APPLICATION PROTOCOL DESIGN
Protocol

• A protocol defines:
  • Message formats
  • Message sequences in communication
  • How to process a message

• Goals
  • Everyone must know
  • Everyone must agree
  • Unambiguous
  • Complete
Example: POP session

C: <client connects to service port 110>
S: +OK POP3 server ready <1896.6971@mailgate.dobbs.org>
C: USER bob
S: +OK bob
C: PASS redqueen
S: +OK bob's maildrop has 2 messages (320 octets)
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
C: <client hangs u>
Example: FTP authentication

>   ftp 202.191.56.65
C: Connected to 202.91.56.65
S: 220 Servers identifying string
User: tungbt (C: USER tungbt)
S: 331 Password required for tungbt
Password: (C: PASS)
S: 230 User tungbt logged in
Issues

- Questions are raised while designing an application protocol:
  - Is it to be stateful vs stateless?
  - Is the transport protocol reliable or unreliable?
  - Are replies needed?
  - Is it to be broadcast, multicast or unicast?
  - Are there multiple connections?
Message format

- Two pieces of data
  - Header: contain message type, describing what type of data in payload
    - Distinguish different type messages.
  - Payload
    - Data

- Message type
  - Short and descriptive type
  - SHOULD has fix length
    - So we can parse the message and understand its type easily
  - Example 1: see POP session
Data Format of messages

• In byte format
  • The first part of the message is typically a byte to distinguish between message types.
  • Further bytes in the message would contain message content according to a pre-defined format
• Advantages: compactness
• Disadvantages: harder to process
• Example: IP message (but IP is not application protocol)
Data Format of messages

• In character format
  • A message is a sequence of one or more lines
  • The start of the first line of the message is typically a word that represents the message type.
  • The rest of the first line and successive lines contain the data.
• Ex: HTTP message
Example: HTTP request

request line (GET, POST, HEAD commands)

GET /dccn/index.html HTTP/1.1
Host: www.it-hut.edu.vn
User-agent: Mozilla/4.0
Connection: close
Accept-language: en-us

(header lines)

CR, LF
indicates end of message

(extra carriage return, line feed)
Example: HTTP response

status line (protocol status code status phrase)

HTTP/1.1 200 OK
Connection close
Date: Tue, 16 Mar 2008 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 15 Mar 2008 ......
Content-Length: 8990
Content-Type: text/html

data, e.g., requested HTML file

data data data data data data data ...
UML Protocol State Machine Diagram

- State: 
- Transaction: Trigger[Precondition]/[Effect]
- Choose: 
- State Table

<table>
<thead>
<tr>
<th>Current state</th>
<th>Transaction</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Receive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Send</td>
<td></td>
</tr>
</tbody>
</table>
Exercise 1

• Define protocol for the application Monolog chat. The “monolog” chat works as follows:
  • Client can send to server either login name or some messages.
  • When client sends login then server accepts and remembers his login
  • When client sends to server a text message, server saves the message into a log file. Each client has a separate log file.

• Hints:
  • Need a message type for login, another one for text message
  • Define fields of each messages and their length/type.
  • Define how client/server processes each message.
  • Draw protocol state machine.
  • Then now you can write code!!!
Exercise 2

• Create a UDP/TCP Client/Server “dialog” chat program that work as follows:
  • Client can send to server either login name or text message
  • When client sends login then server accepts and remembers his login
  • When client sends server a text message for transferring to another client, server sends this message to the client. Server should tell also the receiver that from whom the message come.

• Hints:
  • Started from the protocol of exercice 1 and define new message formats
Exercise 3

• Going back to Student Schedule Management application.
• What was the application protocol using between client and server?
  • Messages
  • Procedure for each functionality
• Working in pair with your classmates.
  • Redefine messages and procedure for each functionality.
  • Revise the server and client accordingly. You revise server and your partner revise the client.
  • Server and client should print on terminal the messages they receive and send.
  • Test the application.
Exercise- sending complex data

• Reuse Echo server and Echo client

• Sending complex data from client to server
  • Send a struct
    • struct{
      • char username[];
      • char password[];
      • int count;
    }
  • Send an array
  • See if server receive well data when client and server are running on two machines
Project

• Create your own HTTP client that can work with existing HTTP server
  • Client can visualize the page web in your own style.
• Online Chinese Chess game for multiple pairs of players
• Chat software with user authentication
  • Option to chat 2 persons
  • Option to chat to all users in the same time
• Puzzle game playing by two persons
• Penalty shoot in football game

• Your proposal for project …
Working plan

- **Week 1: 3/11**
  - Team identification: 3 students/team.
  - Topic identification: Application over internet that you want to develop.
  - Topic Review with lecturer
- **Nov 17th:**
  - Design review, by presentation
    - Application introduction
    - Architecture of the application: client/server, P2P, hybrid
    - Functionality
    - Protocol design (very important): message, state machine, message processing
    - get feedback and revise
- **Nov 24th:**
  - Design review (if necessary)
  - Coding
  - Progress update
Working plan

• **Weeks:**
  • Progress update, by presentation

• **Final test: Final presentation.**
  • Brief introduction with slide about the application, the design
  • Demo
  • Each team has to submit a report (hard copy): including design and application evaluation.
Design presentation

- Application description
- Game rule
- Application architecture (figure)
- Functionality
  - Use case
- Working procedure for each functionality
  - Communication diagram between client/servers, or between clients
- Message design
  - Message formats
  - Message sequences in communication
  - How to process a message