor</th>
<th>TTL</th>
<th>Hops</th>
<th>Payload Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22</td>
</tr>
</tbody>
</table>

Descriptor 0x00
Pong 0x01

Sent only in response to a ping

Servent can cache pongs of other servents

Payload

<table>
<thead>
<tr>
<th>Byte offset</th>
<th>Port</th>
<th>IP Address</th>
<th>Number of files shared</th>
<th>Number of kilobytes shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13</td>
</tr>
</tbody>
</table>

Port that responding servent can accept incoming connections

IP Address of responding servent

This field uses big-endian format
## Query 0x08

<table>
<thead>
<tr>
<th>Byte offset</th>
<th>Minimum Speed</th>
<th>Search Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### Minimum Speed

Minimum speed (of connection) in kb/second of servents that should respond to this message

### Search Criteria

Nul (0x00) terminated search string

Length of string must be included in the payload length field
**QueryHit 0x81**

Sent in response to a Query
Descriptor ID in header should contain same value as the Query

<table>
<thead>
<tr>
<th>Description</th>
<th>Byte offset</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>6</th>
<th>7</th>
<th>10</th>
<th>11</th>
<th>…</th>
<th>n</th>
<th>n+16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Payload</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of hits</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result Set</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Servent Identifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Number of hits**

Number of hits in the result set

**Port**

Port number on which responding servent can accept incoming connections

**IP Address**

IP Address of responding servent
This field uses big-endian format

**Speed**

Speed of responding host’s connection in kb/second
## QueryHit 0x81

<table>
<thead>
<tr>
<th>Payload</th>
<th>Number of hits</th>
<th>Port</th>
<th>IP Address</th>
<th>Speed</th>
<th>Result Set</th>
<th>Servent Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte offset</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result Set</th>
<th>File Index</th>
<th>File Size</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte offset</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

### File Index
A number used by host to identify the file

### File Size
Size in bytes of the file

### File Name
Double-nul (0x0000) terminated name of the file

### Servent Identifier
A 16-byte string uniquely identifying the responding servent on the network.

“This is typically some function of the servent’s network address”
Query Example
# Extended Query Hit

<table>
<thead>
<tr>
<th>Payload</th>
<th>Number of hits</th>
<th>Port</th>
<th>IP Address</th>
<th>Speed</th>
<th>Result Set</th>
<th>Trailer</th>
<th>Servent Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte offset</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trailer</th>
<th>Vender Code</th>
<th>Open Data Size</th>
<th>Open Data</th>
<th>Private data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte offset</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

How do we know if the trailer exists?

How do we know the length of the private data?
### Push 0x40

<table>
<thead>
<tr>
<th>Byte offset</th>
<th>Servent Identifier</th>
<th>File Index</th>
<th>IP Address</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
<td>16</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
</tr>
</tbody>
</table>

#### Servent Identifier

A 16-byte string uniquely identifying the servent on the network that should push the file.

#### File Index

Index of the file to push.

#### IP Address

IP Address of to which the file should be pushed.

This field uses big-endian format.

#### Port

Port to which the file should be pushed.
Some Routing
Pong messages
Can only be send along path the carried the Ping
Servents should not forward a pong if they did not see the ping

QueryHit
Can only be send along path the carried the Query
Servents should not forward a query hit if they did not see the query

Push
Can only be send along path the carried the QueryHit
Servents should not forward a push if they did not see the query hit

Fowarding
Forward all Ping and Querys to all directly connected servents except to the one that sent it

Decrement TTL and increment Hops field
Don’t forward messages that you have seen before
File Downloads

In response to a QueryHit download the file by using http.

Request the file uses following format:

GET /get/<File Index>/<File Name>/ HTTP/1.0
Connection: Keep-Alive
Range: bytes=0-
User-Agent: Gnutella

Remote servant responses with:

HTTP 200 OK
Server: Gnutella
Content-type: application/binary
Content-length: fileSize
File Example