## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>2</td>
</tr>
<tr>
<td>Well defined</td>
<td>3</td>
</tr>
<tr>
<td>Complete</td>
<td>4</td>
</tr>
<tr>
<td>Parsable</td>
<td>5</td>
</tr>
<tr>
<td>Available</td>
<td>6</td>
</tr>
<tr>
<td>Protocol Types</td>
<td>7</td>
</tr>
<tr>
<td>Protocol Design Issues</td>
<td>8</td>
</tr>
</tbody>
</table>

## References

CS 580, Spring 1997 lecture notes,
[http://www.eli.sdsu.edu/courses/spring97/cs596/notes/protocol/protocol.html](http://www.eli.sdsu.edu/courses/spring97/cs596/notes/protocol/protocol.html)

*Copyright ©, All rights reserved.*  
2000 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA.  
OpenContent ([http://www.opencontent.org/opl.shtml](http://www.opencontent.org/opl.shtml)) license defines the copyright on this document.
Protocol

Communication between client and server

Good protocols are hard to design

Requirements for a "good protocol":

- Well defined
- Complete
- Parsable
- Extendable
- Available protocol document
Well defined

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.
Parsable

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
  
  For example name-value pairs
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

Two basic types

- Synchronous
- Asynchronous

Typical synchronous

- Client sends request to server
- Server responds with a reply

Examples

HTTP, POP, SMTP, GOPHER, XMODEM

Typical asynchronous

Client and server both send information to each other concurrently.

Examples

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
  
  Easy of debugging
  Efficiency
- Synchronous vs. Asynchronous
  
  Protocol overhead
  Roundtrip delays
- State
  
  Who is writing, who is reading?
- Timeouts
  
  Timeouts vs. Synchronous protocols